UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione in Python di un sistema per la gestione di un magazzino

Design and implementation of a warehouse management system in Python

Relatore

Prof. Domenico Ursino

Candidato

Daniel Diocis Calero

ANNO ACCADEMICO 2023-2024

If you had one shot or one opportunity
to seize everything you ever wanted in one moment,
would you capture it or just let it slip?

Eminem, "Lose Yourself"

Sommario

La gestione logistica del magazzino, che comprende la pianificazione, l'operatività e il controllo del flusso delle merci, è diventata essenziale per le aziende che dispongono di vasti magazzini e di un inventario ampio. La Warehouse Logistics si concentra principalmente sulla gestione efficiente dell'inventario fisico all'interno del magazzino per raggiungere gli obiettivi aziendali. In questa tesi, si sono analizzati dati fittizi relativi alla TomWare Solutions, un'azienda del settore tecnologico. L'obiettivo è stato spiegare come la progettazione del database e lo sviluppo di un sistema gestionale possano facilitare le operazioni dello staff, rendendo più agevole la gestione dei prodotti e ottimizzando i processi operativi.

Keyword: Warehouse Logistics, Database, Python, PyQt, Pytest, Management System, Physical Inventory, Efficiency

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Nell'attuale contesto aziendale, i dati rivestono un ruolo cruciale, e la capacità di gestirli efficacemente rappresenta un vantaggio competitivo significativo. Questa abilità include l'organizzazione e il trattamento di grandi quantità di dati, assicurando al contempo la loro integrità e ottimizzando le prestazioni del database. L'adozione di un software gestionale offre numerosi benefici, tra cui un miglioramento dell'efficienza operativa, l'ottimizzazione dei processi aziendali e la garanzia di dati precisi e affidabili.

Secondo un rapporto di IDC, Reinsel *et al.* [2017] i dati stanno diventando centrali per le decisioni aziendali e le operazioni. Il 95% delle grandi imprese afferma che i database hanno trasformato il modo in cui conducono il loro business, permettendo loro di prendere decisioni più informate ed efficienti (IDC, Data Age 2025).

Un sistema di gestione del magazzino basato su tecnologie moderne consente di controllare e gestire l'inventario in modo più efficace, contribuendo a ridurre i costi operativi complessivi (Habsi *et al.* [2023]). Le aziende che implementano tali tecnologie possono anche evitare errori, aumentando la precisione nella gestione delle scorte.

Esistono tanti tipi di dati e tanti modi in cui gestirli, ma non tutti sono efficienti, questo è il motivo per il quale le imprese assumono degli esperti di Warehouse Logistics. Affidandosi a un software gestionale efficiente è possibile:

- organizzare grandi quantità di dati in modo efficiente e accessibile;
- velocizzare l'acquisizione di informazioni utili per le decisioni aziendali;
- aumentare la produttività riducendo tempi e costi di gestione dei dati;
- accrescere la sicurezza e protezione delle informazioni aziendali;
- migliorare i processi decisionali sfruttando al meglio le informazioni.

Quindi un software gestionale risulta una decisione chiave per sfruttare al massimo i propri dati e trasformarli in un valore concreto e fondamentale per la propria azienda. Inoltre, un database ben progettato e gestito da un esperto rappresenta un prerequisito fondamentale (prima di integrarlo in un software gestionale) per qualsiasi azienda, in quanto consente di memorizzare e fornire accesso a grandi quantità di dati in modo efficiente, organizzato e facilmente recuperabile.

La presente tesi è composta da cinque capitoli strutturati nel seguente modo:

- Nel Capitolo 1 dopo aver svolto l'intervista con il cliente, viene delineato il contesto del progetto e si fornisce una descrizione dettagliata dei requisiti funzionali e non funzionali del sistema gestionale per il magazzino. Si analizzeranno i problemi aziendali da risolvere e gli obiettivi che il software mira a raggiungere.
- Nel Capitolo 2 si passa alla progettazione del database, con una descrizione approfondita dei dati rilevanti e la presentazione dei diagrammi E-R (Entità-Relazione) per rappresentare le relazioni tra le entità principali del sistema.
- Nel Capitolo 3 verrà esposta la progettazione della componente applicativa. Saranno descritti l'architettura client-server adottata e i diagrammi UML (classi, sequenza, attività) che rappresentano il funzionamento e l'interazione tra le varie componenti del software.
- Il Capitolo 4 sarà dedicato all'implementazione del sistema, riportando in dettaglio le scelte tecniche adottate durante lo sviluppo. Verranno, inoltre, descritti i test effettuati per validare le funzionalità e garantire il corretto funzionamento del software.
- Nel Capitolo 5 sarà presentato un manuale utente, che spiegherà come utilizzare il software gestionale in modo efficiente, includendo le principali operazioni di inserimento, modifica e gestione dei dati nel magazzino.

CAPITOLO 1

Definizione del contesto e specifica dei requisiti

In questo capitolo iniziale l'obiettivo è quello di dare una definizione del contesto per poter poi analizzare le informazioni raccolte. Evidenzieremo le parole chiave in un opportuno glossario. Questo processo iniziale è di fondamentale importanza per sviluppare i requisiti del progetto. Inoltre illustreremo l'analisi dei requisiti e i diagrammi dei casi d'uso.

1.1 Raccolta informazioni (intervista)

Il 17 Luglio 2024 abbiamo avuto un appuntamento con X., un imprenditore che ha aperto di recente un negozio di abbigliamento. X. ci ha contattato per lo sviluppo di un software gestionale volto ad ottimizzare il flusso del lavoro e dare una mano ai dipendenti.
Di seguito sarà riportata la conversazione:

D.: Buongiorno X., le dò il benvenuto nel nostro ufficio.

X.: Buongiorno, la ringrazio.

D.: Mi aveva chiesto riguardo allo sviluppo di un software per la sua nuova azienda, dico bene? Per poter realizzare un software su misura avrei bisogno di sapere nel dettaglio quali sono i suoi principali obiettivi e le sue principali aspettative.

