

Tesina in Software Architecture Design

Capemont: gestore di vendite

Anno accademico 2020/2021

Alessio Andreozzi - Mat. M63/1159

Biagio Grasso - Mat. M63/1191

Michele Armida - Mat. M63/1111

Vincenzo Carandente - Mat. M63/1229

20 October 2021

Contents

1	Presentazione del progetto	4
2	Descrizione del progetto	6
2.1	Processo di sviluppo	6
2.2	Gestione del team	7
2.3	Riunioni di progetto	8
2.4	Stima dei costi di sviluppo	8
2.5	Workshop	9
2.5.1	Primo workshop	9
2.5.2	Secondo workshop	10
2.5.3	Terzo Workshop	10
2.5.4	Quarto Workshop	10
2.5.5	Quinto Workshop	10
2.6	Strumenti utilizzati	10
2.6.1	Visual Paradigm	10
2.6.2	IntelliJ	11
2.6.3	Trello	11
2.6.4	Microsoft Teams	11
2.6.5	Overleaf	12
2.6.6	PostgreSQL	12
2.6.7	pgAdmin	12
2.6.8	JUnit	13

3	Glossario dei termini	14
4	Specifica dei requisiti	16
4.1	Descrizione testuale dei requisiti	16
4.2	Descrizione generale	16
4.2.1	Prospettive del prodotto	16
4.2.2	Opportunità di business	16
4.2.3	Demografia di mercato	16
4.3	Descrizione delle parti di interesse	16
4.4	Obiettivi di alto livello e problemi delle parti di interesse	18
4.5	Requisiti funzionali	19
4.6	Requisiti non funzionali	19
4.6.1	Sicurezza	19
4.6.2	Usabilità	19
4.6.3	Affidabilità	20
4.6.4	Performance	20
4.6.5	Supportabilità	20
4.7	Regole di business	20
5	Analisi dei requisiti	21
5.1	Identificazione degli attori	21
5.2	Use case diagrams	21
5.3	Descrizione breve dei casi d'uso	21
5.4	Tabella Attori - Obiettivi	22
5.5	Descrizione di dettaglio dei casi d'uso	23
5.5.1	Caso d'uso 1: Inserisci prodotto	23
5.5.2	Caso d'uso 2: Visualizzazione catalogo prodotti	24
5.5.3	Caso d'uso 3: Visualizzazione prodotti più acquistati	25
5.5.4	Caso d'uso 4: Visualizzazione prodotti più recenti	26
5.5.5	Caso d'uso 5: Visualizzazione prodotti propri	27
5.5.6	Caso d'uso 6: Visualizzazione storico delle vendite	28
5.5.7	Caso d'uso 7: Effettua ordine	29
5.5.8	Caso d'uso 8: Consulenza sui prodotti	30
5.5.9	Caso d'uso 9: Fornitura del prodotto	31
5.5.10	Caso d'uso 10: Registrazione Utente	32
5.6	System domain model	33
5.7	Sequence diagrams di analisi	34
5.7.1	Sequence diagram: Inserisci prodotto	34
5.7.2	Sequence diagram: Visualizzazione catalogo prodotti	34
5.7.3	Sequence diagram: Visualizzazione prodotti più acquistati	35
5.7.4	Sequence diagram: Visualizzazione prodotti più recenti	35
5.7.5	Sequence diagram: Visualizzazione prodotti propri	36
5.7.6	Sequence diagram: Visualizzazione storico delle vendite	36
5.7.7	Sequence diagram: Effettua ordine	37
5.7.8	Sequence diagram: Consulenza sui prodotti	38
5.7.9	Sequence diagram: Fornitura del prodotto	38

5.7.10	Caso d'uso 10: Registrazione Utente	39
5.8	Activity diagrams	40
5.8.1	Activity diagram: Inserisci prodotto	40
5.8.2	Activity diagram: Visualizzazione catalogo prodotti . . .	41
5.8.3	Activity diagram: Visualizzazione prodotti più acquistati	41
5.8.4	Activity diagram: Visualizzazione prodotti più recenti . .	42
5.8.5	Activity diagram: Visualizzazione prodotti propri	43
5.8.6	Activity diagram: Visualizzazione storico delle vendite . .	43
5.8.7	Activity diagram: Effettua ordine	44
5.8.8	Activity diagram: Consulenza sui prodotti	45
5.8.9	Activity diagram: Fornitura del prodotto	46
5.8.10	Activity diagram: Registrazione Utente	47
6	Architettura software	48
6.1	Viste architetturali di alto livello	48
6.2	Component diagram	49
6.3	Descrizione dell'architettura di dettaglio	50
6.3.1	Diagrammi architetturali di dettaglio lato client	51
6.3.2	Diagrammi architetturali di dettaglio lato server	52
6.3.3	Sequence diagrams di dettaglio	54
6.3.4	Sequence diagram: Inserisci prodotto	54
6.3.5	Sequence diagram: Effettua Ordine	55
6.3.6	Sequence diagram: Register Azienda	56
6.3.7	Sequence diagram: Register Acquirente	56
6.4	Pattern architetturale: Architettura a livelli	57
6.4.1	Caratteristiche dell'architettura	60
6.5	Stile architetturale	61
6.6	Pattern comportamentale Command	62
6.7	Pattern creazionale Singleton	63
6.8	Connettori	64
7	Persistenza dei dati	65
7.1	Definizione delle relazioni	66
7.2	Traduzione in tabelle	66
8	Implementazione	67
8.1	Deployment diagram	67
8.2	Documenti di implementazione	68
8.3	Implementazione dell'applicazione	69
8.3.1	Lato Client: GUI Java	69
8.4	Server remoto	72
9	Testing	73
9.1	Testing di Unità	73
9.2	Testing di integrazione	74
9.3	Testing d'interfaccia	75

1 Presentazione del progetto

Il progetto prevede la realizzazione di un E-commerce. Col termine e-commerce ci si riferisce ad una tipologia di commercio elettronico nella quale il mezzo di realizzazione dell'acquisto è il sito web, e in cui la transazione economica si svolge, appunto, interamente via internet. La consegna del prodotto, invece, si differenzia a seconda delle caratteristiche dello stesso:

- In linea, se in formato elettronico. Per esempio canzoni in formato MP3, programmi, videogiochi, applicazioni (quali le app dell'iPhone o per Android) e altri (spesso questi negozi online fanno parte degli store alternativi);
- Trasporto fisico negli altri casi. I negozi in linea si rivolgono generalmente a corrieri espressi (es. FedEx, UPS, TNT, ecc.) in base al mercato da servire e alla velocità di consegna garantita ai clienti.

Il sistema che verrà realizzato prevederà la possibilità di registrarsi sulla piattaforma come azienda o come utente, darà la possibilità di visualizzare i prodotti presenti ed eventualmente acquistarli. Nel processo d'acquisto si darà la possibilità di inserire il codice dell'agente in modo da ricevere una scontistica sull'acquisto; inoltre, i prodotti saranno resi disponibili da un fornitore.

ID Test	Azione	Risultati attesi	Risultati ottenuti
Prima bozza del documento	Ago 20, 2021	Prima scrittura del documento.	Team
Primo raffinamento	Ago 23, 2021	Raffinamento della bozza iniziale.	Team
Aggiornamento 1	Ago 25, 2021	Definizione degli obiettivi del sistema e descrizione dei requisiti funzionali e non funzionali.	Team
Aggiornamento 2	Ago 28, 2021	Dichiarazione dei problemi e descrizione delle parti interessate.	Team
Aggiornamento 3	Ago 31, 2021	Definizione obiettivi di alto livello e problemi delle parti interessate.	Team
Aggiornamento 4	Set 3, 2021	Aggiornamento della descrizione generale del prodotto.	Team
Aggiornamento 5	Set 5, 2021	Descrizione breve dei casi d'uso.	Team
Aggiornamento 6	Set 7, 2021	Raffinamento dei casi d'uso già presenti e inserimento di nuovi casi d'uso utili al funzionamento del sistema.	Team
Aggiornamento 7	Set 10, 2021	Descrizione dettagliata dei casi d'uso.	Team
Aggiornamento 8	Set 12, 2021	Inserimento di altre sezioni per la descrizione generale del prodotto.	Team
Aggiornamento 9	Set 15, 2021	Raffinamento del documento.	Team
Aggiornamento 10	Set 18, 2021	Raffinamento del documento.	Team

Aggiornamento 11	Set 20, 2021	Inserimento di nuove sezioni e aggiunta di grafici rappresentativi del sistema.	Team
Aggiornamento 12	Set 23, 2021	Raffinamento della sezione relativa ai casi d'uso.	Team
Aggiornamento 13	Set 26, 2021	Inserimento dei diagrammi prodotti nella prima iterazione.	Team
Aggiornamento 14	Set 29, 2021	Inserimento dei diagrammi prodotti nella seconda iterazione.	Team
Aggiornamento 15	Ott 1, 2021	Inserimento dei diagrammi prodotti nella terza iterazione.	Team
Aggiornamento 16	Ott 4, 2021	Inserimento dei diagrammi prodotti nella quarta iterazione.	Team
Aggiornamento 17	Ott 7, 2021	Raffinamento del documento e inserimento della sezione relativa alla configurazione.	Team
Aggiornamento 18	Ott 10, 2021	Inserimento della sezione relativa al testing del sistema.	Team
Aggiornamento 19	Ott 13, 2021	Inserimento del capitolo conclusivo e correzione finale del documento.	Team
Aggiornamento 20	Ott 17, 2021	Consegna del software.	Team

2 Descrizione del progetto

2.1 Processo di sviluppo

Per lo sviluppo sono state adottate le metodologie agile in quanto si è scelto un approccio di tipo iterativo ed evolutivo, con iterazioni brevi e timeboxed e una pianificazione adattativa. Le fasi di modellazione e di codifica avvengono di pari passo, in quanto la modellazione non produce modelli da consegnare agli sviluppatori, ma sono loro stessi che si aiutano nel processo di sviluppo tramite i modelli. Il processo di sviluppo viene diviso in tre fasi:

- **Fase iniziale:** è una fase di pianificazione in cui si stabiliscono gli obiettivi generali del progetto e si progetta l'architettura del software;
- **cicli di Sprint:** ogni ciclo sviluppa un incremento del sistema;

- **Fase di chiusura:** al termine dello sviluppo il sistema viene testato e la documentazione completata.

Per lo sviluppo sono state adottate anche diverse tecniche, quali:

- **Facilitated Workshop:** Una pratica a supporto dei principi di comunicazione e collaborazione, unitamente al mantenimento del focus sugli obiettivi di business. Questa tecnica consiste nel prevedere incontri (workshop) facilitati durante il progetto; la presenza di un facilitatore neutrale (facilitator) garantirà il successo del meeting, mantenendo costantemente l'incontro in linea con i suoi obiettivi e il contesto adatto (libertà di parola, assenza di pressioni tra i partecipanti, decisioni non forzate, ecc.), e garantendo che vengano trasmesse a tutte le parti interessate tutte le informazioni necessarie sia precedenti che derivanti (follow-up) dal workshop stesso;
- **Pair Programming:** Lo sviluppo viene fatto da coppie di programmatori che si alternano alla tastiera;
- **Refactoring:** La ristrutturazione di parti di codice mantenendone invariato l'aspetto e il comportamento esterno;
- **Iterative development:** Un'importante pratica attraverso la quale la soluzione da consegnare si evolve da quella che era soltanto "un'idea" (un concetto, una proposta, un insieme di esigenze) fino a divenire un prodotto di valore per il cliente.

2.2 Gestione del team

Il team è composto da quattro membri; Ogni fase di sviluppo è stata suddivisa in task e ad ogni membro del gruppo è stato assegnato uno o più di questi task. Al termine di ogni ciclo, il team si è riunito per discutere dei risultati ottenuti e si è proceduto insieme all'individuazione dei prossimi task. Fondamentale per la gestione del team in tale modo è stato l'uso di **Trello**, che ha permesso al team di monitorare step by step l'andamento dei vari task; e l'uso della piattaforma Microsoft Teams, necessaria per permettere al team di lavorare anche in remoto.

2.3 Riunioni di progetto

Lo sviluppo ha richiesto circa 45 giorni, con un impegno per membro di 6 giorni a settimana.

<i>Attività svolta</i>	<i>Sett.1</i>	<i>Sett.2</i>	<i>Sett.3</i>	<i>Sett.4</i>	<i>Sett.5</i>	<i>Sett.6</i>	<i>Totale ore</i>	<i>Totale ore %</i>
<i>Studio di fattibilità</i>	30						30	3.7%
<i>Analisi requisiti</i>	50	25	20	10			105	13%
<i>Progettazione</i>	45	60	60	45	50	40	300	37.3%
<i>Implementazione</i>		40	40	40	60	60	240	29.8%
<i>Test</i>				30	20	15	65	8%
<i>Gestione documentazione</i>	10	10	15	10	10	10	65	8.2%
<i>Totale ore</i>	135	135	135	135	140	125	805	100%

2.4 Stima dei costi di sviluppo

Per il calcolo dei costi di sviluppo del progetto, ci si è basati sulla tecnica di calcolo degli **Use Case Point**. Dopo un'attenta analisi sui possibili use-case del progetto si è fatta una stima dei differenti indici di stima: come prima approssimazione si è proceduto al calcolo dell'indicatore **UUCP** (Unadjusted Use Case Point):

$$\text{UUCP} = \text{UUCW} + \text{UAW} = 120 + 14 = 134$$

Successivamente sono stati calcolati il fattore di complessità tecnica **TFC** e il fattore ambientale **EF**:

$$\text{TFC} = 0.6 + (0.01 * \text{TFactor}) = 0.6 * (0.01 * 28) = 0.88$$

$$\text{EF} = 1.4 + (-0.03 * \text{EFactor}) = 1.4 + (-0.03 * 32) = 0.44$$

Moltiplicando gli indicatori calcolati si ricava l'indicatore **UCP** (Use Case Point):

$$\text{UCP} = \text{UUCP} * \text{TFC} * \text{EF} = 51.88$$

Supponendo un rapporto di 25 ore per Use Case Point si ottiene un quantitativo di ore pari a:

$$\text{Ore} = 25 * 51.88 \simeq 1297 \text{ H}$$

<i>Complessità</i>	<i>Peso</i>	<i>Numeri casi d'uso</i>	<i>Prodotto</i>
<i>Semplice</i>	5	0	0
<i>Medio</i>	10	6	60
<i>Complesso</i>	15	4	60
<i>Totale</i>		10	120

<i>Tipologia Attore</i>	<i>Peso</i>	<i>Numero Attori</i>	<i>Prodotto</i>
<i>Semplice</i>	1	0	0
<i>Medio</i>	2	1	2
<i>Complesso</i>	3	4	12
<i>Totale</i>		5	14

Fattore	Descrizione	Peso	Valutazione	Prodotto
T1	Sistema distribuito	2.0	1	2
T2	Performance	1.0	4	4
T3	Efficienza per l'utente finale	1.0	4	4
T4	Complessità di elaborazione interna	1.0	2	2
T5	Riusabilità del codice	1.0	1	1
T6	Facile da installare	0.5	1	0.5
T7	Facile da usare	0.5	5	2.5
T8	Portabile	2.0	0	0
T9	Facile da cambiare	1.0	1	1
T10	Elaborazione parallela	1.0	1	1
T11	Caratteristiche di sicurezza	1.0	5	5
T12	Accesso diretto da terzi	1.0	1	1
T13	Formazione per utenti finali	1.0	4	4
Totale				28

Fattore	Descrizione	Peso	Valutazione	Prodotto
E1	Familiarità con il processo di sviluppo	1.5	5	7.5
E2	Esperienza di applicazione	0.5	3	1.5
E3	Esperienza object-oriented del team	1.0	5	5
E4	Capacità di analisi	0.5	4	2
E5	Motivazione del team	1.0	5	5
E6	Stabilità dei requisiti	2.0	3	6
E7	Personale part-time	-1.0	1	-1
E8	Complessità del linguaggio utilizzato	2.0	3	6
Totale				32

2.5 Workshop

Nel processo di sviluppo sono stati effettuati cinque workshop, per ognuno di essi sono stati raffinati i risultati ottenuti dalle iterazioni scaturite dai task individuati nel workshop precedente.

2.5.1 Primo workshop

Nel primo workshop è stata effettuata una raccolta di requisiti funzionali di alto livello e sono stati identificati i principali requisiti non funzionali che il sistema deve rispettare. Successivamente sono stati individuati tutti i casi d'uso necessari alla copertura dei requisiti individuati e si è passati alla modellazione tramite il linguaggio **UML**. In particolare, sono stati realizzati:

- Use case diagram;
- Class diagram;
- Component diagram
- Sequence diagram di analisi;

- **Sequence diagram di dettaglio;**
- **Activity diagram.**

2.5.2 Secondo workshop

Nel secondo workshop è stato realizzato il modello ER, la cui traduzione ha portato alla realizzazione della base dati; e si è passati all'implementazione delle classi in linguaggio Java secondo il class diagram.

2.5.3 Terzo Workshop

Nel terzo workshop è stata modificata ad hoc la base dati, è stato ultimato il codice Java terminando così lo sviluppo della parte Back End e si è passati alla realizzazione del Front End(GUI).

2.5.4 Quarto Workshop

Nel quarto workshop è stata ultimata l'interfaccia e la web application.

2.5.5 Quinto Workshop

Nel quinto ed ultimo workshop si è proceduto al testing del sistema, al completamento della documentazione e alla consegna del software.

2.6 Strumenti utilizzati

2.6.1 Visual Paradigm



Visual Paradigm è uno strumento di supporto di UML 2, SysML e Business Process Modeling Notation (BPMN) dal Object Management Group (OMG) . Oltre al supporto per la modellazione, fornisce funzionalità di generazione di report e di ingegnerizzazione del codice, inclusa la generazione di codice . Può decodificare i diagrammi dal codice e fornire ingegneria di andata e ritorno per vari linguaggi di programmazione .

2.6.2 IntelliJ



IntelliJ IDEA è un ambiente di sviluppo integrato (IDE) per il linguaggio di programmazione Java. Sviluppato da JetBrains (prima conosciuto come IntelliJ), è disponibile sia in licenza Apache che in edizione proprietaria commerciale.

2.6.3 Trello



Trello è un software gestionale in stile Kanban basato sul web. Gli utenti possono creare le loro schede attività con più colonne, assegnarsi e scambiarsi le attività tra di loro. In genere le colonne sono organizzate in stati dell'attività: To Do, Doing e Done. Il software è utilizzabile per uso personale e aziendale.

2.6.4 Microsoft Teams



Microsoft Teams è una piattaforma di comunicazione e collaborazione unificata che combina chat di lavoro persistente, teleconferenza, condivisione di contenuti (incluso lo scambio e il lavoro simultaneo sui file) e integrazione delle applicazioni.

2.6.5 Overleaf



Overleaf è un editor LaTeX collaborativo basato su cloud utilizzato per scrivere, modificare e pubblicare documenti scientifici.

2.6.6 PostgreSQL



PostgreSQL è un completo DBMS ad oggetti rilasciato con licenza libera.