X.: Esatto, grazie dell'appunto. Il mio obiettivo principale è quello di ottimizzare i dati; vorrei, ad esempio, avere sotto controllo le scorte di tutti i prodotti del magazzino evitando, quindi, al mio staff di contare giorno per giorno le rimanenze. Vorrei anche evitare di esaurire un prodotto senza che sia già stato riordinato. Grazie al vostro nuovo software spero di poter risparmiare tempo e denaro; per me si tratta di un vero e proprio investimento a lungo termine.

D.: Certo! Comprendo. Nella sua situazione la prima idea che mi viene in mente è proprio la creazione di un database che possa tenere conto di tutta la merce che lei possiede. In questo modo potrà registrare, modificare e cancellare ogni prodotto presente in magazzino. Ogni volta che un cliente acquisterà un prodotto le scorte potranno essere facilmente aggiornate da un addetto. In questo modo potrà sempre avere la situazione sotto controllo e non rischierà di rimanere senza un determinato prodotto.

X.: Mi piace l'idea! Se possibile mi piacerebbe aggiungere anche una sezione contenente le informazioni più importanti sulle nostre aziente fornitrici.

D.: Sì, potremmo aggiungere le aziende fornitrici nel database, proprio perché sono importanti, dato che è da loro che vi arrivano i prodotti. Vi aggiungerò una sezione dove potrete registrarle quando farete un accordo, ed eventualmente modificarle o eliminarle in base alle vostre esigenze.

X.: Mmh, bello spunto. Ora pensavo che sarebbe importante avere anche informazioni sui nostri clienti, e in particolare su cosa e quanto comprano da noi.

D.: Potremmo aggiungere anche loro nel database e implementare delle funzioni per sapere quanti prodotti ha acquistato ogni cliente. Dopotutto la chiave di un'azienda di successo sono proprio i clienti, non è così?

X.: Si! Confermo! Allora sono entusiasta del vostro progetto, mi tenga aggiornato sugli sviluppi del software. Buona giornata e... buon lavoro!

D.: Buona giornata anche a lei. Ci sentiamo presto.

1.2 Descrizione in linguaggio naturale

Il progetto consiste nella creazione di un software gestionale per la gestione dei dati di un magazzino di un'azienda:
Ciò che ci interessa maggiormente sono i seguenti aspetti:

- la gestione dei dati dei prodotti;
- la gestione dei dati delle aziende;
- la gestione dei dati dei clienti;
- la possibilità di registrare gli acquisti effettuati dai clienti.

Il database si suddividerà principalmente in tre tabelle:

- *la tabella dati*: che conterrà tutti i dati relativi ai prodotti disponibili e non disponibili in magazzino;
- *la tabella delle aziende*: che conterrà tutti i dati relativi alle aziende che collaborano con la nostra impresa;
- *la tabella dei clienti*: che conterrà tutti i dati relativi ai clienti.

1.3 Glossario dei termini

Il glossario dei termini contiene tutte le parole di non immediata comprensione alle quali verrà data una spiegazione accurata. Nella Tabella 1.1 viene mostrato il glossario dei termini relativo al nostro contesto di riferimento.

Termine	Descrizione	Sinonimi
Prodotto	Unità fisica stoccata in magazzino pronta per essere distribuita o utilizzata.	articolo, merce
Azienda	Sono intese le aziende fornitrici, delle entità esterne che forniscono prodotti. Le aziende fornitrici collaborano con l'azienda del magazzino per garantire la disponibilità di prodotti da stoccare e distribuire.	ditta, impresa, ente
Cliente	Soggetto che acquista prodotti dall'azienda gestore del magazzino. Nel contesto del magazzino il cliente può essere un rivenditore o il consumatore finale.	acquirente, compratore
Codice Prodotto	Identificativo univoco associato a ciascun prodotto stoccato in magazzino.	
Prodotti Acquistati	Quantità di articoli che un determinato cliente ha acquistato.	
Codice Azienda	Identificativo univoco assegnato a ciascuna azienda fornitrice che collabora con il magazzino. Questo codice, solitamente numerico o alfanumerico, serve a distinguere in modo preciso e immediato le diverse aziende all'interno del sistema gestionale del magazzino.	
Tipologia	Categoria utilizzata per classificare le aziende in base al settore merceologico. Nel contesto di un magazzino, la tipologia di azienda indica il tipo di beni forniti o gestiti, come ad esempio abbigliamento, calzature, ecc.	catalogazione, classificazione
Localizzazione	Posizione geografica di un'azienda, indicata attraverso l'indirizzo fisico o il paese in cui opera. Nel contesto di un magazzino, la localizzazione di un'azienda è un'informazione importante per gestire la logistica e la distribuzione delle merci, dato che influisce sui tempi di consegna, i costi di trasporto e la pianificazione delle spedizioni.	ubicazione, collocamento, collocazione, posizionamento
Numero Scorte	Quantità totale di un determinato prodotto disponibile in quel dato momento nel magazzino. Questo valore permette di fare l'inventario, ovvero di monitorare costantemente le scorte in modo che siano sempre sufficienti a soddisfare la domanda dei clienti.	

Tabella 1.1: Glossario dei termini relativo alla descrizione del sistema nel linguaggio naturale

1.4 Analisi dei requisiti

I requisiti del sistema si suddividono in due categorie: requisiti funzionali e requisiti non funzionali. I requisiti funzionali stabiliscono le azioni che il nostro sistema dovrà compiere. Invece, i requisiti non funzionali delineano i vincoli che il sistema deve soddisfare. Nella Figura 1.1 viene riportata l'analisi dei requisiti relativa al nostro software.