2.6.7 pgAdmin



pgAdmin è un'applicazione C++ libera, una interfaccia grafica che consente di amministrare in modo semplificato database di PostgreSQL. L'applicazione è indirizzata sia agli amministratori del database, sia agli utenti. Gestisce i permessi prelevandoli dal database PostgreSQL. pgAdmin permette di creare un database da zero, creare le tabelle ed eseguire operazioni di ottimizzazione sulle stesse. Presenta un feedback sulla creazione delle tabelle per evitare eventuali errori. Sono previste delle funzionalità per l'inserimento dei dati (popolazione del database), per le query, per il backup dei dati, ecc. L'amministratore, invece ha a disposizione un'interfaccia grafica per la gestione degli utenti: l'interfaccia permette l'inserimento di un nuovo utente, la modifica della relativa password e la gestione dei permessi che l'utente ha sul database, utilizzando lo standard SQL. È multiplatforma in quanto realizzato in linguaggio Python. È disponibile in una dozzina di lingue.

2.6.8 JUnit



Per quanto riguarda il testing di unità e quello di integrazione abbiamo usato il framework JUnit. Si tratta di un framework assai rilevante, che ha promosso il ruolo fondamentale dei test nelle attività di programmazione, secondo un modello di programmazione Test Driven. Enfatizza dunque l'uso ripetuto dei test, per assicurare stabilità nel codice e ridurre le attività di debugging a posteriore. JUnit fornisce asserzioni per testare i risultati attesi, annotazioni per identificare i metodi di prova e test runner grafici e testuali. Offre una suite per organizzare ed eseguire facilmente i test, e permette la condivisione dei risultati.



3 Glossario dei termini

Per rendere chiari i concetti successivi si è deciso di redigere un glossario per specificare i termini utilizzati e per definire le entità in gioco. Di seguito la tabella con i termini individuati.

Termine	Definizione e informazioni	Formato	Regole di validazione
Acquirente	L'utente si registra sulla piattaforma per effettuare degli ordini		L'utente è identificato da una e-mail e una password
Azienda	L'utente si registra sulla piattaforma per vendere prodotti, gestire la fornitura e i servizi di consulenza		L'utente è identificato da una e-mail e una password
Prodotto	Oggetto proposto in vendita sulla piattaforma, caratterizzato da un codice, da un nome, dalla categoria e dal prezzo		È identificato da un codice
Ordine	Elenco di prodotti che l'acquirente vuole acquistare, caratterizzato da un codice.		È identificato da un codice
Agente di vendita	Offre servizi di consulenza sull'acquisto di prodotti su richiesta di un acquirente		È identificato dalla partita IVA
Fornitore	Si occupa della fornitura dei prodotti		È identificato da un codice

Dati d'acquisto	Elenco di dati caratterizzanti l'acquisto	Numerico e testuale	
Catalogo	Elenco dei prodotti presenti sul sito	Testuale	
Gestione ordine	Gestione dell'ordine associato ad un acquirente		La gestione dell'ordine è lasciata alla logica di business del sistema
Consulenza	Attività svolta dall'agente che consiglia il prodotto più adatto alle esigenze dell'acquirente		La consulenza è offerta da un agente solo se richiesta
Partita IVA	La partita IVA è una sequenza di 11 cifre che identifica univocamente un soggetto che esercita un'attività, di impresa e <u>non</u> , rilevante ai fini dell'imposizione fiscale indiretta.	Numerico	L'azienda e l'agente devono necessariamente essere dotati di partita IVA
Tipologia di fornitura	Ogni prodotto appartiene ad una categoria, in base alla categoria si adotta una fornitura diversa	Testuale	
Data emissione dell'ordine	La data di emissione indica quando l'ordine è stato effettuato	Testuale	

4 Specifica dei requisiti

4.1 Descrizione testuale dei requisiti

Si vuole realizzare un sistema software per la gestione degli acquisti, da parte di utenti della piattaforma, di prodotti forniti da aziende specializzate. Un'azienda, che si identifica con e-mail e password, può inserire sulla piattaforma uno o più prodotti che vanno ad aggiungersi al catalogo. Un'acquirente, anch'esso identificato tramite e-mail e password, può invece navigare nel sito in modo da poter trovare il prodotto di suo interesse e può comprare uno o più di quegli oggetti a seconda della disponibilità della piattaforma. Questi acquisti possono avvenire per tramite di un agente di commercio (che può essere a seconda del ruolo un agente semplice oppure un capo, relativo ad una determinata area geografica) che indirizza i clienti verso determinati prodotti e al quale, nel momento dell'acquisto, vengono riconosciuti i meriti della persuasione del cliente, tramite l'inserimento da parte dello stesso nelle informative dell'acquisto. I prodotti che vengono acquistati dal cliente sono messi a disposizione da diversi fornitori, assoldati dalle varie aziende.

4.2 Descrizione generale

4.2.1 Prospettive del prodotto

Lo scopo di questa piattaforma è quello di poter gestire tutto quello che concerne la vendita al dettaglio avvenuta per tramite di un agente di commercio. Permettendo in questo modo una semplice eventuale gestione degli sconti da applicare al cliente oppure semplificando una politica che potrebbe essere quella di premiare gli agenti più attivi a livello locale.

4.2.2 Opportunità di business

La piattaforma si propone come scopo quello di essere l'ultimo anello della catena di distribuzione dei prodotti, la parte che si connette col consumatore finale. In futuro potranno essere rilasciati degli incrementi che riguardano la gestione della distribuzione dei prodotti a partire dalla fornitura fino al rilascio al cliente.

4.2.3 Demografia di mercato

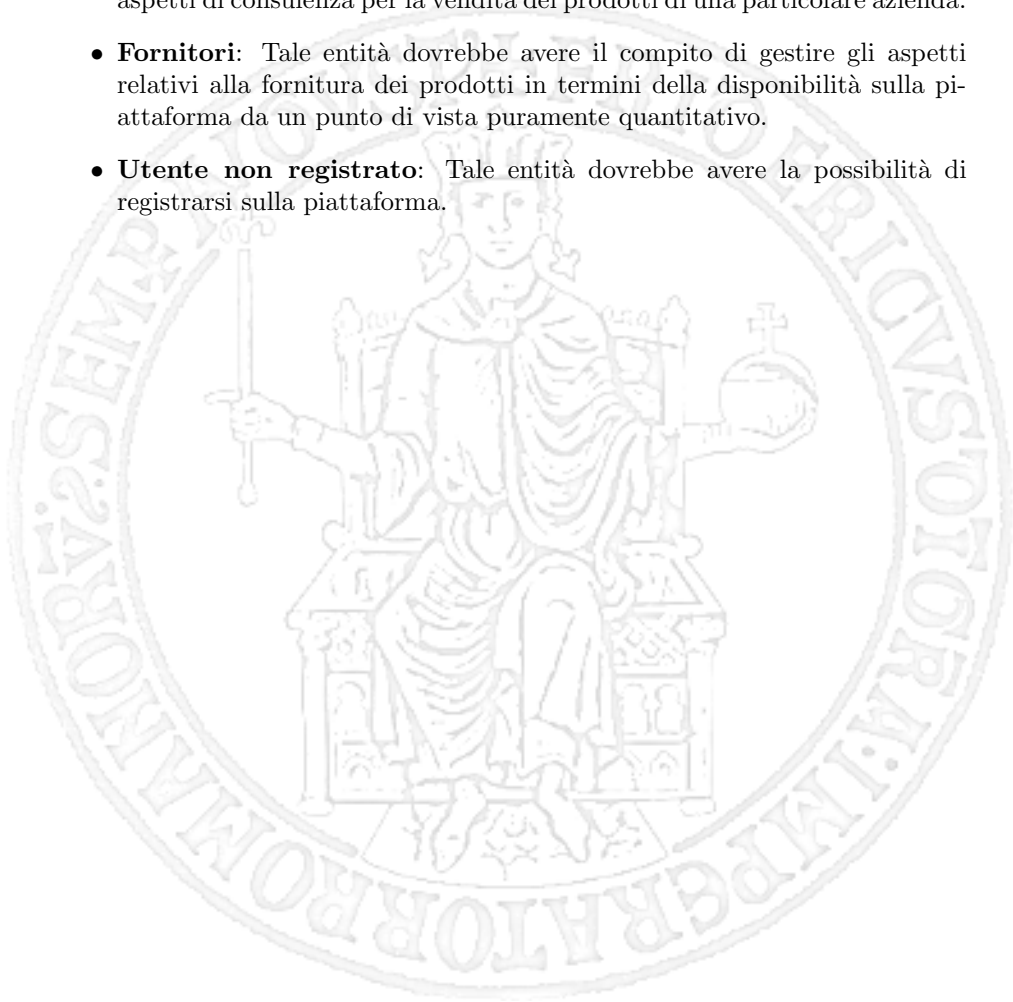
Il prodotto è destinato a tutte le aziende, le quali hanno la necessità di gestire una vasta gamma di prodotti, di agenti di mercato e fornitori, dandole la possibilità, con un solo strumento, di avere una gestione ottimale delle sue attività.

4.3 Descrizione delle parti di interesse

- **Acquirente:** Persona che può visualizzare il catalogo e in particolare applicare filtri alla ricerca per visualizzare un prodotto specifico, o i prodotti

più acquistati o quelli recentemente inseriti e, eventualmente previo consulenza da un'agente di vendita, effettuare un ordine che comprende l'acquisto di uno o più prodotti.

- **Azienda:** Utente interessato alla vendita dei prodotti sulla piattaforma. Anch'esso può visualizzare l'intero catalogo dei prodotti, applicare filtri quali la visualizzazione dei soli prodotti più acquistati, dei prodotti inseriti più di recente, oltre alla visualizzazione dei prodotti propri, delle proprie vendite effettuate e alla possibilità di inserire uno o più prodotti sulla piattaforma destinati alla vendita.
- **Agente di vendita:** Tale entità dovrebbe avere la possibilità di gestire aspetti di consulenza per la vendita dei prodotti di una particolare azienda.
- **Fornitori:** Tale entità dovrebbe avere il compito di gestire gli aspetti relativi alla fornitura dei prodotti in termini della disponibilità sulla piattaforma da un punto di vista puramente quantitativo.
- **Utente non registrato:** Tale entità dovrebbe avere la possibilità di registrarsi sulla piattaforma.



4.4 Obiettivi di alto livello e problemi delle parti di interesse

<i>Obiettivo di alto livello</i>	<i>Attore</i>	<i>Priorità</i>	<i>Problemi e preoccupazioni</i>	<i>Soluzione corrente</i>
<i>Inserimento dei prodotti</i>	<i>Azienda</i>	<i>Alta</i>	<i>Bisogna permettere l'inserimento di nuovi prodotti con i relativi prezzi e quantità.</i>	Si prevede un'interfaccia specifica alla quale l'azienda può accedere a seguito del login per effettuare la registrazione e la modifica di un prodotto.
<i>Acquisto dei prodotti</i>	<i>Acquirente</i>	<i>Alta</i>	<i>Bisogna permettere all'utente la selezione del prodotto da acquistare e le relative quantità per esso. Vi deve essere anche un campo per l'inserimento dell'agente di commercio con cui la persona è stata a contatto.</i>	Si prevede un'interfaccia specifica alla quale l'utente può accedere a seguito del login per effettuare l'acquisto di uno o più prodotti e l'inserimento dell'agente di commercio ed infine poter pagare.

4.5 Requisiti funzionali

Un utente, previa necessaria registrazione e autenticazione tramite apposita interfaccia di Login, può eseguire diverse operazioni, alcune indipendenti dal ruolo e altre dipendenti, a seconda se l'utente è un acquirente o un'azienda. L'utente genetico può:

- Registrarsi alla piattaforma;
- Effettuare l'accesso al sistema.

L'azienda può:

- Visualizzare il catalogo dei prodotti;
- Visualizzare l'elenco dei prodotti da lei messi in vendita;
- Visualizzare i prodotti più venduti;
- Visualizzare i prodotti inseriti più di recente;
- Visualizzare lo storico delle vendite;
- Inserire uno o più prodotti.

L'acquirente può:

- Visualizzare il catalogo dei prodotti;
- Visualizzare i prodotti più venduti;
- Visualizzare i prodotti inseriti più di recente;
- Effettuare un ordine.

4.6 Requisiti non funzionali

4.6.1 Sicurezza

Per evitare che chiunque possa effettuare operazioni attraverso il sistema, ogni utente che vuole interagire con esso deve essere autenticato.

4.6.2 Usabilità

Il cliente deve essere in grado di visualizzare le schermate dell'applicativo in maniera chiara. La velocità, la facilità e l'elaborazione senza errori sono fondamentali nell'inserimento dei prodotti e nell'aggiornamento della piattaforma, in quanto si desidera non incappare in alcun errore di elaborazione di esse, dal momento che si potrebbero avere degli errori di inconsistenza nell'acquisto e di conseguenza ci si troverà a dover annullare alcuni ordini con la conseguente disapprovazione e scontento del cliente. Le informazioni devono essere trasmesse con una grafica user-friendly, in quanto deve risultare il più semplice possibile inserire le informazioni per evitare errori.

4.6.3 Affidabilità

Si garantisce la persistenza dei dati tramite sistemi esterni, in modo da garantire un corretto recupero delle informazioni anche in caso di fallimento del software.

4.6.4 Performance

Il nostro obiettivo è quello di permettere l'inserimento di nuove istanze in meno di 1 minuto.

4.6.5 Supportabilità

- **Manutenibilità:** Il software presentato sarà facilmente modificabile, includendo correzioni, migliorie e adattamenti. Sarà facilmente analizzabile per la localizzazione di eventuali errori e stabile in modo da evitare effetti inaspettati dovuti a modifiche errate.
- **Configurabilità:** I differenti utilizzatori dell'applicazione hanno esigenze differenti ed eseguono diverse operazioni. Pertanto, il cliente si aspetta che ad ogni operatore corrisponda una configurazione di interfacce differenti.

4.7 Regole di business

ID	Regola	Modificabilità	Sorgente
REGOLA1	Solo le aziende possono inserire prodotti	Non è modificabile	Politica della piattaforma
REGOLA2	Solo gli acquirenti possono ordinare prodotti	Non è modificabile	Politica della piattaforma
REGOLA3	I clienti devono obbligatoriamente inserire nell'ordine la quantità di ogni singolo prodotto	Non è modificabile	Politica della piattaforma
REGOLA4	Un'azienda può inserire solo prodotti propri	Non è modificabile	Politica della piattaforma

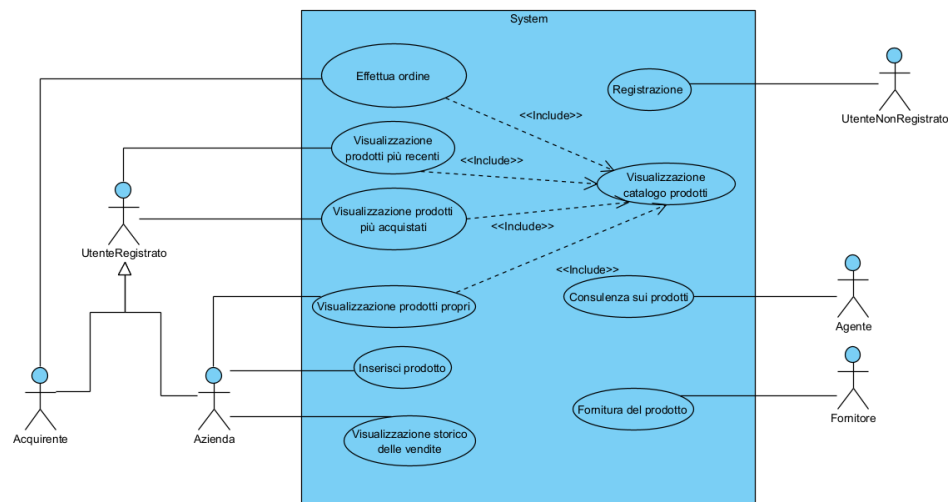
5 Analisi dei requisiti

5.1 Identificazione degli attori

Durante lo sviluppo del sistema sono stati individuati i seguenti attori primari:

- Azienda
- Acquirente
- Agente di vendita
- Fornitore
- Utente non registrato

5.2 Use case diagrams



5.3 Descrizione breve dei casi d'uso

- **Caso d'uso 1: Inserisci prodotto**
Il caso d'uso permette all'azienda di inserire un prodotto all'interno del database.
- **Caso d'uso 2: Visualizzazione catalogo prodotti**
Il caso d'uso permette ad aziende e acquirenti di visualizzare tutti i prodotti presenti nel catalogo.
- **Caso d'uso 3: Visualizzazione prodotti più acquistati**
Il caso d'uso permette ad aziende ed acquirenti di visualizzare i prodotti più acquistati sulla piattaforma.

- **Caso d'uso 4: Visualizzazione prodotti più recenti**
Il caso d'uso permette ad aziende ed acquirenti di visualizzare i prodotti inseriti più di recente sulla piattaforma.
- **Caso d'uso 5: Visualizzazione prodotti propri**
Il caso d'uso permette a ciascun azienda di visualizzare l'elenco completo dei prodotti che essa ha messo in vendita.
- **Caso d'uso 6: Visualizzazione storico delle vendite**
Il caso d'uso permette a ciascun azienda di visualizzare quanti, quali e in che quantità i prodotti da essa venduti.
- **Caso d'uso 7: Effettua ordine**
Il caso d'uso permette all'acquirente di inserire all'interno del carrello uno o più prodotti che intende acquistare, indicandone la quantità d'acquisto, e confermare l'ordine.
- **Caso d'uso 8: Consulenza sui prodotti**
Il caso d'uso permette all'agente di offrire un servizio di consulenza all'acquirente sull'acquisto di uno o più prodotti.
- **Caso d'uso 9: Fornitura del prodotto**
Il caso d'uso permette al fornitore di effettuare operazioni di fornitura dei prodotti.
- **Caso d'uso 10: Registrazione dell'utente**
Il caso d'uso permette all'utente di registrarsi sulla piattaforma

5.4 Tabella Attori - Obiettivi

Attore	Obiettivi
Persona	<ul style="list-style-type: none"> • Registrarsi alla piattaforma • Effettua il login all'interno del sistema • Effettuare un ordine
Azienda	<ul style="list-style-type: none"> • Registrarsi alla piattaforma • Effettua il login all'interno del sistema • Inserire un prodotto

5.5 Descrizione di dettaglio dei casi d'uso

5.5.1 Caso d'uso 1: Inserisci prodotto

Nome del caso d'uso	Inserisci prodotto
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda
Attore primario	Azienda
Parti interessate	Azienda che vuole inserire un prodotto
Precondizioni	L'azienda deve essere registrata e deve aver effettuato il login alla piattaforma
Garanzia di successo	Il prodotto viene inserito nel Database
Principale scenario di successo	<ol style="list-style-type: none">1. L'azienda si registra alla piattaforma2. L'azienda effettua il login alla piattaforma3. L'azienda inserisce i dati del prodotto4. Il sistema verifica la validità dei dati5. Il sistema aggiunge il prodotto al Database
Estensioni	Se i dati inseriti non sono validi 1a. Il sistema risponde con un messaggio d'errore
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda è identificata tramite e-mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.2 Caso d'uso 2: Visualizzazione catalogo prodotti

Nome del caso d'uso	Visualizzazione catalogo prodotti
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda – Obiettivo Acquirente
Attore primario	Azienda - Acquirente
Parti interessate	L'Azienda o l'acquirente vuole conoscere l'intero catalogo dei prodotti sulla piattaforma
Precondizioni	L'azienda e l'acquirente devono essere registrati e devono aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema mostra il catalogo corretto
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'azienda (acquirente) si registra alla piattaforma 2. L'azienda (acquirente) effettua il login alla piattaforma 3. L'azienda (acquirente) richiede la visualizzazione del catalogo completo della piattaforma 4. Il sistema effettua la ricerca di tutti i prodotti presenti sulla piattaforma 5. L'azienda (acquirente) visualizza il catalogo
Estensioni	<p>Se non vi sono prodotti nel catalogo</p> <p>1a. Il sistema risponde con un messaggio d'errore</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda e l'acquirente sono identificati tramite e-mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.3 Caso d'uso 3: Visualizzazione prodotti più acquistati

Nome del caso d'uso	Visualizzazione prodotti più acquistati
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda – Obiettivo acquirente
Attore primario	Azienda - Acquirente
Parti interessate	L'Azienda o l'acquirente vuole sapere quali sono i prodotti più in voga sulla piattaforma
Precondizioni	L'azienda e l'acquirente devono essere registrati e devono aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema mostra l'elenco dei prodotti più acquistati
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'azienda (acquirente) si registra alla piattaforma 2. L'azienda (acquirente) effettua il login alla piattaforma 3. L'azienda (acquirente) richiede la visualizzazione dei prodotti più acquistati 4. Il sistema effettua la ricerca dei prodotti più acquistati 5. L'azienda (acquirente) visualizza l'elenco dei prodotti più acquistati
Estensioni	<p>Se nessun prodotto è stato acquistato</p> <p>1a. Il sistema risponde con il messaggio "Nessun prodotto acquistato"</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda e l'acquirente sono identificati tramite e-mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.4 Caso d'uso 4: Visualizzazione prodotti più recenti

Nome del caso d'uso	Visualizzazione prodotti più recenti
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda – Obiettivo acquirente
Attore primario	Azienda - Acquirente
Parti interessate	L' Azienda o l'acquirente vuole sapere quali sono i prodotti più recenti inseriti nella piattaforma
Precondizioni	L'azienda e l'acquirente devono essere registrati e devono aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema mostra il corretto elenco dei prodotti più recenti
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'azienda (acquirente) si registra alla piattaforma 2. L'azienda (acquirente) effettua il login alla piattaforma 3. L'azienda (acquirente) richiede la visualizzazione dei prodotti più recenti inseriti sulla piattaforma 4. Il sistema compie la query di ricerca dei prodotti più recenti 5. Si visualizza l'elenco dei prodotti più recenti
Estensioni	<p>Se non sono stati aggiunti di recente dei prodotti</p> <p>1a. Il sistema risponde con un messaggio di errore</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda e l'acquirente sono identificati tramite mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.5 Caso d'uso 5: Visualizzazione prodotti propri

Nome del caso d'uso	Visualizzazione prodotti propri
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda
Attore primario	Azienda
Parti interessate	Azienda che vuole visualizzare i propri prodotti attualmente in vendita sulla piattaforma
Precondizioni	L'azienda deve essere registrata e deve aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema mostra tutti i prodotti dell'azienda
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'azienda si registra alla piattaforma 2. L'azienda effettua il login alla piattaforma 3. L'azienda richiede la visualizzazione dei propri prodotti 4. Il sistema compie la ricerca dei prodotti da essa inseriti 5. L'azienda visualizza il proprio catalogo
Estensioni	<p>Se non vi sono prodotti venduti dall'azienda</p> <p>1a. Il sistema risponde con il messaggio "Nessun prodotto trovato"</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda è identificata tramite mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.6 Caso d'uso 6: Visualizzazione storico delle vendite

Nome del caso d'uso	Visualizza storico delle vendite
Ambito	Piattaforma Capemont
Livello	Obiettivo azienda
Attore primario	Azienda
Parti interessate	Azienda che vuole visualizzare lo storico delle sue vendite
Precondizioni	L'azienda deve essere registrata e deve aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema mostra lo storico corretto
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'azienda si registra alla piattaforma 2. L'azienda effettua il login alla piattaforma 3. L'azienda richiede la visualizzazione dello storico 4. Il sistema compie la query di ricerca dei prodotti dello storico 5. Si visualizza lo storico
Estensioni	<p>Se non vi sono prodotti nello storico</p> <p>1a. Il sistema risponde con il messaggio "Storico vuoto"</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda è identificata tramite mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.7 Caso d'uso 7: Effettua ordine

Nome del caso d'uso	Effettua Ordine
Ambito	Piattaforma Capemont
Livello	Obiettivo acquirente
Attore primario	Acquirente
Parti interessate	L' Acquirente vuole effettuare un ordine di acquisto
Precondizioni	L' acquirente deve essere registrato e deve aver effettuato il login alla piattaforma
Garanzia di successo	Il sistema conferma l'ordine effettuato
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'acquirente si registra alla piattaforma 2. L'acquirente effettua il login alla piattaforma 3. L'acquirente sceglie cosa comprare e in che quantità 4. Il sistema procede con la verifica dell'effettiva disponibilità dei prodotti 5. L'ordine viene effettuato
Estensioni	<p>Se il prodotto non è più disponibile</p> <p>1a. Il sistema risponde con un messaggio d'errore</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'acquirente è identificato tramite e-mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.8 Caso d'uso 8: Consulenza sui prodotti

Nome del caso d'uso	Consulenza sui prodotti
Ambito	Piattaforma Capemont
Livello	Obiettivo agente
Attore primario	Agente
Parti interessate	L' Agente vuole effettuare una consulenza su un prodotto
Precondizioni	L' Agente deve essere registrato presso l'azienda che vende quel prodotto
Garanzia di successo	Il sistema fornisce il codice promozionale per la scontistica sul prodotto a seguito della consulenza
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'agente si registra alla piattaforma 2. L'agente effettua il login alla piattaforma 3. L'agente comunica al cliente i prodotti che fanno al suo caso 4. Il sistema fornisce il codice promozionale
Estensioni	<p>Se la consulenza non va a buon fine</p> <p>1a. Il sistema non fornisce alcun codice promozionale</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.9 Caso d'uso 9: Fornitura del prodotto

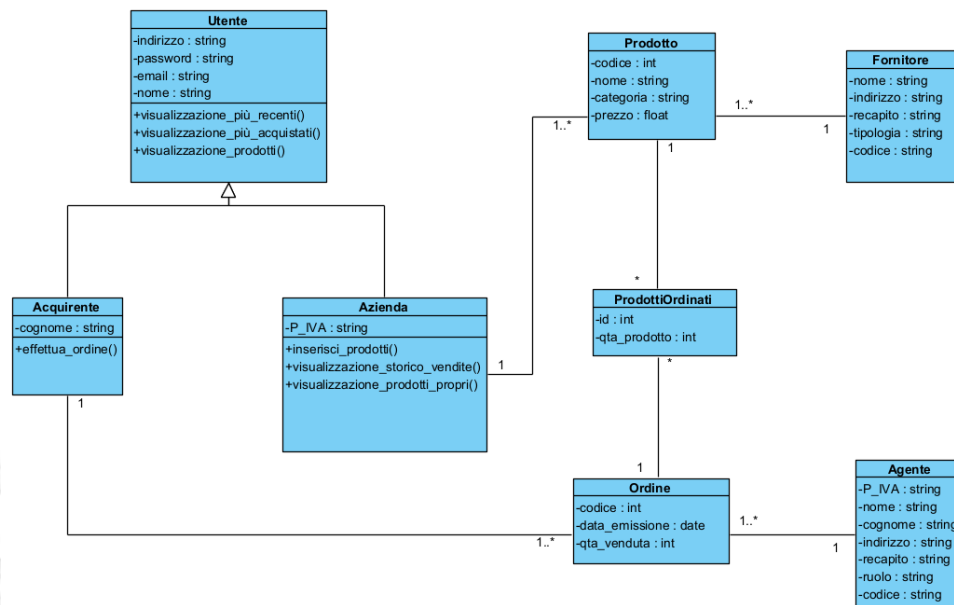
Nome del caso d'uso	Fornitura del prodotto
Ambito	Piattaforma Capemont
Livello	Obiettivo fornitore
Attore primario	Fornitore
Parti interessate	Il fornitore aggiorna la quantità di prodotto disponibile
Precondizioni	Il Fornitore deve essere registrato sulla piattaforma
Garanzia di successo	Il sistema aggiorna correttamente la quantità di prodotto disponibile
Principale scenario di successo	<ol style="list-style-type: none">1. Il fornitore si registra alla piattaforma2. Il fornitore effettua il login alla piattaforma3. Il fornitore indica i nuovi valori da inserire4. Il sistema aggiorna i dati5. Il sistema fornisce un messaggio di conferma
Estensioni	Se l'operazione non va a buon fine 1a. Il sistema fornisce un messaggio di errore
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.5.10 Caso d'uso 10: Registrazione Utente

Nome del caso d'uso	Registrazione utente
Ambito	Piattaforma Capemont
Livello	Obiettivo Acquirente/Azienda
Attore primario	Utente
Parti interessate	Utente che vuole registrarsi alla piattaforma
Precondizioni	L'utente non deve essere registrato
Garanzia di successo	L'utente può effettuare il login
Principale scenario di successo	<ol style="list-style-type: none"> 1. L'utente inserisce i dati per la registrazione 2. Il sistema verifica che i dati siano validi 3. Il sistema aggiunge i dati al Database 4. Il sistema invia un messaggio di conferma
Estensioni	<p>Se i dati inseriti non sono validi</p> <p>1a. Il sistema risponde con un messaggio d'errore</p>
Requisiti particolari	Sistema di comunicazione tra terminale e base dati, che risponda in maniera affidabile il 90% delle volte
Tecnologie e variazioni dei dati	L'azienda si registra tramite mail e password
Frequenza di occorrenza	Potenzialmente continuo
Varie	

5.6 System domain model

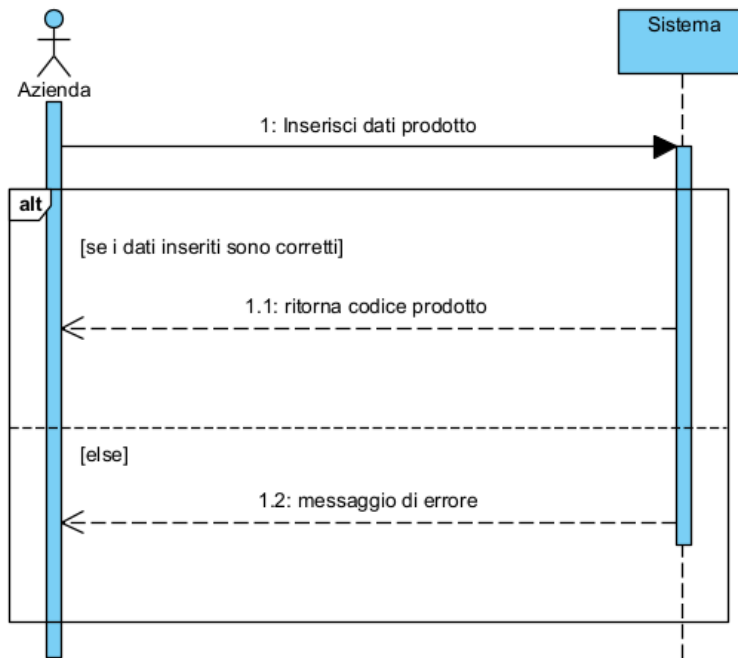
Al fine di mettere a fuoco i concetti fondamentali del sistema e definire un vocabolario specifico, si è pensato di introdurre il system domain model. Un importante merito di questo modello è quello di fornire una base comune di concetti su cui ragionare e una terminologia condivisa rigorosa e specifica. Tale modello dà quindi una descrizione statica della struttura del sistema.



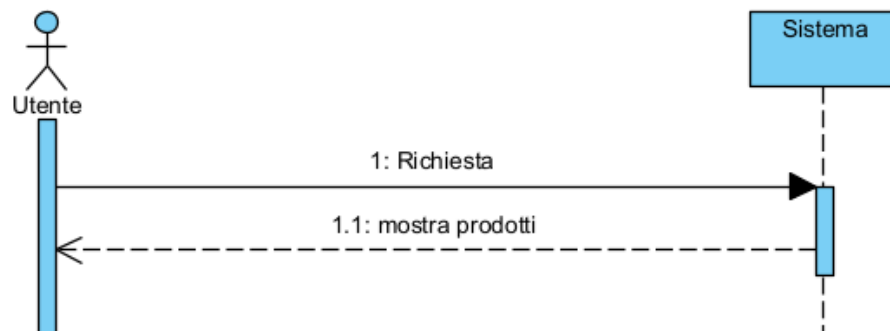
5.7 Sequence diagrams di analisi

Un diagramma utile per visualizzare le interazioni tra un attore e il sistema è il diagramma di sequenza, che mette in luce una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate.

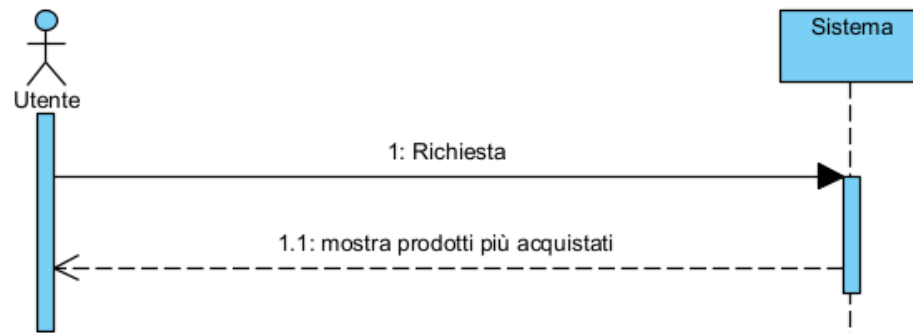
5.7.1 Sequence diagram: Inserisci prodotto



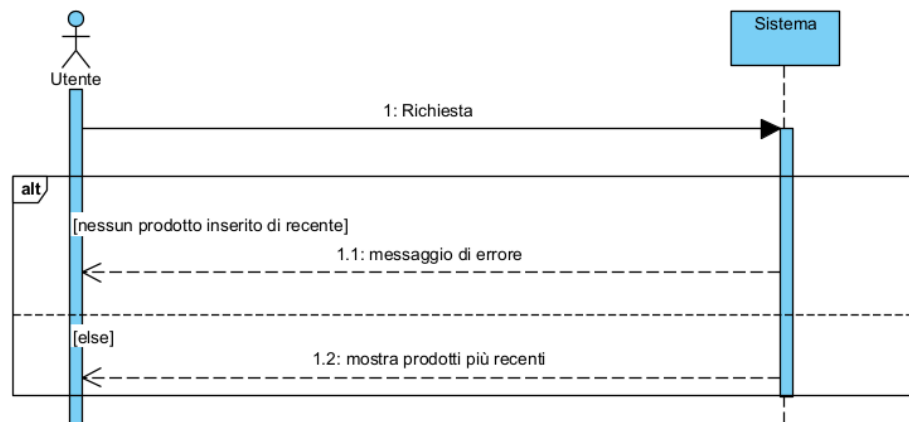
5.7.2 Sequence diagram: Visualizzazione catalogo prodotti



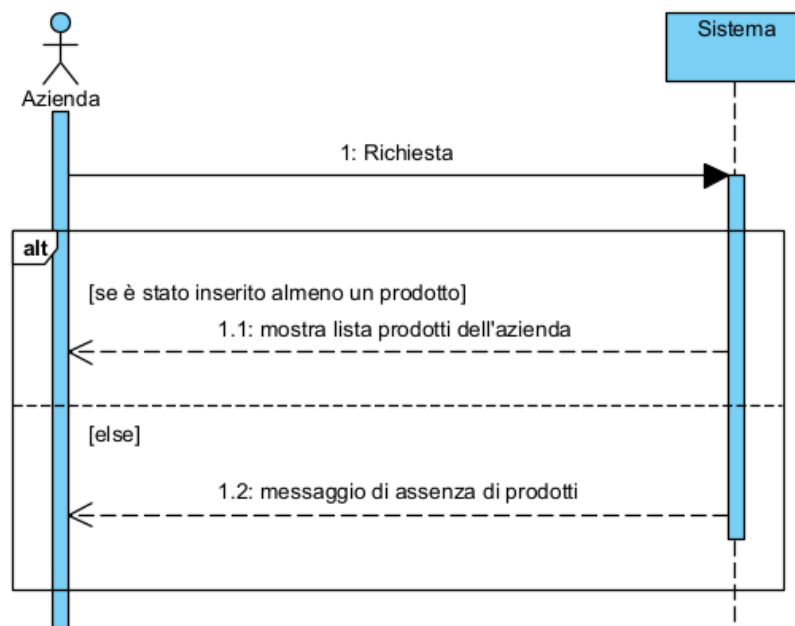
5.7.3 Sequence diagram: Visualizzazione prodotti più acquistati



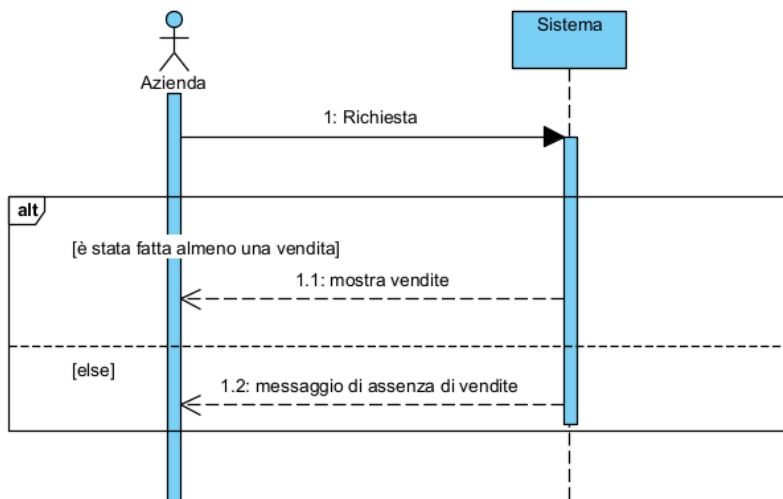
5.7.4 Sequence diagram: Visualizzazione prodotti più recenti