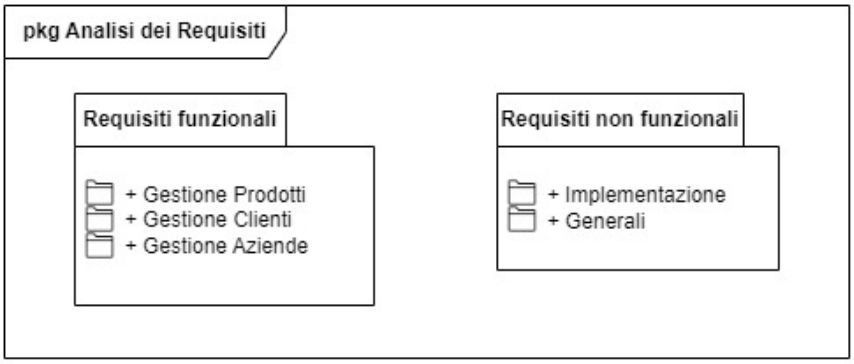

Figura 1.1: Analisi dei requisiti funzionali e non funzionali, schema generale

1.4.1 Requisiti funzionali

I requisiti funzionali descrivono le funzionalità specifiche che un sistema o software deve fornire, definendo cosa deve fare per soddisfare le esigenze degli utenti o dei processi aziendali.

Nella Figura 1.2 vengono riportati i requisiti non funzionali relativi al nostro software.

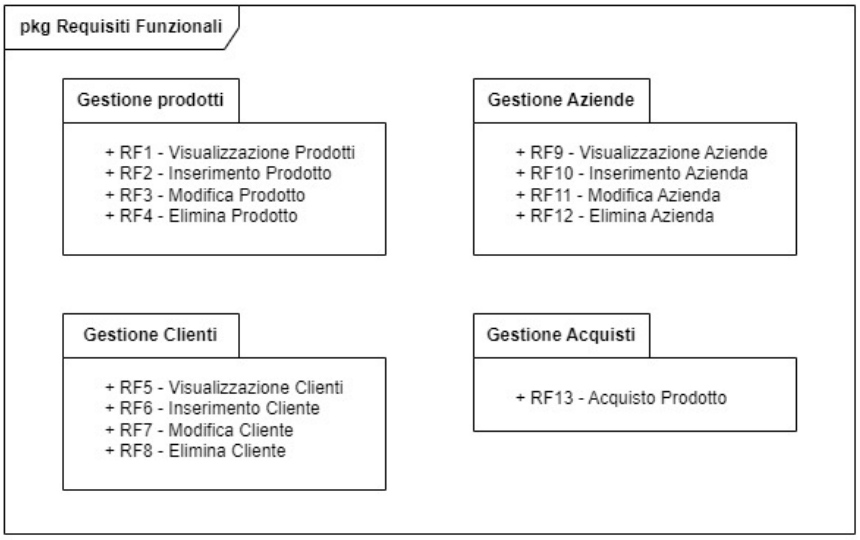

Figura 1.2: Analisi dei requisiti funzionali

Gestione Prodotti

Nella Tabella 1.2 vengono riportati i requisiti funzionali relativi alla gestione dei prodotti.

Requisito	Descrizione
RF1 - Visualizza prodotti	Il sistema dovrà consentire la visualizzazione dei prodotti
RF2 - Inserisci Prodotto	Il sistema dovrà consentire l'inserimento di un nuovo prodotto
RF3 - Modifica Prodotto	Il sistema dovrà consentire la modifica di un prodotto
RF4 - Elimina Prodotto	Il sistema dovrà consentire l'eliminazione di un prodotto

Tabella 1.2: Requisiti funzionali - Gestione Prodotti

Gestione Clienti

Nella Tabella 1.3 vengono riportati i requisiti funzionali relativi alla gestione dei clienti.

Requisito	Descrizione
RF5 - Visualizza clienti	Il sistema dovrà consentire la visualizzazione dei clienti
RF6 - Inserisci Cliente	Il sistema dovrà consentire l'inserimento di un nuovo cliente
RF7 - Modifica Cliente	Il sistema dovrà consentire la modifica di un cliente
RF8 - Elimina Cliente	Il sistema dovrà consentire l'eliminazione di un cliente

Tabella 1.3: Requisiti funzionali - Gestione Clienti

Gestione Aziende

Nella Tabella 1.4 vengono riportati i requisiti funzionali relativi alla gestione delle aziende.

Requisito	Descrizione
RF9 - Visualizza Aziende	Il sistema dovrà consentire la visualizzazione delle aziende
RF10 - Inserisci Aziende	Il sistema dovrà consentire l'inserimento di una nuova azienda
RF11 - Modifica Aziende	Il sistema dovrà consentire la modifica di un'azienda
RF12 - Elimina Aziende	Il sistema dovrà consentire l'eliminazione di un'azienda

Tabella 1.4: Requisiti funzionali - Gestione Aziende

Gestione Acquisti

Nella Tabella 1.5 vengono riportati i requisiti funzionali relativi alla gestione degli acquisti.

Requisito	Descrizione
RF13 - Acquisto Prodotto	Il sistema dovrà consentire l'acquisto di un prodotto da parte di un cliente

Tabella 1.5: Requisiti funzionali - Gestione Acquisti

1.4.2 Requisiti non funzionali

I requisiti non funzionali rappresentano i vincoli, di natura tecnologica e non, che il sistema deve garantire.

Nella figura 1.3 vengono riportati i requisiti non funzionali relativi al nostro software.

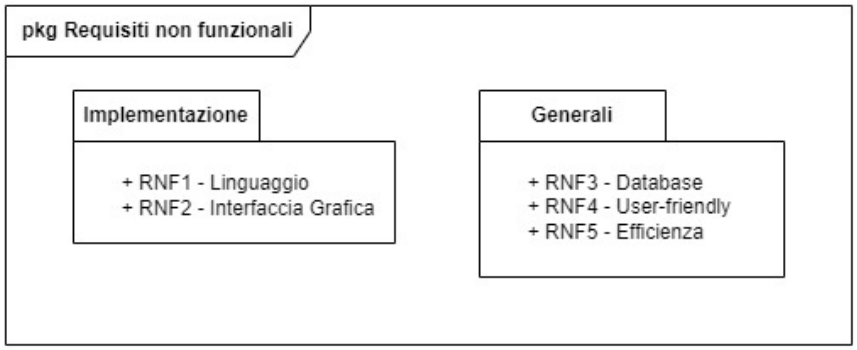

Figura 1.3: Analisi dei requisiti non funzionali

Implementazione

Nella Tabella 1.6 vengono riportati i requisiti non funzionali relativi alla gestione degli acquisti.