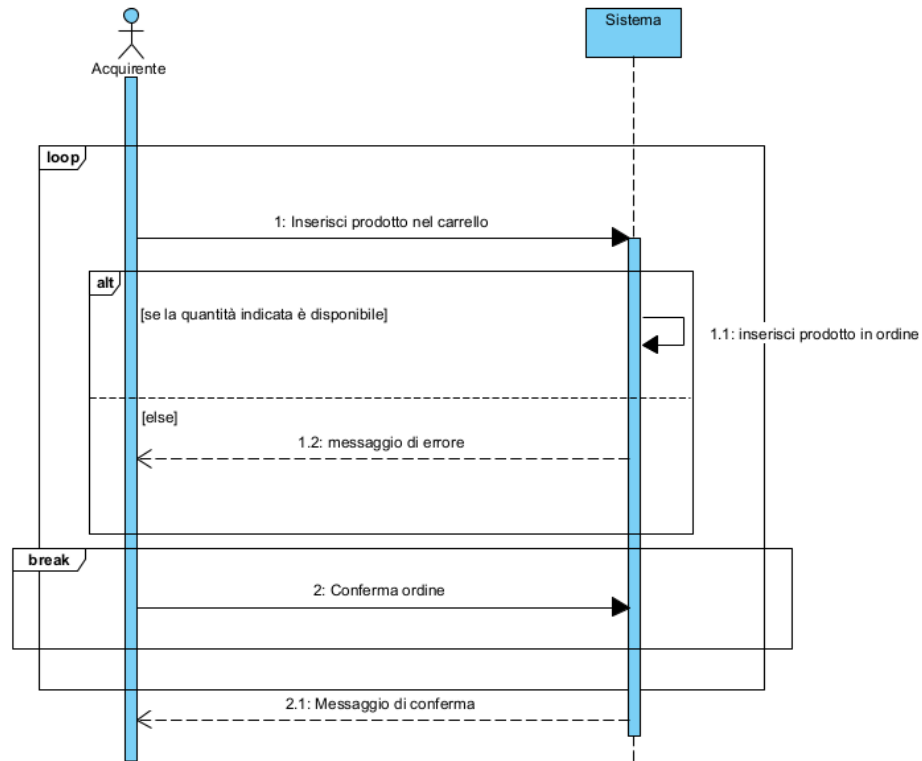
5.7.5 Sequence diagram: Visualizzazione prodotti propri



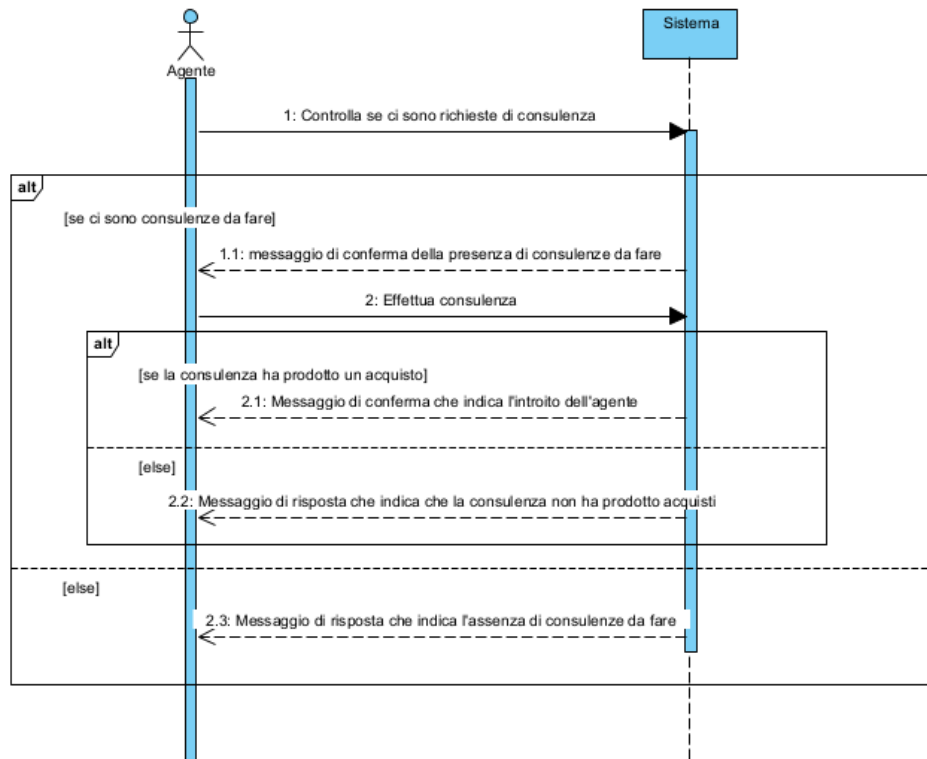
5.7.6 Sequence diagram: Visualizzazione storico delle vendite



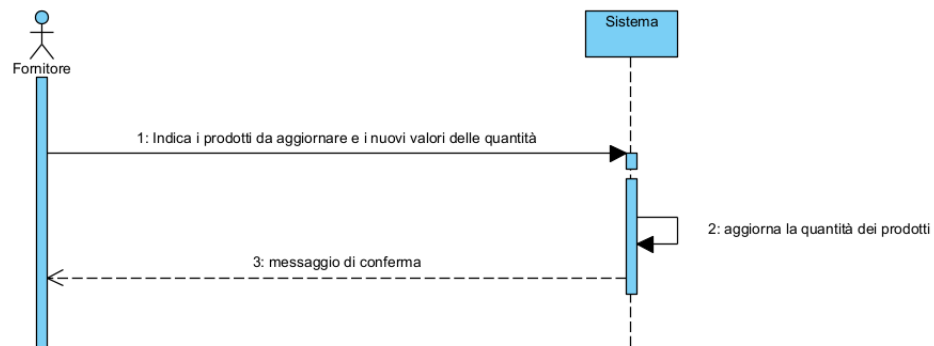
5.7.7 Sequence diagram: Effettua ordine



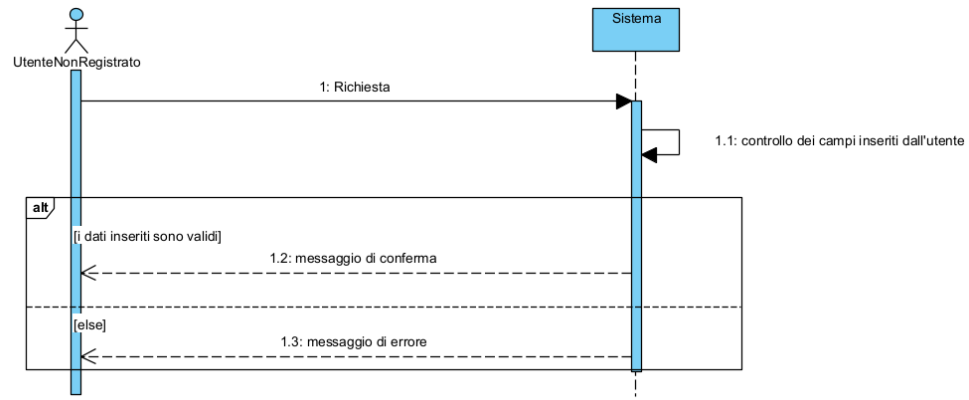
5.7.8 Sequence diagram: Consulenza sui prodotti



5.7.9 Sequence diagram: Fornitura del prodotto



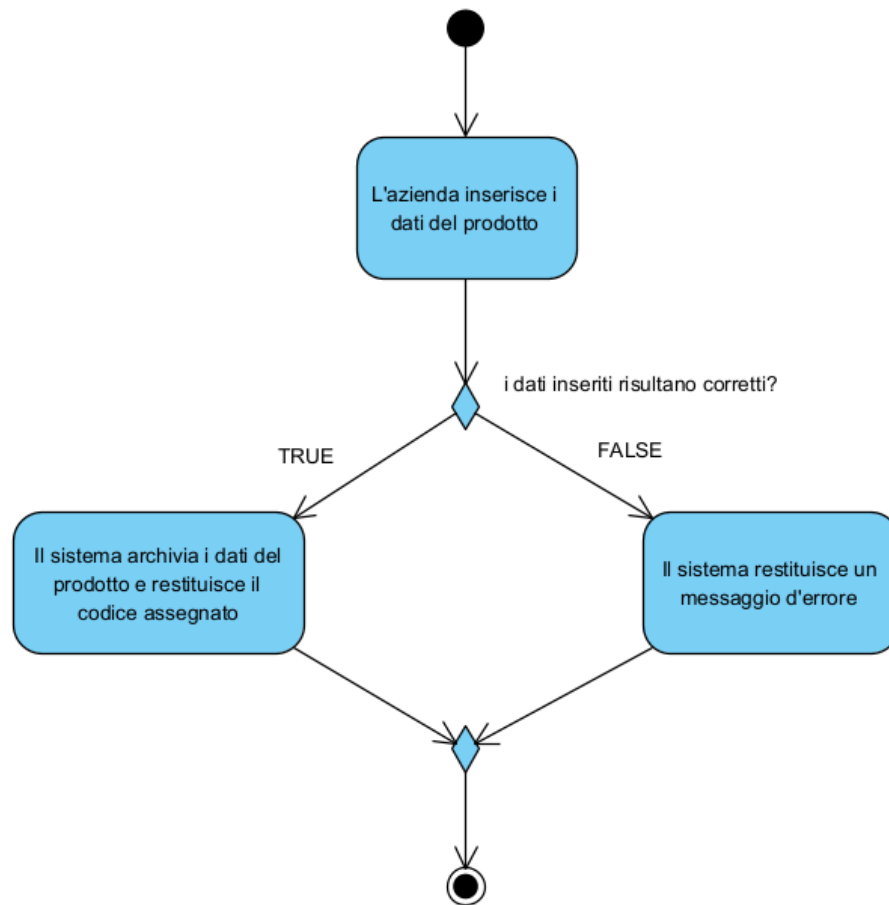
5.7.10 Caso d'uso 10: Registrazione Utente



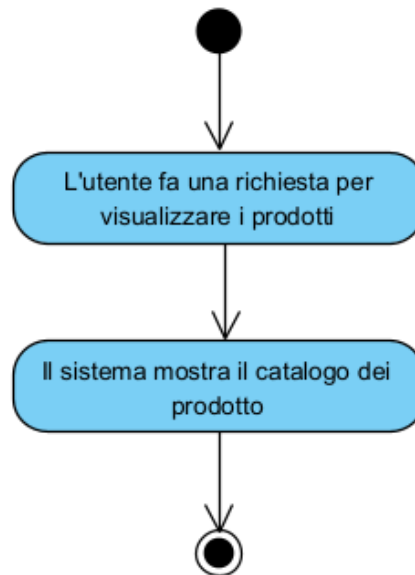
5.8 Activity diagrams

Gli activity diagrams consentono di descrivere un processo attraverso grafi orientati in cui i nodi rappresentano le attività, e gli archi rappresentano l'ordine con cui vengono eseguite le attività.

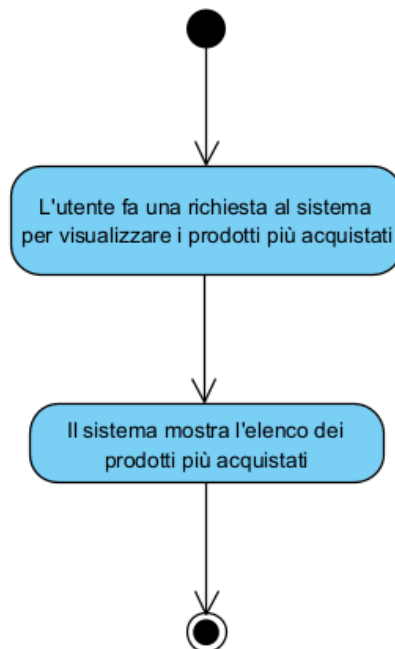
5.8.1 Activity diagram: Inserisci prodotto



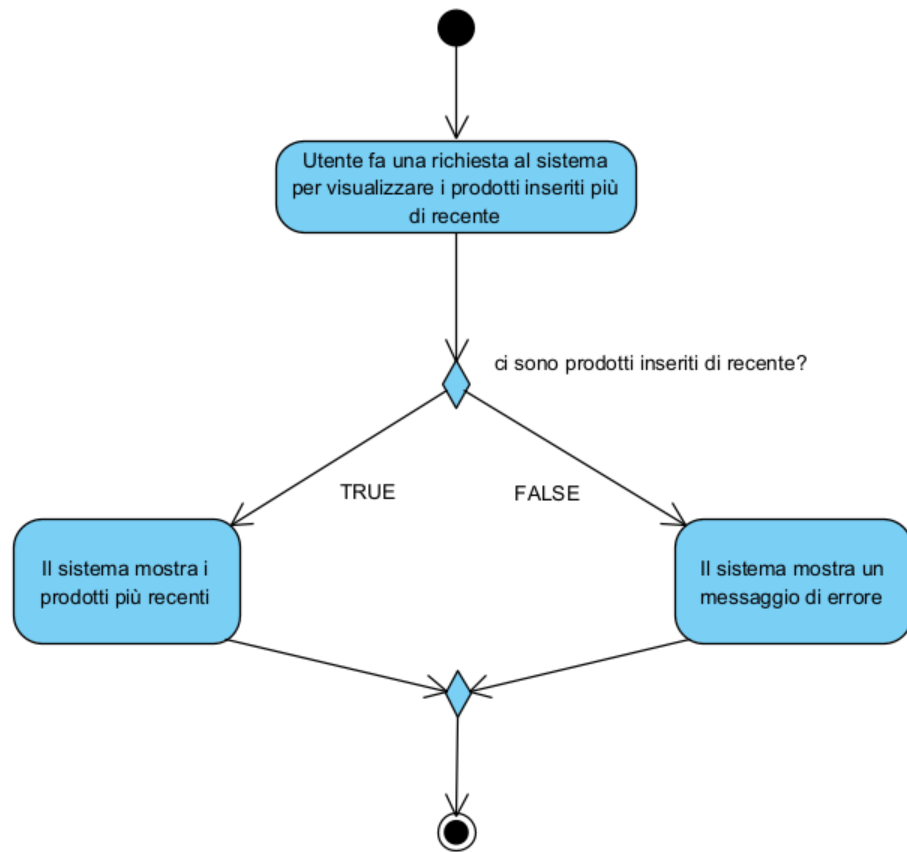
5.8.2 Activity diagram: Visualizzazione catalogo prodotti



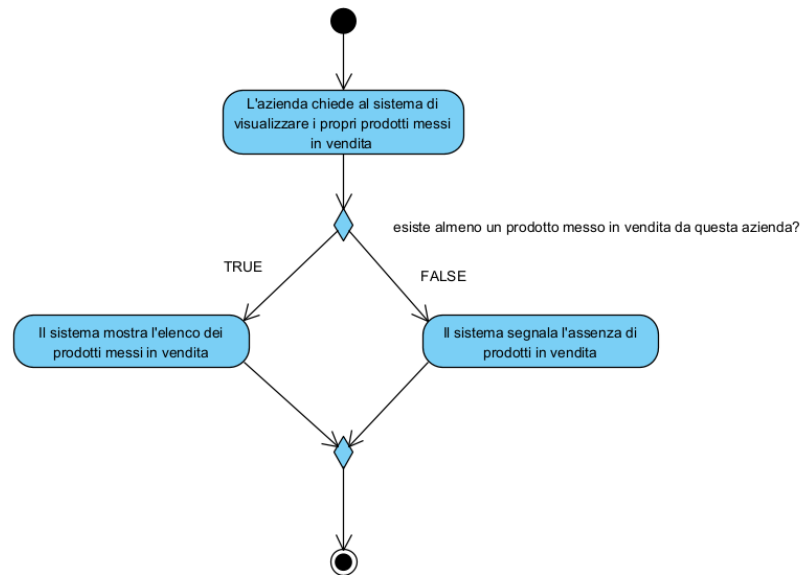
5.8.3 Activity diagram: Visualizzazione prodotti più acquistati



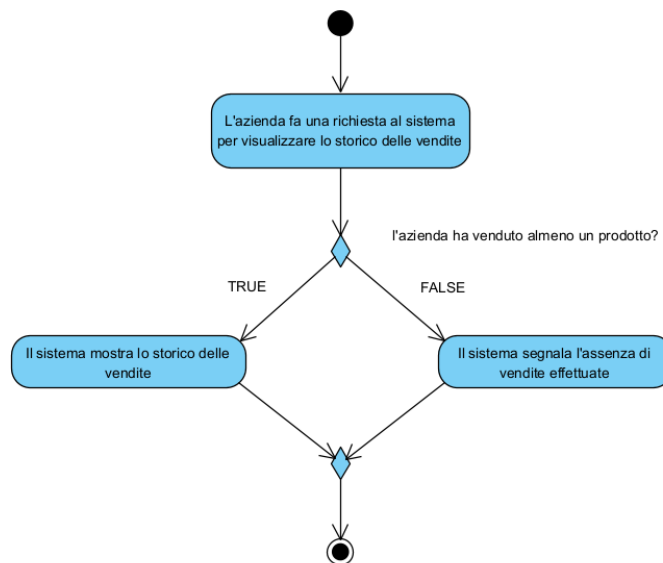
5.8.4 Activity diagram: Visualizzazione prodotti più recenti



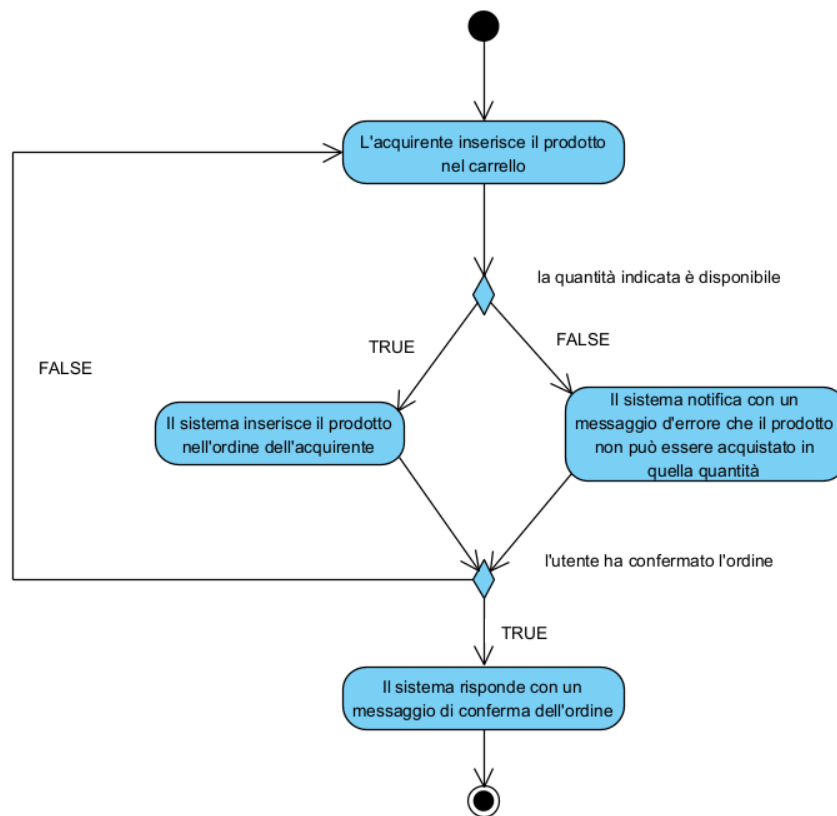
5.8.5 Activity diagram: Visualizzazione prodotti propri