Requisito	Descrizione
RNF1 - Linguaggio	Il sistema dovrà essere realizzato mediante il linguaggio di programmazione Python
RNF2 - Interfaccia Grafica	Il sistema dovrà essere dotato di un'interfaccia grafica

Tabella 1.6: Requisiti non funzionali - Implementazione

Generali

Nella Tabella 1.7 vengono riportati i requisiti non funzionali di natura più generale.

Requisito	Descrizione
RNF3 - Database	Il sistema dovrà essere dotato di un database per consentire la persistenza dei dati
RNF4 - User-friendly	Il sistema dovrà essere di facile utilizzo anche per persone non esperte
RNF5 - Efficienza	Il sistema dovrà consentire l'ottimizzazione dei tempi

Tabella 1.7: Requisiti non funzionali - Generali

1.5 Diagramma dei casi d'uso

Il caso d'uso è qualcosa che un attore (entità esterna al sistema) vuole che il sistema faccia. In seguito analizzaremo i diversi casi d'uso relativi al nostro sistema.

1.5.1 Gestione dei prodotti

Nella Figura 1.4 vengono riportati i casi d'uso riguardanti la gestione dei prodotti.

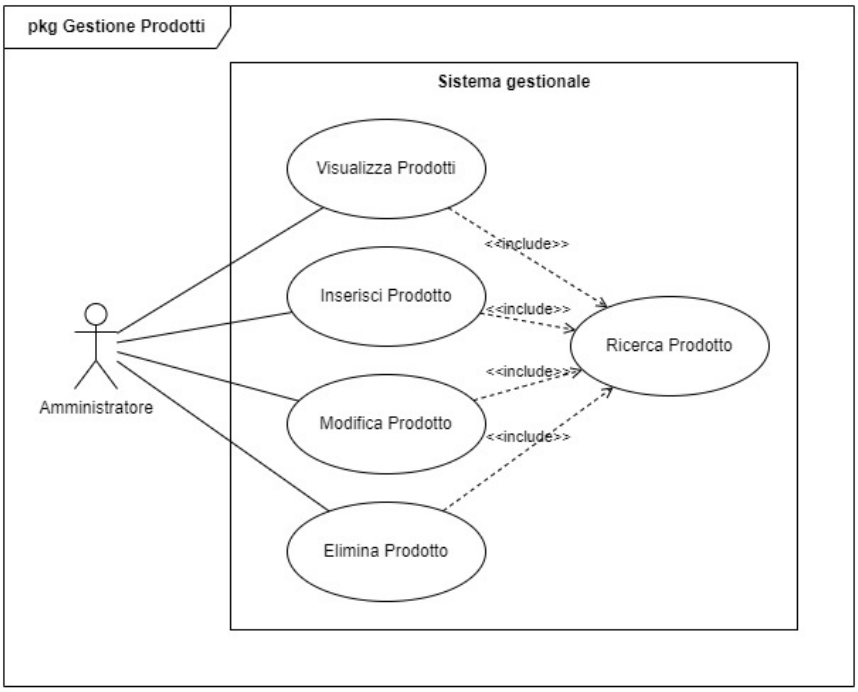

Figura 1.4: Casi d'uso relativi alla gestione dei prodotti

Visualizzazione dei prodotti

Nella Tabella 1.8 viene riportato il caso d'uso relativo alla visualizzazione dei prodotti.

Caso d'uso: Visualizzazione dei prodotti
ID: 1
Breve descrizione: L'amministratore visualizza i prodotti.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il prodotto esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole visualizzare le informazioni su un prodotto. 2. Il sistema ricerca le informazioni su quel prodotto. 3. Il sistema visualizza nello schermo le informazioni sul prodotto.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: Nessuna.

Tabella 1.8: Caso d'uso relativo alla visualizzazione dei prodotti

Inserimento di un prodotto

Nella Tabella 1.9 viene riportato il caso d'uso relativo all'inserimento di un prodotto.

Caso d'uso: Inserimento di un prodotto
ID: 2
Breve descrizione: L'amministratore inserisce un prodotto.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il prodotto non esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole inserire un nuovo prodotto. 2. L'amministratore inserisce le informazioni sul nuovo prodotto. 3. Il sistema registra il nuovo prodotto.
Postcondizioni: Il prodotto è stato inserito.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.9: Caso d'uso relativo all'inserimento di un prodotto

Modifica di un prodotto

Nella Tabella 1.10 viene riportato il caso d'uso relativo alla modifica di un prodotto.

Caso d'uso: Modifica di un prodotto
ID: 3
Breve descrizione: L'amministratore modifica un prodotto.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il prodotto esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole modificare le informazioni su un prodotto. 2. Il sistema restituisce le informazioni sul prodotto cercato dall'amministratore. 3. L'amministratore modifica le informazioni su quel prodotto. 4. Il sistema registra le informazioni sul prodotto.
Postcondizioni: Il prodotto è stato modificato.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.10: Caso d'uso relativo alla modifica di un prodotto

Eliminazione di un prodotto

Nella Tabella 1.11 viene riportato il caso d'uso relativo all'eliminazione di un prodotto.

Caso d'uso: Eliminazione di un prodotto
ID: 4
Breve descrizione: L'amministratore elimina un prodotto.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il prodotto esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole eliminare un prodotto. 2. Il sistema restituisce le informazioni sul prodotto cercato dall'amministratore. 3. L'amministratore elimina quel prodotto. 4. Il sistema elimina quel prodotto.
Postcondizioni: Il prodotto è stato eliminato.
Sequenze degli eventi alternative: Nessuna.

Tabella 1.11: Caso d'uso relativo all'eliminazione di un prodotto

Ricerca di un prodotto

Nella Tabella 1.12 viene riportato il caso d'uso relativo alla ricerca di un prodotto.