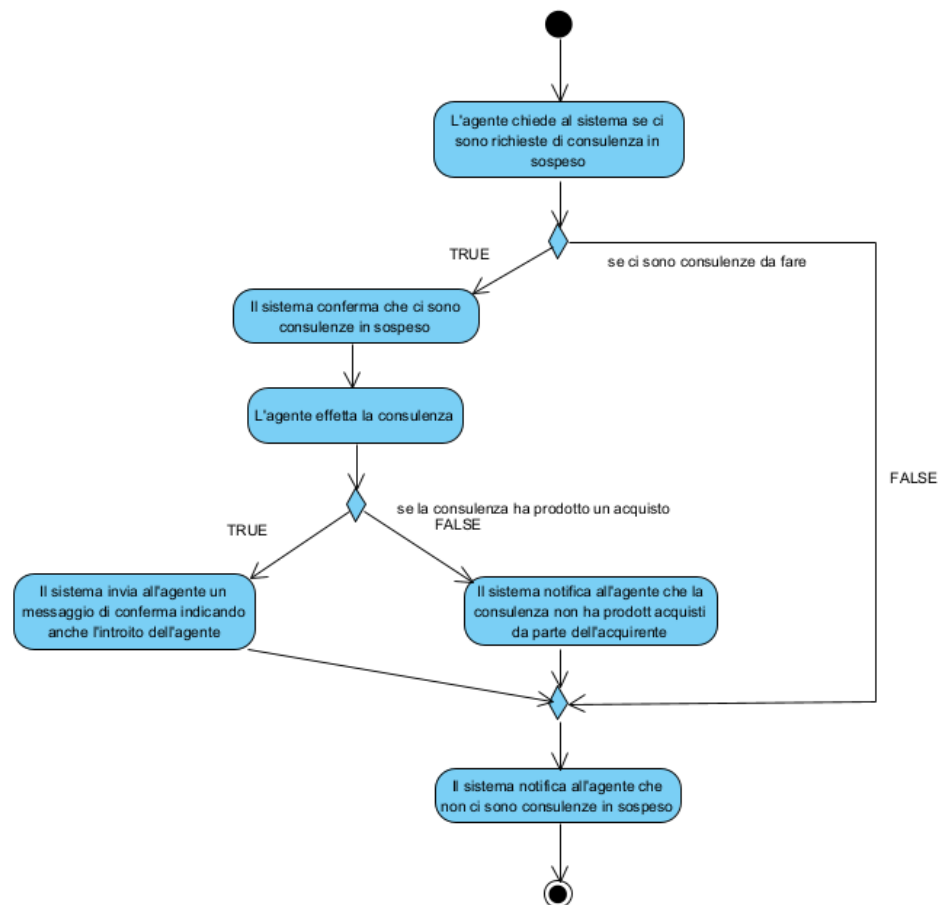
5.8.6 Activity diagram: Visualizzazione storico delle vendite



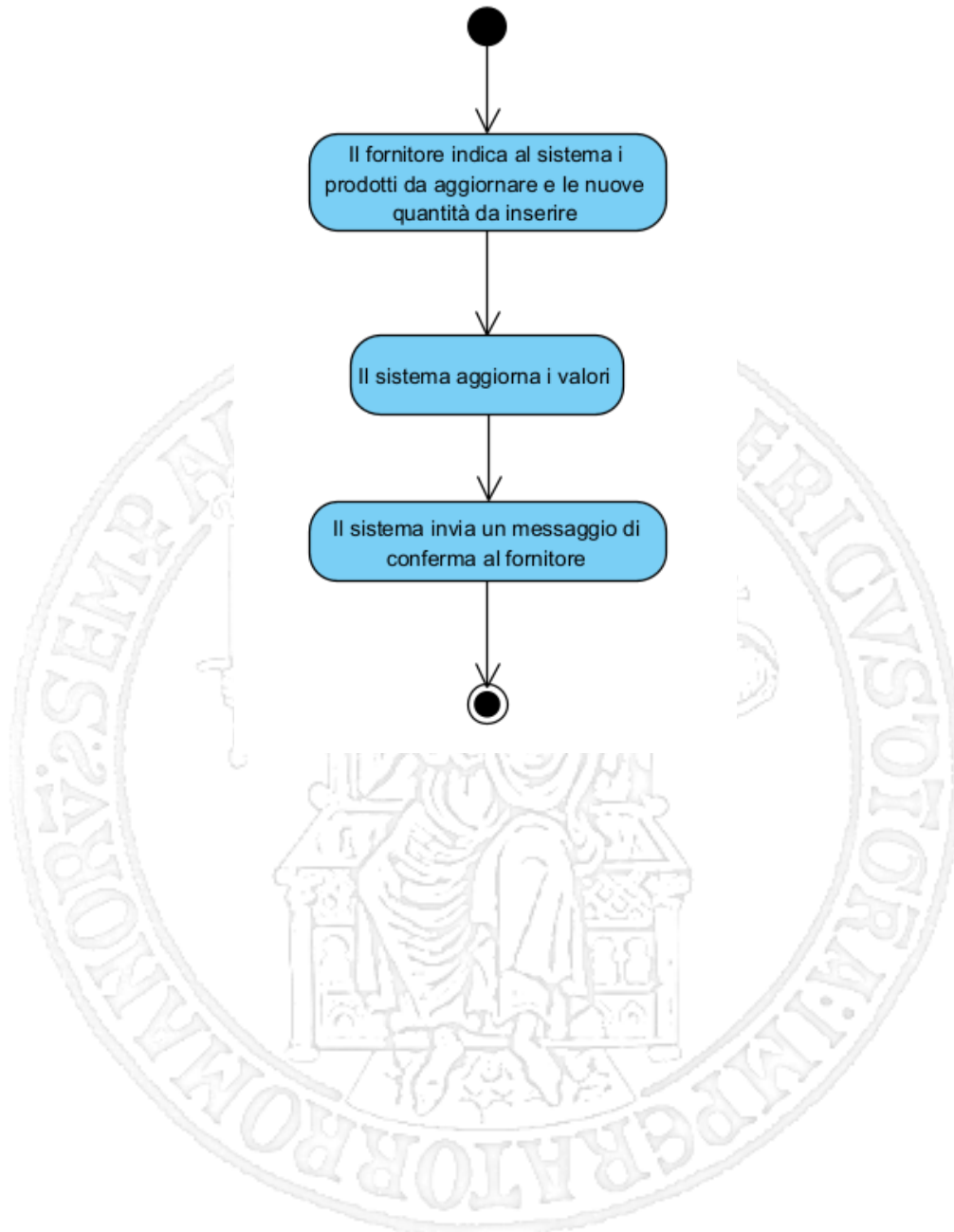
5.8.7 Activity diagram: Effettua ordine



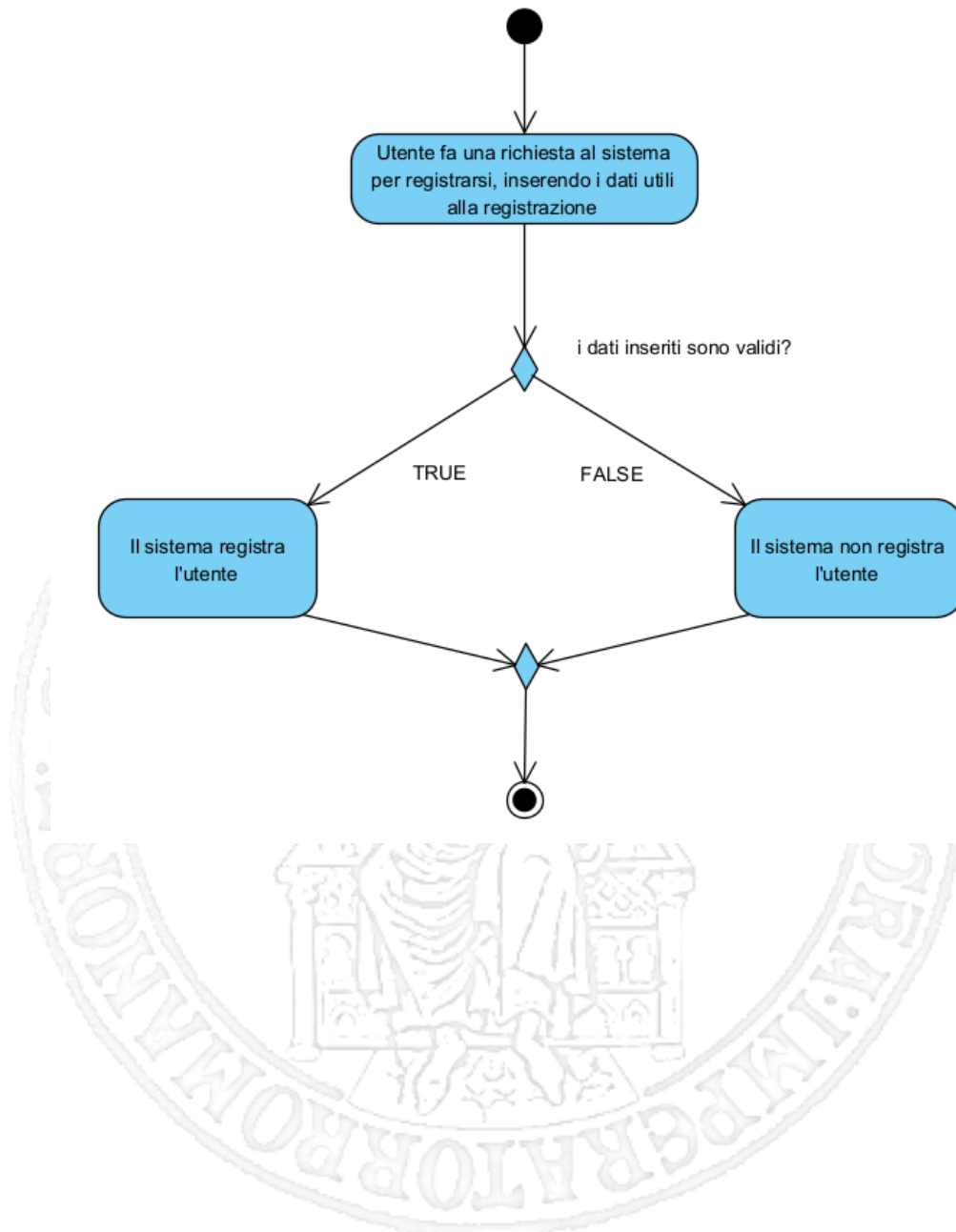
5.8.8 Activity diagram: Consulenza sui prodotti



5.8.9 Activity diagram: Fornitura del prodotto



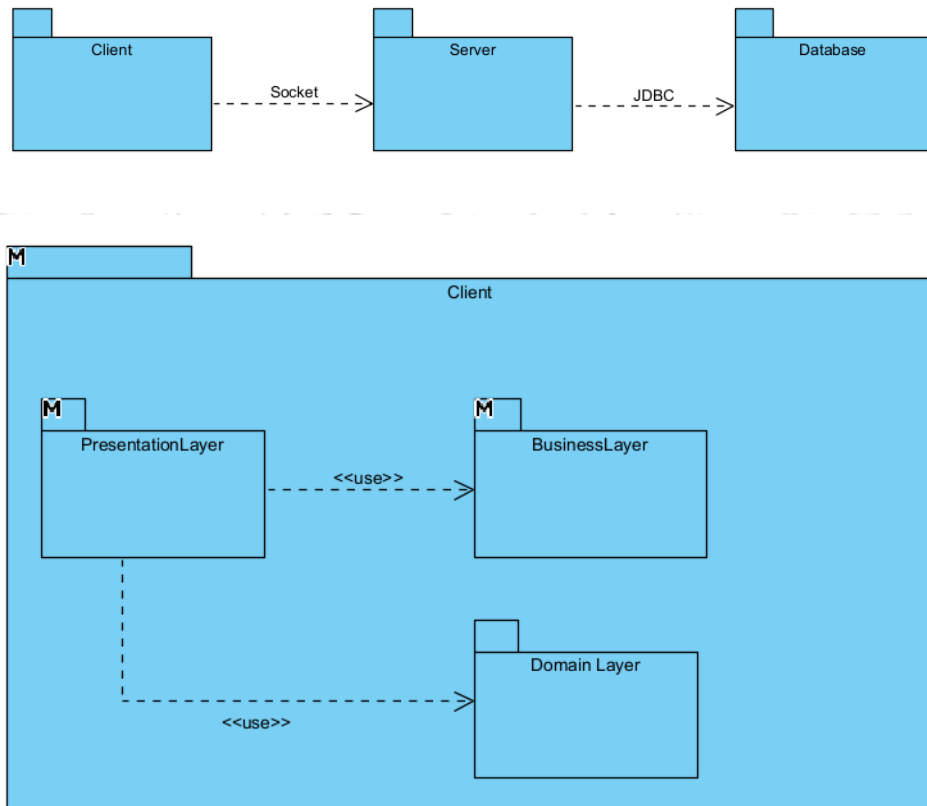
5.8.10 Activity diagram: Registrazione Utente

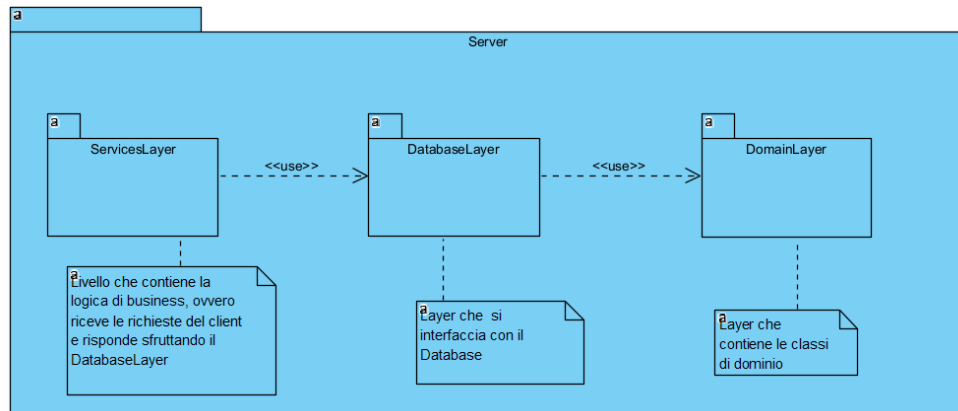


6 Architettura software

All'interno di un sistema software è importante avere una vista di come sarà l'architettura di quest'ultimo. In particolare, una architettura software è definita dai suoi componenti, dalle relazioni reciproche tra i componenti e con l'ambiente e dai principi che ne governano la progettazione e l'evoluzione. Infatti, descrivere l'architettura software di un sistema significa elencarne le sotto parti costituenti ed illustrarne i rapporti inter-funzionali. Per la descrizione dell'architettura utilizzata ci si è avvalso di uno stile architetturale di tipo Client-Server; in particolare tale stile permette di ripartire l'elaborazione tra client e server ed evitare così scarse prestazioni, e inoltre permette l'interoperabilità tra sistemi, dando la possibilità a diversi componenti di lavorare assieme.

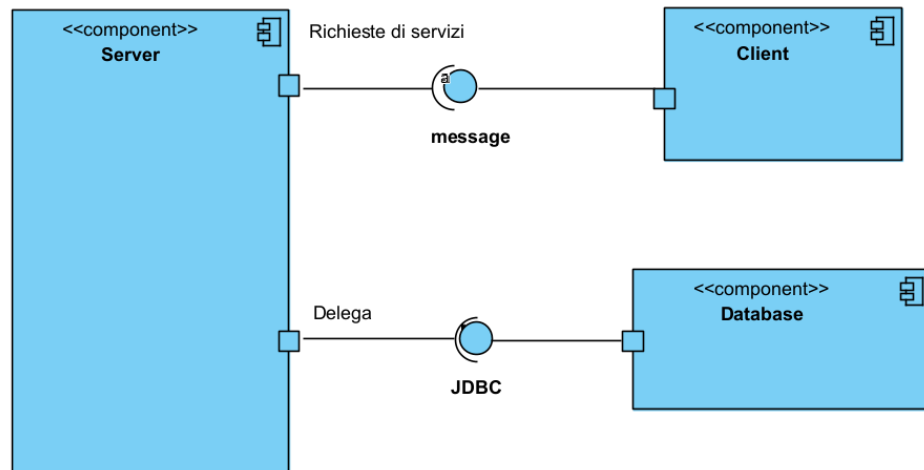
6.1 Viste architetturali di alto livello





6.2 Component diagram

Per avere una vista della struttura interna del software modellato attraverso le interazioni tra i vari componenti del sistema stesso, è utile stendere un diagramma dei componenti.



6.3 Descrizione dell'architettura di dettaglio

Scendendo più nel dettaglio dell'architettura, come stile architetturale si è scelto il client/server: sono presenti 2 tier separati, il client e il server, e la comunicazione tra i due tier avviene mediante socket. Il server a sua volta comunica con un database mediante connettore JDBC. Dal punto di vista logico, sia il Client sia il Server sono divisi in tre livelli.

Per quanto riguarda il Client:

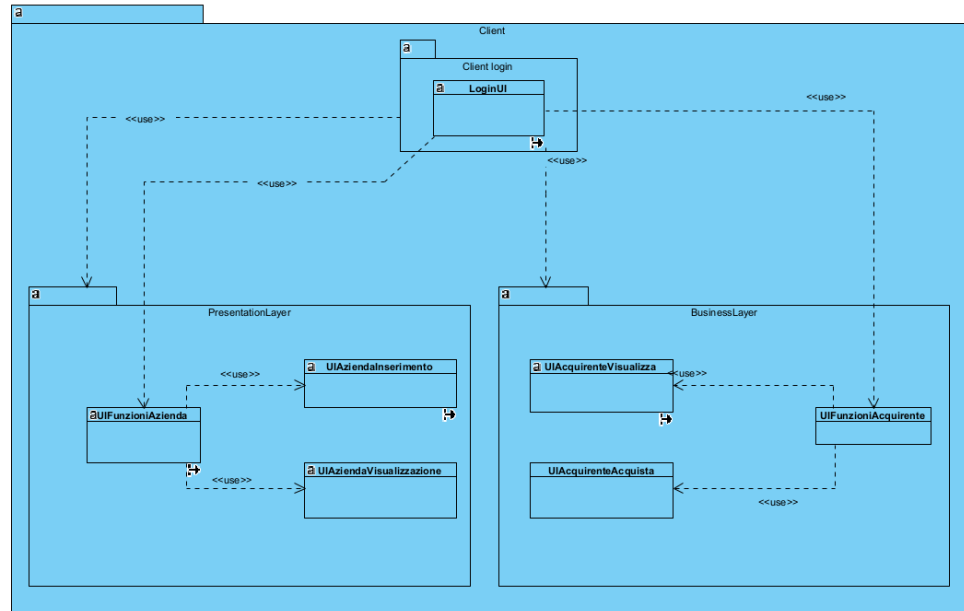
- **Presentation Layer:** è il livello che si occupa dell'interfacciamento con l'utente;
- **Domain Layer:** contiene gli oggetti di dominio che si occupano di realizzare le responsabilità richieste;
- **Business Layer:** si occupa dell'elaborazione dei messaggi ricevuti dallo scambio col server.

Per quanto riguarda il Server:

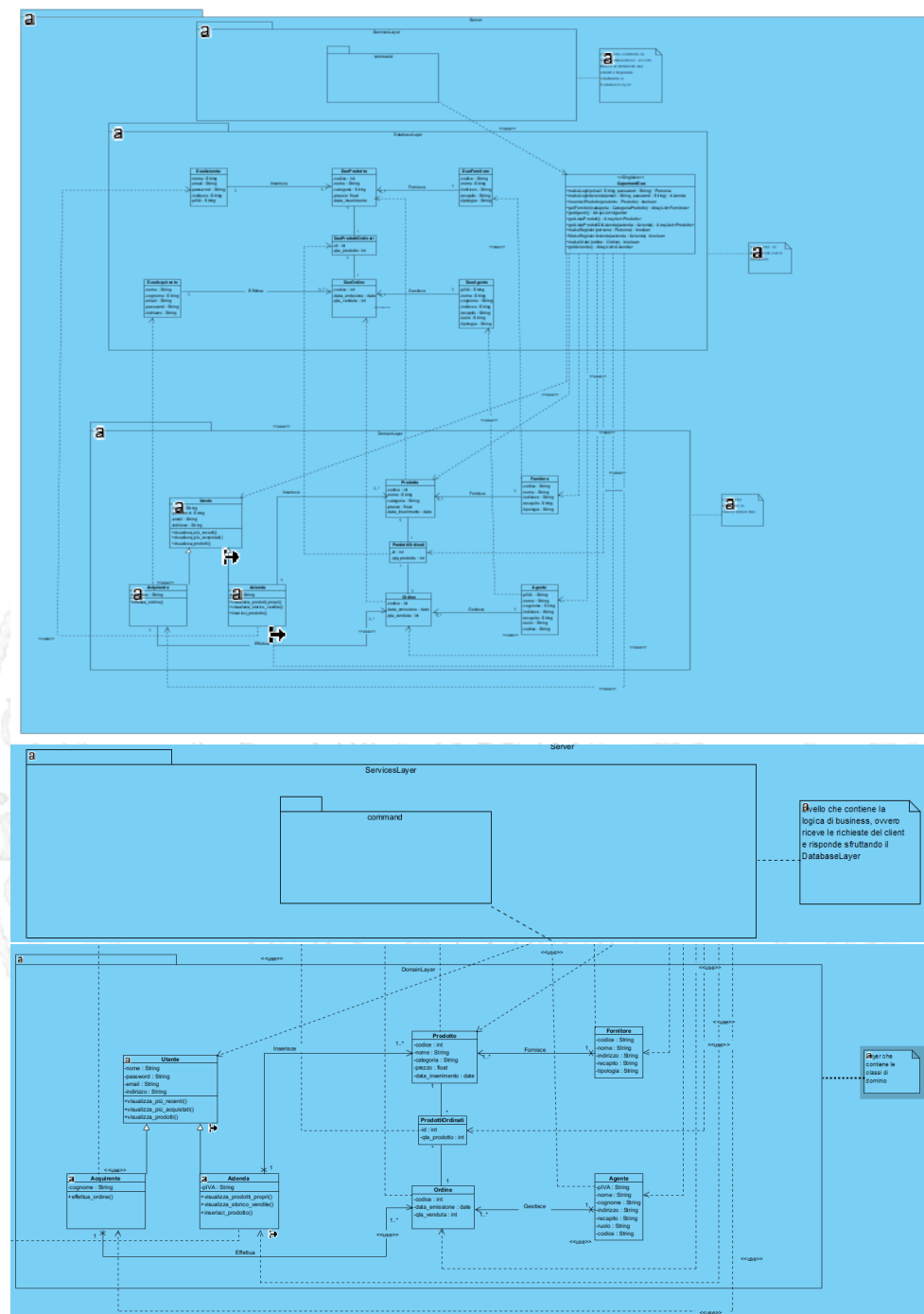
- **Service Layer:** contiene la logica di business, prende le richieste lato Client e le inoltra al Domain Layer;
- **Domain Layer:** contiene gli oggetti di dominio che si occupano di realizzare le responsabilità richieste;
- **Database Layer:** si occupa di comunicare col database mediante JDBC per accedere ai dati.

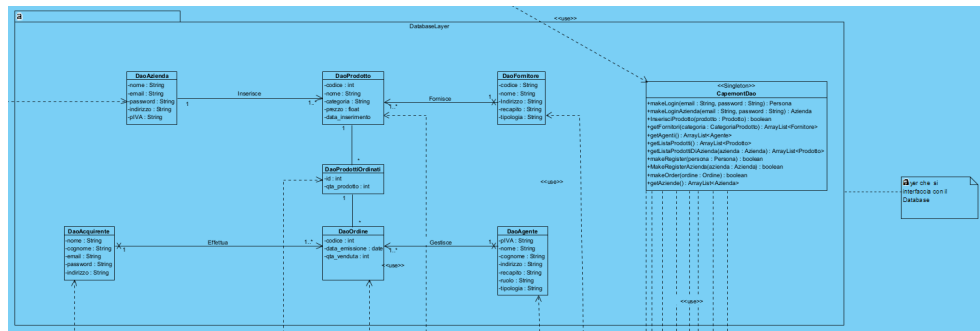
I livelli sono di tipo **closed**, ovvero un livello può comunicare solamente con il livello subito inferiore e non può "bypassarlo": questa scelta può portare ad ulteriori rallentamenti, ma si è scelto di operare in questo modo in quanto il numero di livelli è relativamente basso, e mediante questa scelta è possibile ottenere il massimo disaccoppiamento. Il client è di tipo **thick**, ovvero contiene un minimo di logica applicativa per l'elaborazione dei messaggi e archiviazione delle informazioni. Per problemi di visualizzazione dei diagrammi all'interno del documento, è convenuto separare le implementazioni del lato client e del lato server in due diagrammi differenti. Di seguito vengono mostrati i suddetti diagrammi ma, per una più corretta visualizzazione, si rimanda al file in Visual Paradigm contenente i diagrammi inseriti nel documento.

6.3.1 Diagrammi architetturali di dettaglio lato client



6.3.2 Diagrammi architetturali di dettaglio lato server



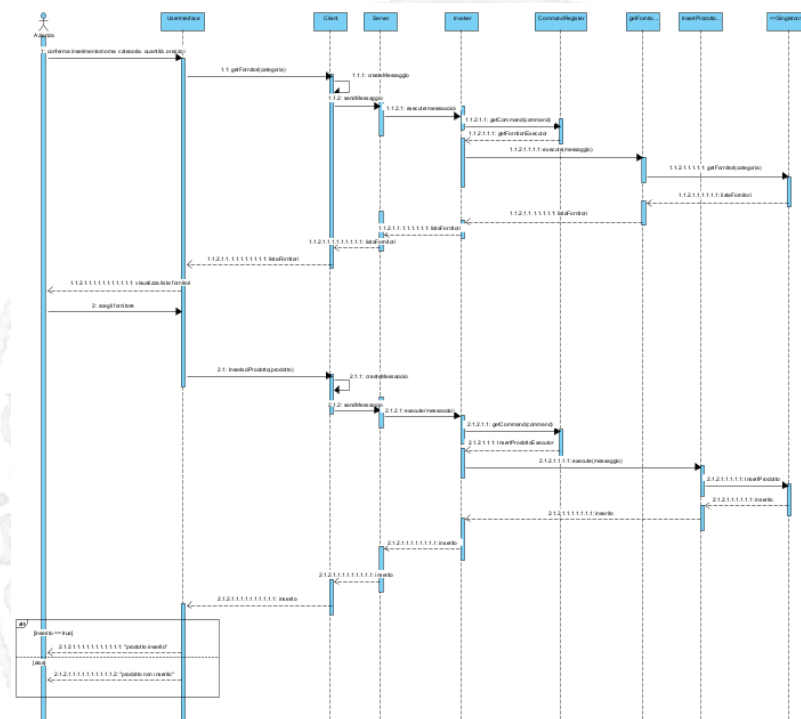


6.3.3 Sequence diagrams di dettaglio

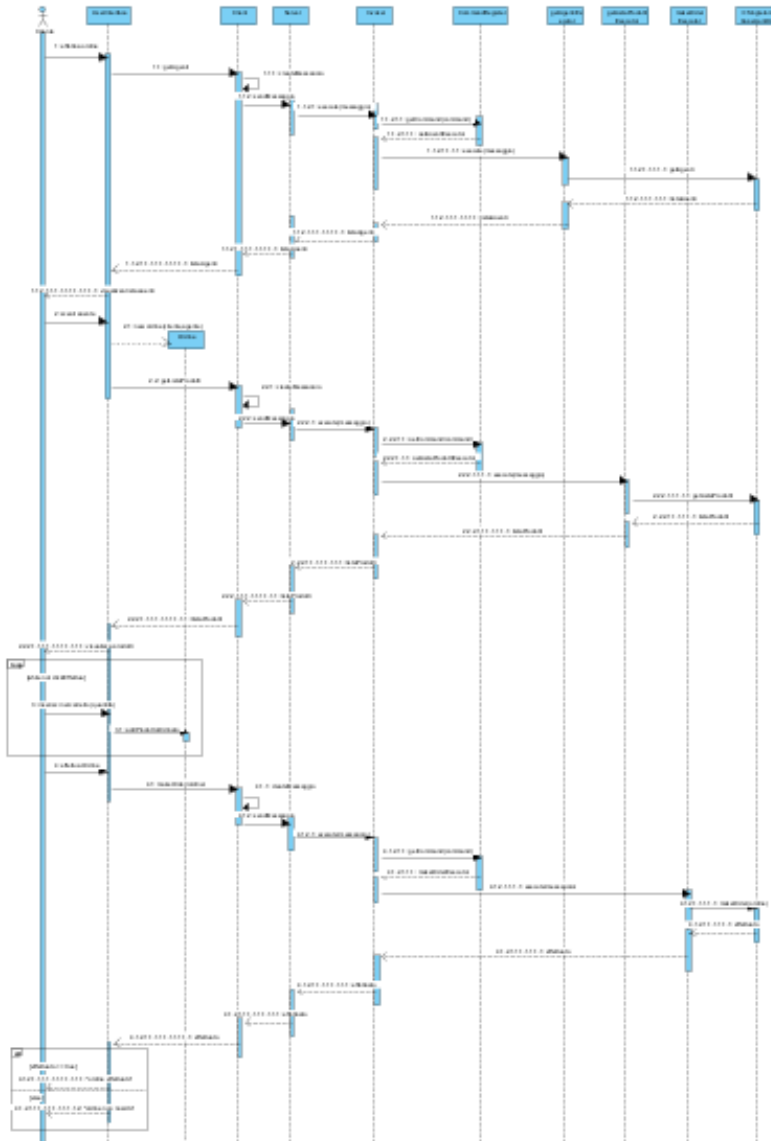
Un diagramma di sequenza di dettaglio può essere usato per definire gli input e gli output del sistema da realizzare. Esso rappresenta un particolare scenario di caso d'uso, dove per scenario si intende una determinata sequenza di azioni in cui tutte le scelte sono già state effettuate. Il diagramma di sequenza di dettaglio descrive le relazioni che intercorrono in termini di messaggi tra Attori, Oggetti di business, Oggetti o Entità del sistema che si sta rappresentando.

NOTA: per una corretta visualizzazione dei diagrammi si rimanda al file contenente la modellazione

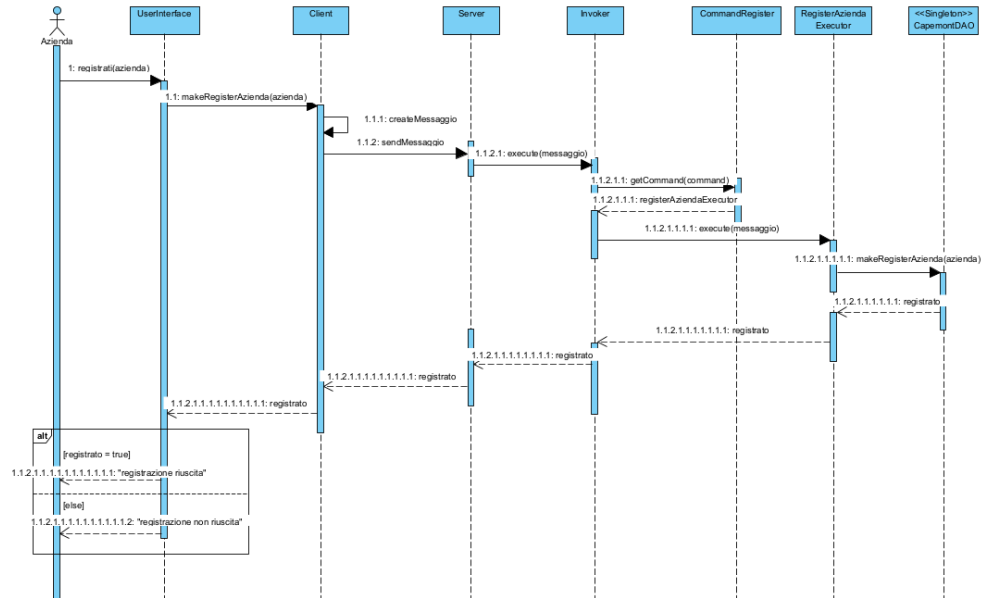
6.3.4 Sequence diagram: Inserisci prodotto



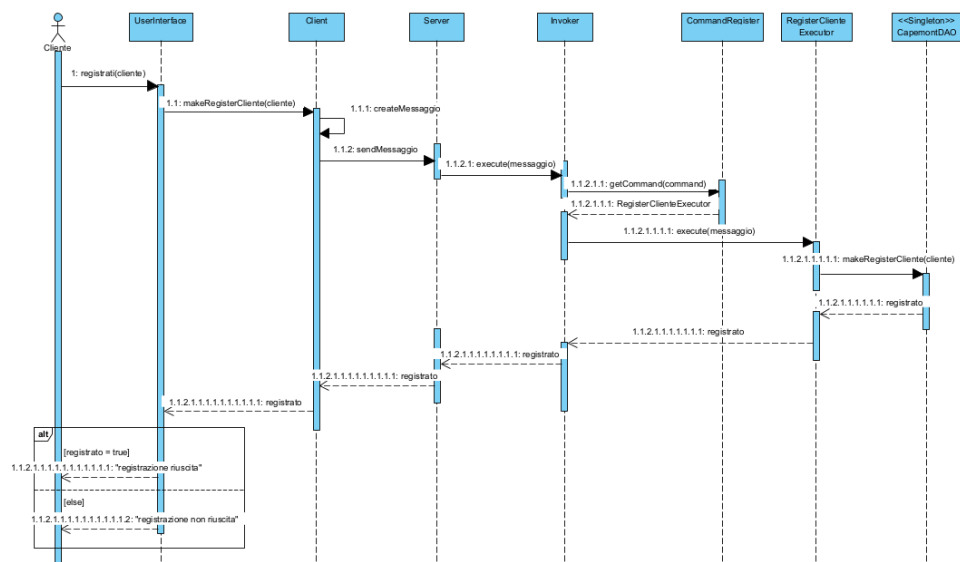
6.3.5 Sequence diagram: Effettua Ordine



6.3.6 Sequence diagram: Register Azienda



6.3.7 Sequence diagram: Register Acquirente



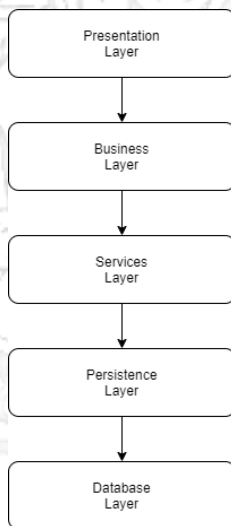
6.4 Pattern architetturale: Architettura a livelli

In una prima fase si è scelto di implementare come patter architetturale lo State Logic Display, è una particolare architettura multitier che comprende 3 livelli fisicamente separati; questi livelli prevedono: una user interface che si occupa di interfacciarsi con il livello di applicazione, quest'ultimo si fa carico di andare a comunicare con il livello di stato che sostanzialmente consiste nel database dove ci sono tutti i dati.

Questa è l'architettura prescrittiva, poi in un seconda fase di sviluppo è stato scelto di implementare un ulteriore livello che è il livello di business, che fa da "adapter" tra le richieste del livello di presentation e i messaggi da scambiare con la macchina remota. Il modello di architettura più comune è il **modello di architettura a strati** altrimenti conosciuto come il modello di architettura a n livelli(multi-layer). Questo è lo standard de facto per la maggior parte delle applicazioni Java EE e quindi è ampiamente conosciuto dalla maggior parte degli architetti, progettisti e sviluppatori. Il modello di architettura a strati corrisponde da vicino alle strutture comunicazione IT tradizionale e alle strutture organizzative che si trovano nella maggior parte delle aziende.

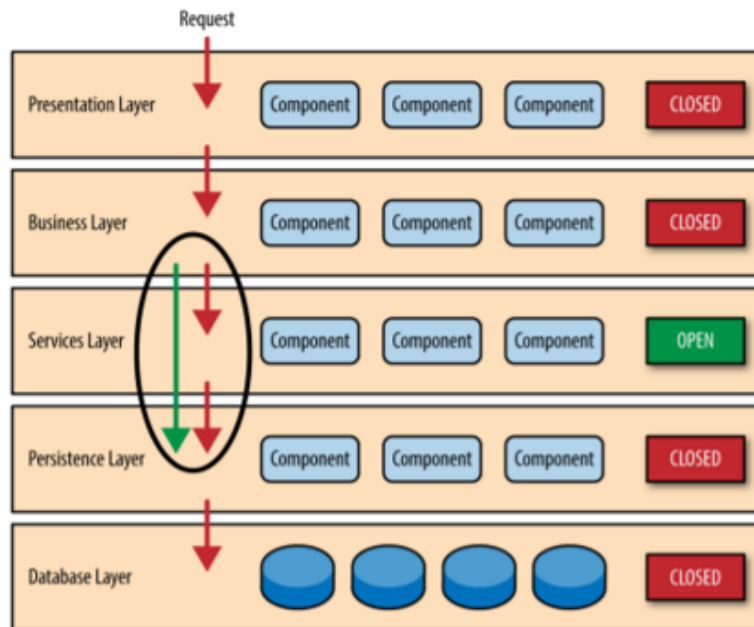
I componenti all'interno del modello di architettura a livelli sono organizzati in strati orizzontali. Sebbene il modello di architettura a strati non specifichi il numero e i tipi di strati che devono esistere nel modello, nella maggior parte consiste di quattro layer standard:

- presentation;
- business;
- services;
- persistence.



In alcuni casi, il livello di business e il livello di persistenza sono combinati in un unico livello. Ogni strato del modello ha un ruolo e una responsabilità all'interno dell'applicazione. Per esempio, un livello di presentazione sarebbe responsabile della gestione di tutta l'interfaccia utente e della logica di comunicazione del browser, mentre un livello di business sarebbe responsabile dell'esecuzione di specifiche regole di business associate alla richiesta. Ogni livello nell'architettura forma un'astrazione intorno il lavoro che deve essere fatto per soddisfare una particolare richiesta. Per esempio, il livello di presentazione non ha bisogno di sapere o preoccuparsi di come ottenere i dati del cliente; ha solo bisogno di visualizzare quelle informazioni su uno schermo in un particolare formato. Questo tipo di classificazione dei componenti rende facile costruire ruoli efficaci e modelli di responsabilità nella nell'architettura, e rende anche facile sviluppare, testare, governare e mantenere le applicazioni che utilizzano questo modello di architettura grazie a interfacce dei componenti ben definite e ambito limitato dei componenti.