Caso d'uso: Ricerca di un prodotto
ID: 5
Breve descrizione: L'amministratore cerca un prodotto.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Nessuna.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole cercare un prodotto. 2. L'amministratore inserisce il nome del prodotto da cercare. 3. L'amministratore cerca se esiste quel prodotto. 4. Il sistema restituisce il prodotto cercato.
Postcondizioni: Nessuna
Sequenze degli eventi alternative: 4. Il sistema non restituisce niente.

Tabella 1.12: Caso d'uso relativo alla ricerca di un prodotto

1.5.2 Gestione dei clienti

Nella Figura 1.5 vengono riportati i casi d'uso riguardanti la gestione dei clienti.

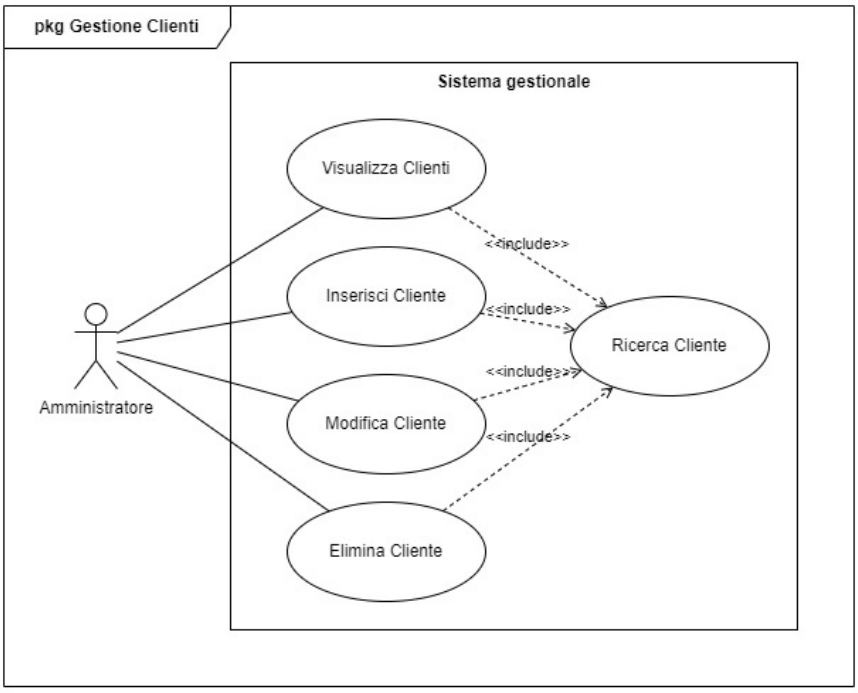

Figura 1.5: Casi d'uso relativi alla gestione dei clienti

Visualizzazione dei clienti

Nella Tabella 1.13 viene riportato il caso d'uso relativo alla visualizzazione dei clienti.

Caso d'uso: Visualizzazione dei clienti
ID: 6
Breve descrizione: L'amministratore visualizza i clienti.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il cliente esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole visualizzare le informazioni su un cliente. 2. Il sistema ricerca le informazioni su quel cliente. 3. Il sistema visualizza nello schermo le informazioni sul cliente.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: Nessuna.

Tabella 1.13: Caso d'uso relativo alla visualizzazione dei clienti

Inserimento di un cliente

Nella Tabella 1.14 viene riportato il caso d'uso relativo all'inserimento di un cliente.

Caso d'uso: Inserimento di un cliente
ID: 7
Breve descrizione: L'amministratore inserisce un cliente.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il cliente non esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole inserire un nuovo cliente. 2. L'amministratore inserisce le informazioni sul nuovo cliente. 3. Il sistema registra il nuovo cliente.
Postcondizioni: Il cliente è stato inserito.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.14: Caso d'uso relativo all'inserimento di un cliente

Modifica di un cliente

Nella Tabella 1.15 viene riportato il caso d'uso relativo alla modifica di un cliente.

Caso d'uso: Modifica di un cliente
ID: 8
Breve descrizione: L'amministratore modifica un cliente.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il cliente esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole modificare le informazioni su un cliente. 2. Il sistema restituisce le informazioni sul cliente cercato dall'amministratore. 3. L'amministratore modifica le informazioni su quel cliente. 4. Il sistema registra le informazioni sul cliente.
Postcondizioni: Il cliente è stato modificato.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.15: Caso d'uso relativo alla modifica di un cliente

Eliminazione di un cliente

Nella Tabella 1.16 viene riportato il caso d'uso relativo all'eliminazione di un cliente.

Caso d'uso: Eliminazione di un cliente
ID: 9
Breve descrizione: L'amministratore elimina un cliente.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Il prodotto esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole eliminare un cliente. 2. Il sistema restituisce le informazioni sul cliente cercato dall'amministratore. 3. L'amministratore elimina quel cliente. 4. Il sistema elimina quel cliente.
Postcondizioni: Il cliente è stato eliminato.
Sequenze degli eventi alternative: Nessuna

Tabella 1.16: Caso d'uso relativo all'eliminazione di un cliente

Ricerca di un cliente

Nella Tabella 1.17 viene riportato il caso d'uso relativo alla ricerca di un cliente.

Caso d'uso: Ricerca di un cliente
ID: 10
Breve descrizione: L'amministratore cerca un cliente.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Nessuna.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole cercare un cliente. 2. L'amministratore inserisce il nome del cliente da cercare. 3. L'amministratore verifica se esiste quel cliente. 4. Il sistema restituisce il cliente cercato.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: 4. Il sistema non restituisce niente.

Tabella 1.17: Caso d'uso relativo alla ricerca di un cliente

1.5.3 Gestione delle aziende

Nella Figura 1.6 vengono riportati i casi d'uso riguardanti la gestione delle aziende.