Ciascuno dei livelli dell'architettura è segnato come chiuso. Questo è un concetto molto importante nel modello. Un livello chiuso significa che quando una richiesta si muove da un livello all'altro, deve passare attraverso il livello sottostante per arrivare al livello successivo. Per esempio, una richiesta che proviene dal livello di presentazione deve prima passare attraverso il livello di business e poi al livello di services prima di raggiungere finalmente il livello di persistenza (che include quello di database). Il concetto di livelli di isolamento significa che i cambiamenti fatti in un layer dell'architettura generalmente non hanno impatto o influenza sui componenti in altri livelli: il cambiamento è isolato ai componenti all'interno di quel strato, e possibilmente ad un altro strato associato. Il concetto di isolamento dei livelli significa anche che **ogni livello è indipendente dagli altri livelli**, avendo così poca o nessuna conoscenza del funzionamento interno degli altri livelli dell'architettura. Mentre i livelli chiusi facilitano i livelli di isolamento e quindi aiutano isolare i cambiamenti all'interno dell'architettura, ci sono momenti in cui ha senso che alcuni livelli siano aperti. Per esempio, supponiamo che si voglia aggiungere un livello di servizi condivisi ad un'architettura che contiene componenti di servizio comuni a cui accedono i componenti all'interno del business layer.



Creare un livello di servizi è di solito una buona idea in questo caso perché, da un punto di vista architetturale limita l'accesso ai servizi condivisi al business layer. Senza un livello separato, non c'è nulla che limiti, da un punto di vista architetturale, il livello di presentazione dall'accesso a questi servizi comuni. Come illustrato, il livello dei servizi in questo caso è contrassegnato come aperto, il che significa che le richieste possono "bypassare" questo livello aperto e andare direttamente al livello sottostante. Sfruttare il concetto di livelli aperti e chiusi aiuta a definire la relazione tra i livelli dell'architettura e i flussi di richiesta e fornisce anche ai progettisti e agli sviluppatori le informazioni necessarie per comprendere le varie restrizioni di accesso ai livelli all'interno dell'architettura. L'incapacità di documentare o comunicare correttamente quali livelli dell'architettura sono aperti e chiusi (e perché) di solito porta architetture strettamente accoppiate e fragili ad essere molto difficili da testare, mantenere e distribuire.

Nella nostra architettura i livelli sono:

- **GUI: Presentation Layer**
- **Client: Business Layer**
- **Server: Services Layer**
- **DB: Persistence Layer**

6.4.1 Caratteristiche dell'architettura

- **Agilità complessiva**

Valutazione: Basso

L'agilità generale è la capacità di rispondere rapidamente a un ambiente in costante cambiamento. Mentre il cambiamento può essere isolato attraverso i livelli di isolamento di questo modello, è ancora ingombrante e dispendioso in termini di tempo fare cambiamenti in questo modello a causa della natura monolitica della maggior parte delle implementazioni, così come lo stretto accoppiamento dei componenti che di solito si trova con questo modello.

- **Facilità di distribuzione**

Valutazione: Basso

A seconda di come si implementa questo pattern, l'implementazione può diventare un problema, in particolare per applicazioni di grandi applicazioni. Una piccola modifica ad un componente può richiedere modifiche a catena dell'intera applicazione (o di una grande porzione dell'applicazione), con il risultato che i deployment devono essere pianificati, programmati ed eseguiti fuori orario o nei fine settimana. Come tale, questo modello non si presta facilmente a una pipeline di consegna continua, riducendo ulteriormente la valutazione complessiva per distribuzione.

- **Testabilità**

Valutazione: Alto

Poiché i componenti appartengono a strati specifici dell'architettura, gli altri livelli possono essere presi o da fonti esterne o può esserne implementato lo stub, rendendo questo pattern relativamente facile da testare. Uno sviluppatore può usare un componente di presentazione per isolare i test all'interno di un componente di business, così come "mokkare" il livello di business per testare certe funzionalità del livello presentazione.

- **Performance**

Valutazione: Basso

Mentre è vero che alcune architetture stratificate possono performare bene, il modello non si presta ad applicazioni ad alte prestazioni a causa delle inefficienze di dover passare attraverso più livelli dell'architettura per soddisfare una richiesta di business.

- **Scalabilità**

Valutazione: Basso

A causa della tendenza verso implementazioni strettamente accoppiate e implementazioni monolitiche di questo pattern, rendono le applicazioni che si basano su esso generalmente difficili da scalare. È possibile scalare un'architettura a strati suddividendo gli strati in distribuzioni fisiche separate o replicando l'intera applicazione in nodi multipli, ma nel complesso la granularità è troppo ampia, rendendo costoso da scalare.

- **Facilità di sviluppo**

Valutazione: Alto

La facilità di sviluppo ottiene un punteggio relativamente alto, soprattutto perché questo modello è così ben noto e non è eccessivamente complesso da implementare. Poiché la maggior parte delle aziende sviluppa applicazioni separando gli insiemi di competenze per livelli (presentazione, business, database), questo modello diventa una scelta naturale per la maggior parte degli sviluppi di applicazioni aziendali.

6.5 Stile architetturale

Uno stile architetturale è una collezione di decisioni di design architetturale che sono applicabili in un dato contesto di sviluppo, vincolano le decisioni di design architetturale e raggiungono un livello di qualità alto; sono meno specifici rispetto ai pattern architetturali e sono utili su più domini e a vari livelli di dettaglio. Si è voluto usare uno stile architetturale per i diversi vantaggi che esso comporta:

- **Design reuse:** Soluzioni riusabili;
- **Riuso del codice;**
- **Comprensibilità della struttura del sistema;**
- **Interoperabilità;**

Per la realizzazione della piattaforma è stato utilizzato uno stile **Object Oriented** abbinato ad uno stile **Multilayered** a livelli chiusi. Con lo stile OO, si sono sfruttati tutti i vantaggi introdotti da quest'ultimo:

- La possibilità di mappare gli oggetti della realtà sugli elementi del software, aumentando la leggibilità;
- Introduzione dell'information hiding che si traduce in un basso accoppiamento tra le varie classi.

Con lo stile closed layers, i vantaggi sono molteplici:

- Con l'introduzione dell'API usate tra i livelli adiacenti c'è una netta separazione tra i vari livelli, garantendo sicuramente evolvibilità;
- Semplicità di riuso, andando semplicemente a modificare l'implementazione interna senza impattare sul sistema complessivo.

Tuttavia, nello stile object oriented tra gli svantaggi c'è:

- Tutti gli oggetti hanno bisogno di altri oggetti in quanto devono conoscere i metodi degli altri;
- Data la semplicità dei metodi ci può essere una complessità maggiore perché bisogna fare più chiamate di metodo. Questo svantaggio è per una parte risolto grazie all'implementazione di alcuni design pattern.

6.6 Pattern comportamentale Command

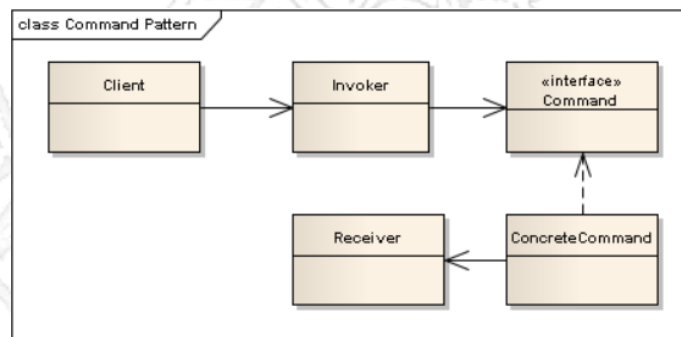
Si tratta di un pattern comportamentale basato su oggetti e viene utilizzato quando si ha la necessità di disaccoppiare l'invocazione di un comando dai suoi dettagli implementativi, separando colui che invoca il comando da colui che esegue l'operazione. Tale operazione viene realizzata attraverso la catena Client-Invoker-Receiver, dove:

- **Client:** non è tenuto a conoscere i dettagli del comando ma il suo compito è solo quello di chiamare il metodo dell' Invocatore che si occuperà di intermediare l'operazione.
- **Invoker:** ha l'obiettivo di incapsulare, nascondere i dettagli della chiamata come nome del metodo e parametri.
- **Receiver:** utilizza i parametri ricevuti per eseguire l'operazione

Tra l'Invoker ed il Receiver viene posto il Command ossia il comando da eseguire. Il Command è una semplice interfaccia che viene implementata da una o più classi concrete che invocano il Receiver.

Questo pattern è composto dai seguenti partecipanti:

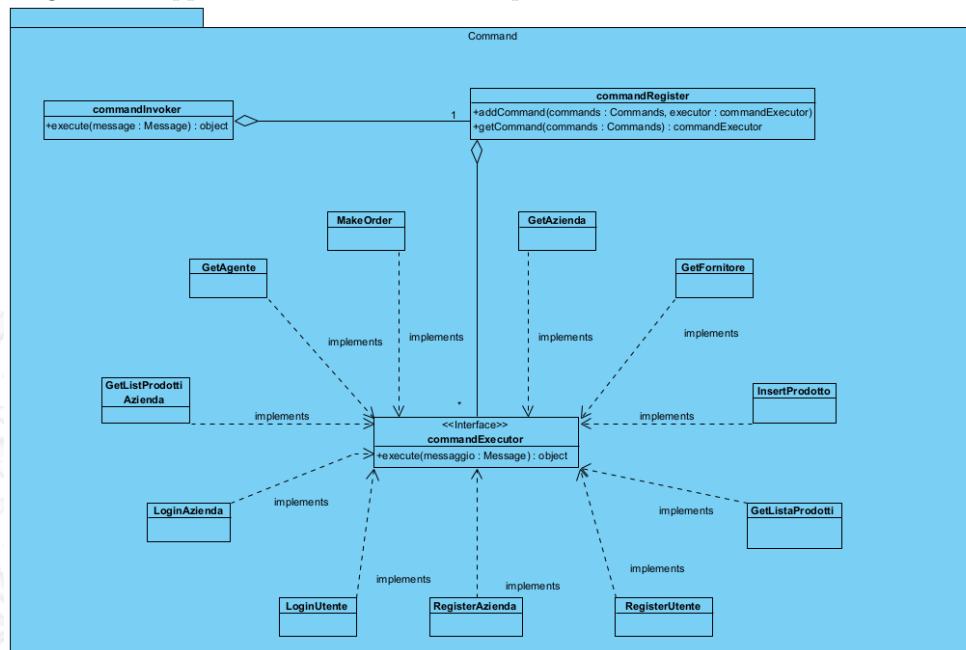
- **Client:** richiede il comando ed imposta il receiver;
- **Invoker:** effettua l'invocazione del comando;
- **Command:** interfaccia generica per l'esecuzione del comando;
- **ConcreteCommand:** implementazione del comando che consente di collegare l'invoker con il receiver;
- **Receiver:** riceve il comando e sa come eseguirlo.



Vantaggi:

- **Riduce l'accoppiamento:** il Command disaccoppia l'Invoker dal Receiver, ossia colui che invoca da colui che esegue facendo in modo che i dettagli implementativi siano a conoscenza solo del Receiver;
- **Facile estendibilità:** è possibile aggiungere facilmente nuovi comandi implementando l'interfaccia Command.

Diagramma rappresentativo delle classi del pattern:



6.7 Pattern creazionale Singleton

In certi casi è necessario avere un certo tipo di dato disponibile a tutti gli altri oggetti dell'applicazione. Di questo tipo di dato ci deve inoltre essere una sola istanza in tutta l'applicazione. Questo non può essere realizzato con una variabile esterna fa sì che l'oggetto sia accessibile globalmente, ma non impedisce che siano istanziati più oggetti. Il Singleton è un pattern creazionale che assicura che una classe abbia una sola istanza e fornisce un punto globale di accesso ad essa, e la classe stessa è responsabile di avere traccia della sua unica istanza e fornire un modo per accedere a tale istanza. Una tecnica per garantire che vi sia una sola istanza della classe consiste nel:

- 1) nascondere l'operazione di creazione in una operazione di classe(funzione static);
- 2) rendendo il costruttore protected. Il pattern è stato usato per avere un'unica istanza della classe CapemontDao.

6.8 Connettori

I connettori sono elementi architettureali che modellano le interazioni fra i componenti e le regole che governano le iterazioni stesse. Le iterazioni possono essere semplici(procedure calls) o complesse(protocolli). L'uso di connettori porta diversi vantaggi:

- Tiene separata l'elaborazione dall'interazione;
- Minimizza le interdipendenze fra i componenti;
- Supportano meglio l'evoluzione del sistema;
- Supportano la dinamicità;
- Facilitano l'eterogeneità;
- Diventano punti di distribuzione;
- Aiutano l'analisi e il testing del sistema.

Per la realizzazione della piattaforma sono stati usati diversi tipi di connettori:

- **Data Access Connectors:** Permettono di accedere a dati mantenuti in un componente di tipo data store. Nel nostro caso viene usato per l'accesso a dati persistenti, in particolare per fare query sul database.
- **Stream connectors:** Vengono usati per trasferire grandi quantità di dati fra processi autonomi; hanno in sostanza un ruolo di comunicazione. Sono risultati fondamentali per lo scambio di messaggi tra il client e il server. Per questa applicazione è stata implementata ad hoc una classe Messaggio che consente di standardizzare la comunicazione tra le due macchine, infatti tale classe ha a disposizione un attributo che specifica il comando da eseguire e un altro attributo di tipo oggetto che potrebbe eventualmente servire all'altro comando per il soddisfacimento della richiesta.

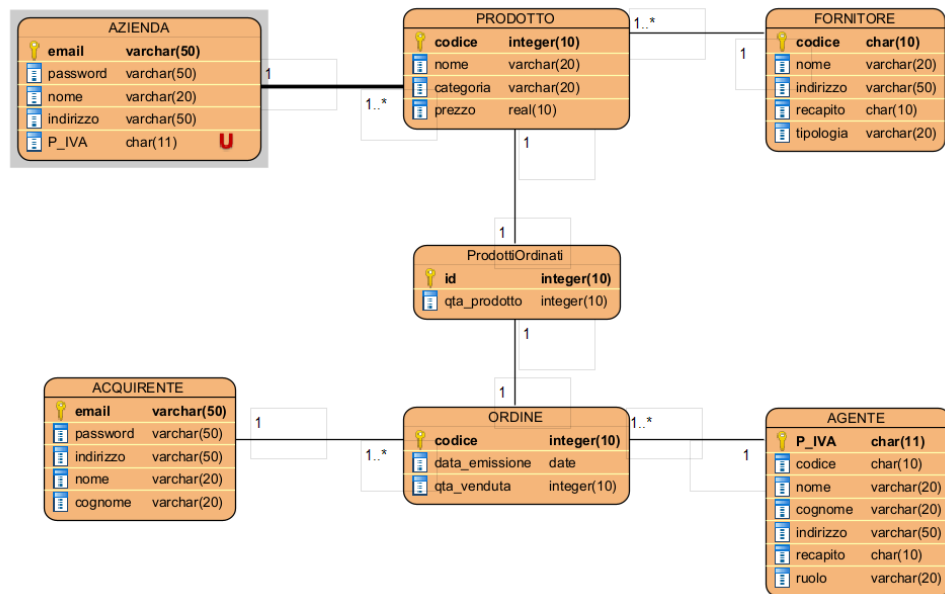
7 Persistenza dei dati

Per effettuare i salvataggi persistenti dei dati è stato implementato un database di tipo relazionale utilizzando PostgreSQL. Prima della realizzazione del database, è stato costruito il modello ER(Entity Relationship), dove sono rappresentate le entità e le associazioni tra le entità del sistema. Grazie al modello ER e alla sua traduzione è stato molto più semplice implementare in maniera corretta la base dati.

Elenco dei passi svolti:

- Costruzione del modello ER;
- Trasformazione del modello ER;
- Traduzione del modello ER
- Costruzione del database.

Di seguito il modello ER trasformato:



Il processo di traduzione del modello segue le regole qui elencate:

- **Risoluzione della gerarchia:** Si è scelto di accorpare la superclasse nelle sottoclassi, in quanto la generalizzazione è di tipo **totale-disgiunta**;
- **Traduzione di associazioni uno a uno:** Un'associazione uno a uno dello schema concettuale si traduce aggiungendo alla relazione che traduce una delle due entità, gli attributi dell'associazione e l'identificatore

dell'altra entità, eventualmente ridenominato. Esiste un vincolo di unicità sul nuovo attributo aggiunto ed un vincolo di integrità referenziale tra quest'ultimo e il corrispondente attributo dell'altra entità.

- **Traduzione di associazione uno a molti:** Un'associazione uno a molti dello schema concettuale si traduce aggiungendo alla relazione che traduce l'entità dal lato 1 gli attributi dell'associazione e l'identificatore dell'entità dal lato molti, eventualmente ridenominato. Esiste un vincolo di integrità referenziale tra questo attributo e il corrispondente attributo dell'entità del lato molti.
- **Traduzione di associazioni molti a molti:** Una associazione molti a molti dello schema concettuale si traduce in una relazione dello schema logico avente lo stesso nome dell'associazione e per attributi gli attributi dell'associazione e gli identificatori delle entità coinvolte. L'insieme di tali identificatori costituisce la chiave primaria della relazione, ed ognuno di essi ha un vincolo di integrità referenziale con il corrispondente attributo dell'entità da cui discende.

7.1 Definizione delle relazioni

- **Acquirente - Ordine:** Un acquirente può effettuare uno o più ordini, un ordine può essere effettuato da un solo acquirente.
- **Azienda - Prodotto:** Un'azienda può inserire uno o più prodotti, un prodotto può essere inserito da una sola azienda.
- **Prodotto - Fornitore:** Un prodotto può essere fornito da un solo fornitore, un fornitore può fornire più prodotti.
- **Ordine - Agente:** Un agente può offrire consulenza per uno o più ordini, un ordine può essere seguito da un solo agente.
- **Prodotto - Ordine:** Un prodotto può essere inserito in più ordini, un ordine può avere più prodotti.

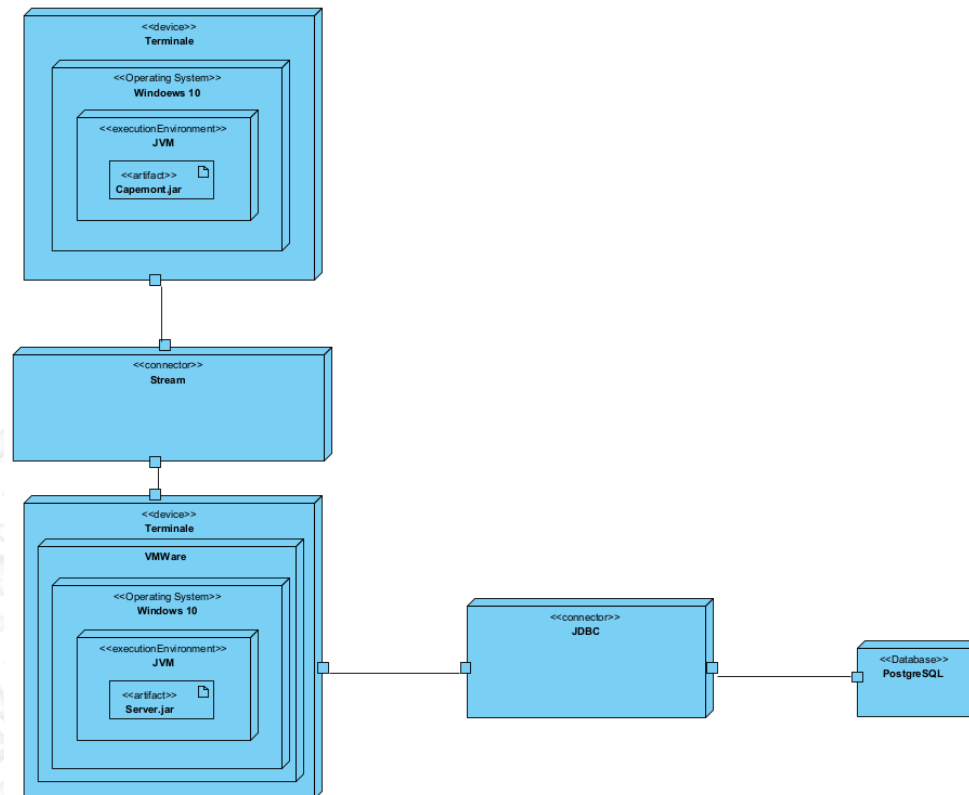
7.2 Traduzione in tabelle

- **Agenti**(piva, nome, cognome, indirizzo, email, password, recapito, ruolo)
- **Aziende**(email, password, nome, piva, indirizzo)
- **Fornitori**(codice, nome, indirizzo, tipologia, recapito)
- **Ordini**(codice, dataEmissione, qtaVenduta, codiceAgente, costo, emailUtente)
- **Acquirenti**(email, password, nome, cognome, indirizzo)
- **Prodotti**(codice, nome, categoria, prezzo, emailAzienda, codiceFornitore)
- **ProdottiOrdinati**(id, codiceProdotto, codiceOrdine, quantita)

8 Implementazione

8.1 Deployment diagram

Per rendere più chiaro il rilascio e le modalità con cui esso avviene, è sempre utile fornire un diagramma di deploy del sistema.