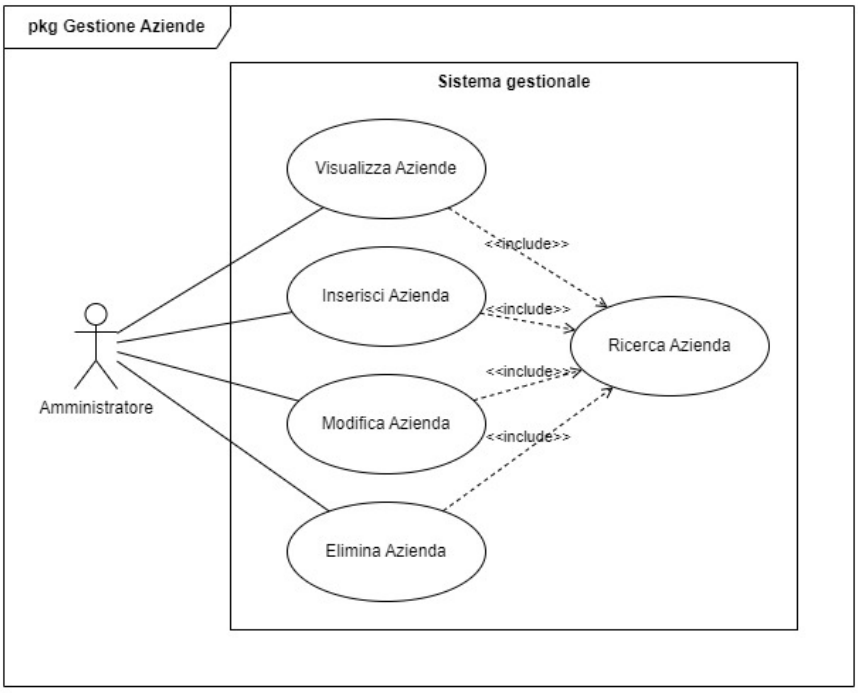

Figura 1.6: Casi d'uso relativi alla gestione delle aziende

Visualizzazione delle aziende

Nella Tabella 1.18 viene riportato il caso d'uso relativo alla visualizzazione delle aziende.

Caso d'uso: Visualizzazione delle aziende
ID: 11
Breve descrizione: L'amministratore visualizza le aziende.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: L'azienda esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole visualizzare le informazioni su un'azienda. 2. Il sistema ricerca le informazioni su quell'azienda. 3. Il sistema visualizza nello schermo le informazioni sull'azienda.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: Nessuna.

Tabella 1.18: Caso d'uso relativo alla visualizzazione delle aziende

Inserimento di un'azienda

Nella Tabella 1.19 viene riportato il caso d'uso relativo all'inserimento di un'azienda.

Caso d'uso: Inserimento di un'azienda
ID: 12
Breve descrizione: L'amministratore inserisce un'azienda.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: L'azienda non esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole inserire una nuova azienda. 2. L'amministratore inserisce le informazioni sulla nuova azienda. 3. Il sistema registra la nuova azienda.
Postcondizioni: L'azienda è stata inserita.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.19: Caso d'uso relativo all'inserimento di un'azienda

Modifica di un'azienda

Nella Tabella 1.20 viene riportato il caso d'uso relativo alla modifica di un'azienda.

Caso d'uso: Modifica di un'azienda
ID: 13
Breve descrizione: L'amministratore modifica un'azienda.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: L'azienda esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole modificare le informazioni su un'azienda. 2. Il sistema restituisce le informazioni sull'azienda cercata dall'amministratore. 3. L'amministratore modifica le informazioni su quell'azienda. 4. Il sistema registra le informazioni sull'azienda.
Postcondizioni: L'azienda è stata modificata.
Sequenze degli eventi alternative: 2. L'amministratore commette degli errori nell'inserire le informazioni. 3. Il sistema restituisce un messaggio di errore.

Tabella 1.20: Caso d'uso relativo alla modifica di un'azienda

Eliminazione di un'azienda

Nella Tabella 1.21 viene riportato il caso d'uso relativo all'eliminazione di un'azienda.

Caso d'uso: Eliminazione di un'azienda
ID: 14
Breve descrizione: L'amministratore elimina un'azienda.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: L'azienda esiste nel sistema.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole eliminare un'azienda. 2. Il sistema restituisce le informazioni sull'azienda cercata dall'amministratore. 3. L'amministratore elimina quell'azienda. 4. Il sistema elimina quell'azienda.
Postcondizioni: L'azienda è stato eliminata.
Sequenze degli eventi alternative: Nessuna

Tabella 1.21: Caso d'uso relativo all'eliminazione di un'azienda

Ricerca di un'azienda

Nella Tabella 1.22 viene riportato il caso d'uso relativo alla ricerca di u'azienda.

Caso d'uso: Ricerca di un'azienda
ID: 15
Breve descrizione: L'amministratore cerca un'azienda.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Nessuna.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole cercare un'azienda. 2. L'amministratore inserisce il nome dell'azienda da cercare. 3. L'amministratore cerca se esiste quell'azienda. 4. Il sistema restituisce l'azienda cercata.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: 4. Il sistema non restituisce niente.

Tabella 1.22: Caso d'uso relativo alla ricerca di un'azienda

1.5.4 Gestione degli acquisti

Nella Figura 1.7 vengono riportati i casi d'uso riguardanti la gestione degli acquisti.

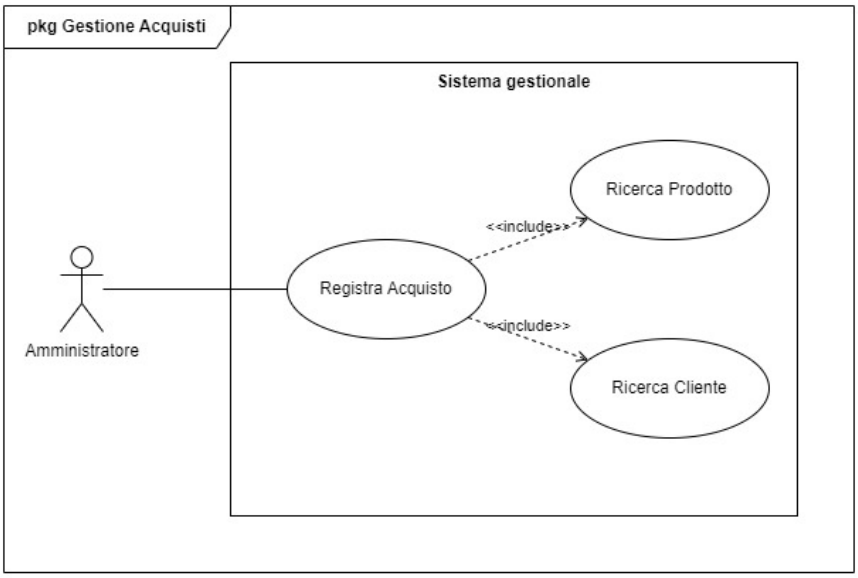

Figura 1.7: Casi d'uso relativi alla gestione degli acquisti

Registrazione di un acquisto

Nella Tabella 1.23 viene riportato il caso d'uso relativo alla registrazione di un acquisto.

Caso d'uso: Registrazione di un acquisto
ID: 16
Breve descrizione: L'amministratore registra un acquisto.
Attori primari: Amministratore.
Attori secondari: Nessuno.
Precondizioni: Nessuna.
Sequenza degli eventi principali: 1. Il caso d'uso inizia quando l'amministratore vuole registrare un acquisto. 2. L'amministratore inserisce il nome del prodotto e il nome del cliente da cercare. 3. L'amministratore cerca se esistono quel prodotto e quel cliente. 4. L'amministratore verifica quante unità di prodotto il cliente acquista. 4. Il sistema registra l'acquisto.
Postcondizioni: Nessuna.
Sequenze degli eventi alternative: 3. Il sistema non restituisce niente. 4. Le unità di prodotto immesse superano le quantità di prodotto disponibili

Tabella 1.23: Caso d'uso relativo alla registrazione di un acquisto

CAPITOLO 2

Progettazione del database

In questo capitolo parleremo della progettazione del database che verrà utilizzato per la gestione del magazzino. Progettare correttamente il database è fondamentale per garantire che i dati relativi a prodotti, clienti e aziende siano gestiti in modo efficace ed efficiente.

2.1 Progettazione concettuale

Individuati dall'intervista i problemi principali, abbiamo le informazioni principali sulle quali dovremo costruire il database. Possiamo schematizzare il processo sul quale abbiamo sviluppato il database sul seguente modo:

1. Intervista con il cliente e successivamente raccolta dei requisiti fondamentali,
2. Realizzazione di un primo schema scheletro così da rendere l'idea dell' ossatura della base di dati,
3. Raffinamento dello schema iniziale utilizzando la metodologia TOP-DOWN che ci ha permesso di raffinare le entità e le relazioni trovate precedentemente,
4. Integrazione di tutto ciò che abbiamo ottenuto nei passi precedenti così da ottenere uno schema concettuale completo e funzionale ai nostri scopi.

2.1.1 Identificazione delle entità principali

Data l'intervista, dopo l'analisi dei requisiti, abbiamo trovato le seguenti, fondamentali, entità.

- *Prodotto*: rappresenta i prodotti presenti in magazzino e acquistabili dai clienti,
- *Cliente*: rappresenta gli acquirenti che possono acquistare i prodotti,
- *Azienda*: rappresenta le aziende fornitrici da cui arrivano i prodotti.

2.1.2 Scheletro dello schema

Abbiamo unito le principali componenti in uno schema scheletro, che verrà in seguito raffinato e implementato in tutte le sue componenti.

Nella Figura 2.1 viene riportato lo schema scheletro del nostro database.

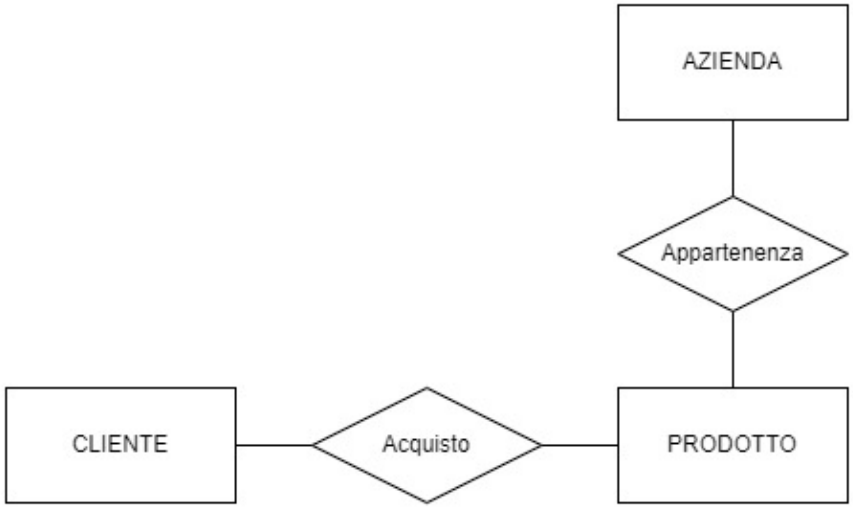

Figura 2.1: Schema scheletro del nostro database

2.1.3 Sviluppo delle componenti dello scheletro

Utilizziamo la strategia Top-Down per trattare ogni entità singolarmente, sviluppandone le caratteristiche e le componenti.

Prodotto

L'entità Prodotto riassume tutti i prodotti, i quali possono essere identificati mediante il proprio codice univoco. Oltre a questo contiene il nome del prodotto, l'azienda di appartenenza, il prezzo e il numero di scorte disponibili in magazzino.

Nella Figura 2.2 viene riportata l'entità Prodotto.

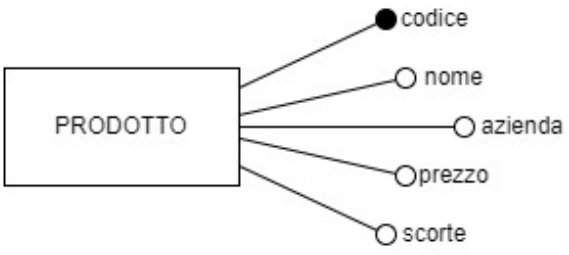

Figura 2.2: Entità Prodotto

Cliente

L'entità Cliente riassume tutti i clienti, i quali possono essere identificati tramite il proprio codice fiscale. Oltre a questo contiene il nome, il cognome, il numero di telefono e la data di nascita del cliente. Nell'entità è presente anche il numero di prodotti acquistati dal cliente.