8.2 Documenti di implementazione

Di seguito viene riportato uno schema in cui si tiene traccia del lavoro svolto quotidianamente a livello implementativo.

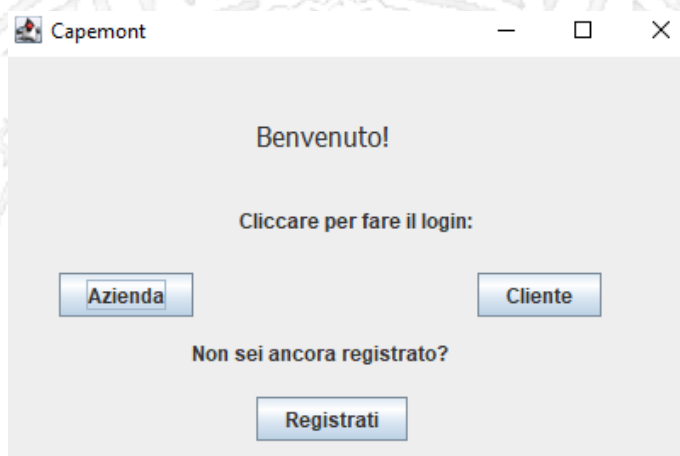
ID	Implementazione	Inizio	Fine	Durata	Status
1	Iterazione 1	Set 15, 2021	Set 25, 2021	11 d	Completato
2	Esplorazione veloce dei requisiti	Set 15, 2021	Set 16, 2021	2 d	
3	Prototipazione rapida UI	Set 17, 2021	Set 18, 2021	2 d	
4	Implementazione interfaccia di Login	Set 19, 2021	Set 21, 2021	3 d	
5	Implementazione caso d'uso "visualizza catalogo"	Set 22, 2021	Set 25, 2021	4 d	
6	Iterazione 2	Set 26, 2021	Ott 2, 2021	7 d	Completato
7	Aggiornamento rapido UI	Set 26, 2021	Set 27, 2021	2 d	
8	Implementazione caso d'uso "Inserisci prodotto"	Set 28, 2021	Ott 2, 2021	5 d	
9	Iterazione 3	Ott 3, 2021	Ott 9, 2021	7 d	Completato
10	Aggiornamento rapido UI	Ott 3, 2021	Ott 4, 2021	2 d	
11	Implementazione caso d'uso "Effettua ordine"	Ott 5, 2021	Ott 9, 2021	5 d	
12	Iterazione 4	Ott 10, 2021	Ott 13, 2021	4 d	Completato
13	Progettazione di dettaglio della UI	Ott 10, 2021	Ott 13, 2021	4 d	

8.3 Implementazione dell'applicazione

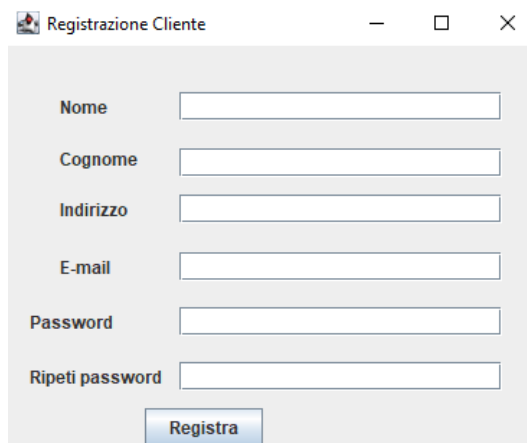
L'implementazione del sistema è stata pensata per ottenere una macchina remota lato server sempre disponibile a ricevere richieste lato client, il quale lato può comprendere anche più di un'unità. Per questo motivo, per far fronte alla possibilità di ricevere richieste in parallelo, è stato implementato un server multithread il quale gira su di una macchina virtuale all'interno di un server remoto. Per fare ciò, è stato necessario implementare una regola di forwarding per stare sempre in ascolto su una determinata porta in internet.

8.3.1 Lato Client: GUI Java

L'interfaccia prevede una serie di schermate che consentono di navigare facilmente tra i vari casi d'uso implementati. All'apertura, il portale di benvenuto consente di loggarsi in entrambi i profili, oppure di effettuare una nuova registrazione. Di seguito è riportata un'immagine di quest'ultima.



Andando in ordine, cliccando sul pulsante "Registrati" è possibile accedere ad un pannello di selezione del profilo da registrare, con opzioni ovviamente l'azienda oppure il cliente. Una volta selezione il profilo, compare un form in cui è possibile inserire tutti i dati necessari, i quali devono essere obbligatoriamente inseriti nel formato corretto. Di seguito è riportato uno dei due form di registrazione a titolo di esempio.



Registrazione Cliente

Nome

Cognome

Indirizzo

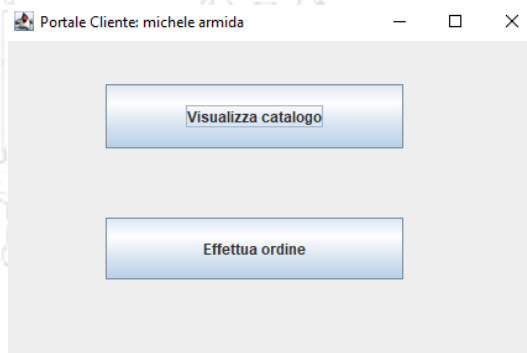
E-mail

Password

Ripeti password

Registra

Di seguito, viene mostrato il portale per il cliente. Quest'ultimo è stato implementato in modo minimale e consente di selezionare i casi d'uso implementati per il profilo menzionato.

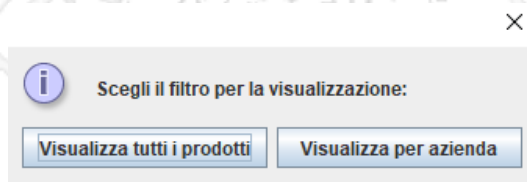


Portale Cliente: michele armida

Visualizza catalogo

Effettua ordine

Il primo caso d'uso consente di visualizzare il catalogo dei prodotti, secondo un filtro selezionato sul pannello che compare non appena viene cliccato il pulsante. Infatti, si può scegliere di visualizzare l'intero catalogo dei prodotti disponibili, oppure di visualizzare i prodotti di una determinata azienda che dovrà essere scelta successivamente. Di seguito è riportato il pannello di opzione per la visualizzazione.



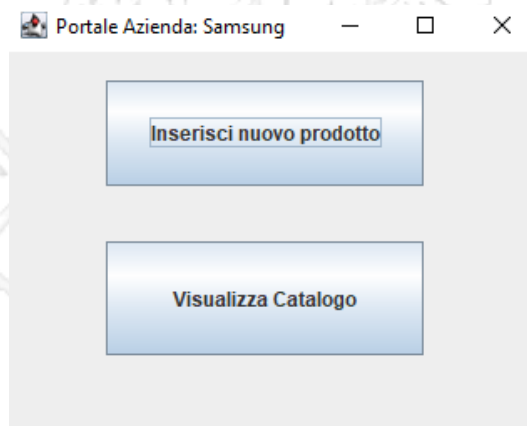
Scegli il filtro per la visualizzazione:

Visualizza tutti i prodotti **Visualizza per azienda**

Il secondo caso d'uso implementato è relativo alla creazione di un'ordine da parte di un cliente. Una volta cliccato sul pulsante relativo, seguendo i requisiti è necessario dapprima scegliere un agente di vendita di riferimento, poi comparirà il portale relativo alla gestione del carrello per l'ordine. Tale portale è composto da una tabella al cui interno sono presenti i prodotti disponibili all'ordine. Per poter inserire un determinato prodotto all'interno del carrello, basta semplicemente selezionare la riga corrispondente e cliccare sul pulsante apposito, sarà chiesta la quantità da voler ordinare e, se la quantità è coerente con quella disponibile, è possibile visualizzare il prodotto all'interno del carrello. Una volta selezionati tutti i prodotti da voler ordinare, basta cliccare su "Effettua ordine". Di seguito è riportata l'immagine del portale appena spiegato.



Si passa ora al portale dell'azienda che, come nel caso del cliente, è un'interfaccia minimale che consente di poter selezionare i casi d'uso implementati.



Il primo caso d'uso riguarda l'inserimento di un nuovo prodotto, al click del pulsante compare un form per inserire i dati del prodotto, compresa la categoria da scegliere tra quelle disponibili. Il fornitore non è presente in tale form, infatti dovrà essere selezionato in un secondo momento non appena viene completato il form. Questo perchè la lista dei fornitori disponibili dipende dalla categoria del prodotto da inserire. Di seguito è mostrato il form per l'inserimento del prodotto.

L'ultimo caso d'uso implementato riguarda una semplice visualizzazione di tutti prodotti per l'azienda loggata. Quello che comparirà è una semplice tabella contenente tali prodotto. Di seguito è mostrato un esempio.

Nome	Categoria	Quantita	Prezzo	Nome fornit.	indirizzo for...
Telefonino	elettronica	500	500.0	Euronics	via Napoli 90
Iphone	elettronica	200	1000.0	Unieuro	via Roma 80
tuta	abbigliame...	100	10.0	Dress	via Milano 50
PC	elettronica	100	600.0	Euronics	via Napoli 90
TV	elettronica	180	200.0	Euronics	via Napoli 90

8.4 Server remoto

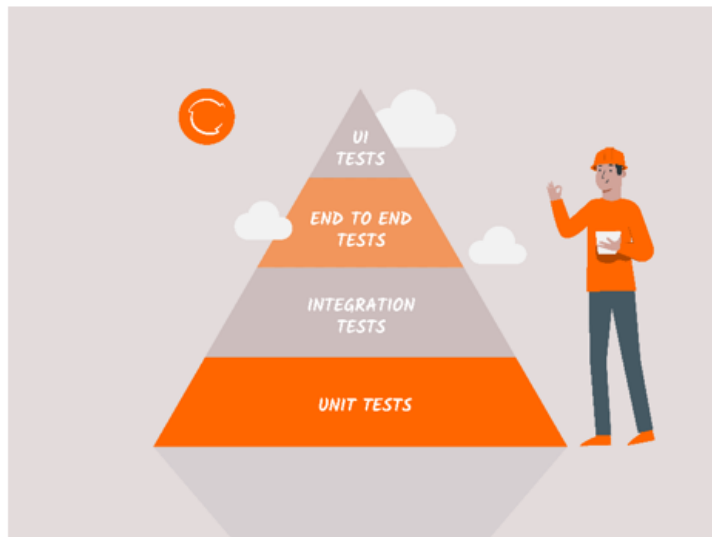
Abbiamo deciso di implementare il livello di servizio su una macchina virtualizzata remota. dire che per fare ciò è stato necessario ovviamente implementare una regola di forwarding su una porta in modo che il serve sia sempre in ascolto e sia sempre in esecuzione. Le risorse messe a disposizione dal server non sono tante perchè la logica in questo caso è molto semplice. Sulla macchina stessa è stato implementato il server di postgres, quindi è bastato l'accesso in local

host per far comunicare col database. La connessione è stata fatta mediante Java Socket. Quindi, il server resta in ascolto su una determinata porta, non appena riceve il messaggio istanzia un nuovo thread per il soddisfacimento della richiesta.

9 Testing

Il testing è un'attività fondamentale del ciclo di sviluppo del sistema software e deve essere eseguita alla fine di ciascuna iterazione per verificare la presenza di errori o imperfezioni nel codice che potrebbero portare ad un sistema poco funzionale e poco affidabile.

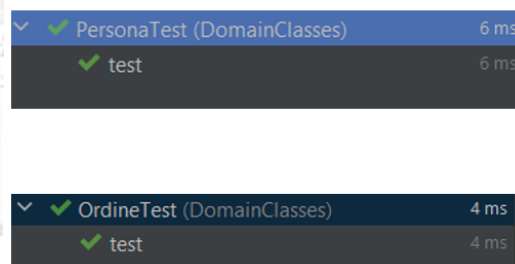
L'approccio al testing segue quello illustrato in figura, si usa un approccio bottom-up dove vengono prima testate le varie unità, in seguito vengono integrate man mano e viene testato il sistema parzialmente integrato, poi vengono fatti dei test end to end, per poi terminare col testing dell'interfaccia.



9.1 Testing di Unità

Per Testing di Unità (Unit Testing) si intende l'attività di testing di singole unità software. Per unità si intende normalmente il minimo componente di un programma dotato di funzionamento autonomo; a seconda del paradigma di programmazione o linguaggio di programmazione, questo può corrispondere per esempio a una singola funzione nella programmazione procedurale, o una singola classe o un singolo metodo nella programmazione a oggetti. Lo Unit testing viene normalmente eseguito dagli sviluppatori, e può essere occasionalmente glass box, ovvero essere esplicitamente basato sulla conoscenza dell'architettura e del funzionamento interno di un componente oltre che sulle sue funzionalità.

esternamente esposte. Come le altre forme di testing, lo unit testing può variare da completamente "manuale" ad automatico. Specialmente nel caso dello unit testing automatico, lo sviluppo dei test case (cioè delle singole procedure di test) può essere considerato parte integrante dell'attività di sviluppo (per esempio, nel caso dello sviluppo guidato da test). Lo unit testing facilita la modifica del codice del modulo in momenti successivi (refactoring) con la sicurezza che il modulo continuerà a funzionare correttamente. Il procedimento consiste nello scrivere test case per tutte le funzioni e i metodi, in modo che se una modifica produce un fallimento del test, si può facilmente individuare la modifica responsabile. Nel nostro caso possiamo vedere come sia stato fatto del testing di unità per quanto riguarda le classi Agente, Azienda, Fornitore, Prodotto, Ordine, Persona e Utente. Di queste classi non è stato testato tutto per utilità e ridondanza ma sono stati testati dei costruttori, e alcune classi get e set per ogni classe. Come si può vedere tutti i test sono andati a buon fine e si può quindi concludere che le funzioni e i costruttori testati siano perfettamente funzionanti. Qui di seguito due esempi che mostrano la correttezza dei test e la loro durata in millisecondi.



▼	✓ PersonaTest (DomainClasses)	6 ms
	✓ test	6 ms
▼	✓ OrdineTest (DomainClasses)	4 ms
	✓ test	4 ms

9.2 Testing di integrazione

Il significato del test di integrazione è quello di integrare/combinare i moduli testati uno per uno e verificare il comportamento come un'unità combinata. La funzione o l'obiettivo principale è quello di testare le interfacce tra le unità/moduli. Normalmente eseguiamo test di integrazione dopo il 'test unitario'. Una volta che tutte le singole unità sono state create e testate, iniziamo a combinare quei moduli 'Unit Tested' e procediamo col test d'integrazione. I singoli moduli vengono prima testati separatamente e poi vengono integrati uno per uno, fino a quando tutti i moduli sono integrati, per verificare il comportamento combinatorio e convalidare se i requisiti sono implementati correttamente o meno. I vantaggi che si ottengono nel testare l'integrazione dei moduli sono due:

- Ci si assicura che i moduli/componenti integrati funzionino correttamente;
- Non richiede il completamento di tutti i moduli per effettuare un test, in quanto è possibile usare Stub o Driver per sostituire temporaneamente il modulo mancante.

Nel nostro caso, andiamo a inserire nella sessione Before del testing un elemento nel database e successivamente nella parte Main del test andiamo a fare

una query sul database per selezionare l'elemento appena inserito. In questo modo si andrà a testare l'effettivo funzionamento dello scambio di informazioni tra database e piattaforma.

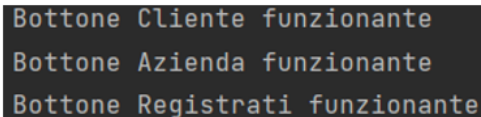
Come possiamo notare, il test d'integrazione effettuato ha dato riscontri positivi.

```
REGISTERUTENTE email_prova password_prova  
REGISTRAZIONE UTENTE EFFETTUATA  
LOGINUTENTE email_prova password_prova  
LOGIN nome_prova cognome_prova RIUSCITO
```

9.3 Testing d'interfaccia

Il testing di sistemi interattivi (in particolare di interfacce utente) viene condotto tipicamente in maniera black box. In genere, i casi di test sono progettati tenendo in conto le possibili interazioni che un utente può eseguire sull'interfaccia utente. A tale proposito, sono stati svolti due tipi di test:

- Il primo, quello sulla schermata iniziale, sono state testate le funzionalità dei bottoni presenti nell'interfaccia, ovvero Azienda, Cliente e Registrati. Il tutto è fatto andando a fondo a livello di codice nella classe JFrame e andando ad eseguire il comando `doClick()` direttamente sui bottoni. Per essere sicuri che il 'click' andasse a buon fine e che non portasse ad eccezioni, sono stati stampati a video dei messaggi di conferma.



```
Bottone Cliente funzionante  
Bottone Azienda funzionante  
Bottone Registrati funzionante
```

- Il secondo, quello sul login dell'Azienda, è stato eseguito sfruttando la classe Robot di Java. Questa prende il controllo di cursore e tastiera e, a seconda di come impostiamo i suoi movimenti, sposterà il cursore, cliccando su bottoni e campi di inserimento e andando a scrivere valori che servono per il test. In questo caso particolare sono stati inseriti i valori di email e password aziendali ed è stata effettuata la richiesta di login.

10 Sviluppi futuri

Nelle successive iterazioni saranno implementati:

- Il caso d'uso **Consulenza sui prodotti** che permette all'agente di offrire un servizio di consulenza all'acquirente sull'acquisto di uno o più prodotti.
- Il caso d'uso **Fornitura del prodotto** che permette ad un fornitore di occuparsi della fornitura di uno o più prodotti.

È possibile apportare migliorie al sistema, aggiungendo nuovi campi ai dati già memorizzati nel sistema, relativi agli acquirenti, alle aziende e ai prodotti; è possibile aggiungere ulteriori meccanismi di controllo per rendere il sistema maggiormente affidabile e sicuro. inoltre, è possibile arricchire il sistema con nuovi componenti e, con l'uso di altre tecnologie, sviluppare una Web Application.