Nella Figura 2.3 viene rappresentata l'entità Cliente.

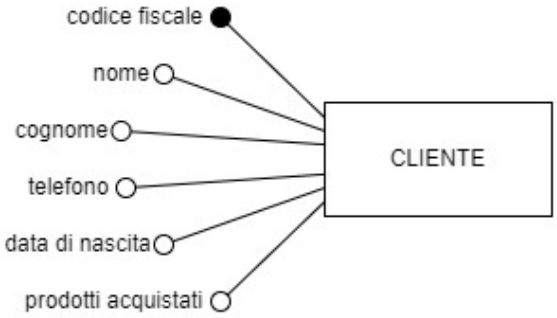

Figura 2.3: Entità Cliente

Azienda

L'entità Azienda riassume tutte le aziende fornitrici, le quali possono essere identificate tramite il proprio codice univoco. Oltre a questo contiene il nome, la tipologia, la localizzazione e la ragione sociale dell'azienda.

Nella Figura 2.4 viene rappresentata l'entità Azienda.

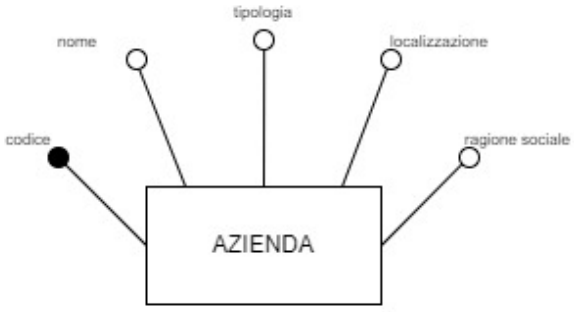

Figura 2.4: Entità Azienda

Schema finale

Nella Figura 2.5 viene rappresentato lo schema finale.

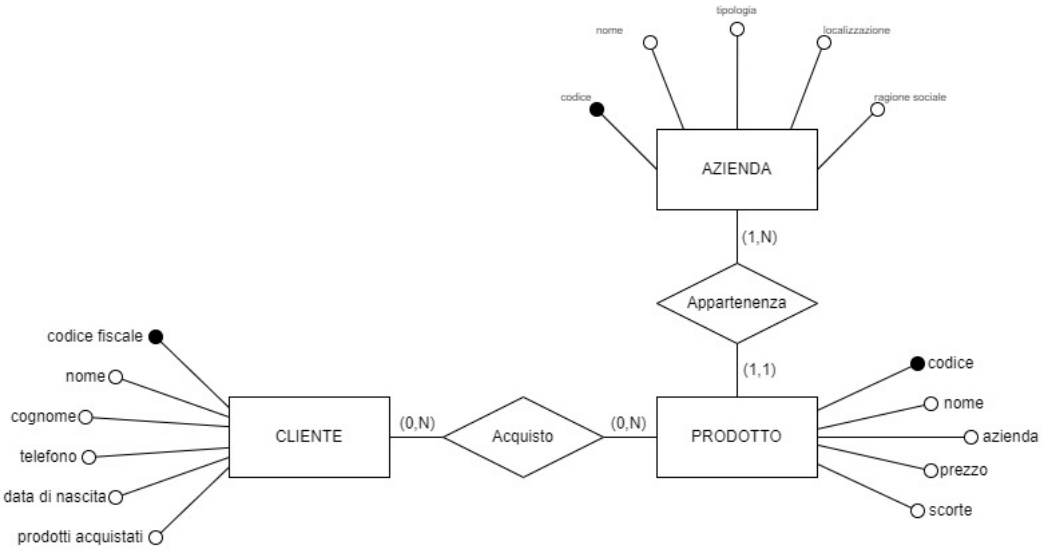

Figura 2.5: Schema finale

2.1.4 Breve analisi di qualità dello schema E-R

Verifichiamo, ora, se il nostro schema finale rispetta le caratteristiche principali di uno schema concettuale:

- *Correttezza*: non sono presenti errori sintattici o semantici, lo schema utilizza correttamente le proprie entità e relazioni.
- *Completezza*: sono stati trattati con efficacia tutti gli aspetti derivanti dall'intervista e dall'analisi dei requisiti.
- *Leggibilità*: non ci sono intrecci e lo schema appare ordinato e facilmente leggibile.
- *Minimalità*: lo schema finale risulta minimale, poiché non presenta cicli o ridondanze.

2.1.5 Dizionario dei dati

Dizionario dei dati relativo alle entità

Nella Tabella 2.1 viene riportato il dizionario dei dati relativo alle entità.

Concetto	Descrizione	Attributi	Identificatori
Prodotto	Bene fisico stoccato nel magazzino	Codice (stringa), nome (stringa), azienda (stringa), prezzo (numerico), scorte (numerico)	Codice
Azienda	Organizzazione esterna che fornisce prodotti al magazzino	Codice (numerico), nome (stringa), tipologia (stringa), localizzazione (stringa), ragione sociale (stringa)	Codice
Cliente	Persona che acquista prodotti dal magazzino	Codice fiscale (stringa), nome (stringa), cognome (stringa), telefono (numerico), data di nasicta(data), prodotti acquistati (numerico)	Codice fiscale

Tabella 2.1: Dizionario dei dati relativo alle entità presenti nello schema finale

Dizionario dei dati relativo alle relazioni

Nella Tabella 2.2 viene riportato il dizionario dei dati relativo alle relazioni.

Relazione	Descrizione	Entità coinvolte	Attributi
Appartenenza	Lega il prodotto e l'azienda fornitrice che lo fornisce al magazzino	Prodotto, Azienda	-
Acquisto	Lega il cliente con i prodotti acquistati dal magazzino	Prodotto, Cliente	-

Tabella 2.2: Dizionario dei dati relativo alle relazioni presenti nello schema finale

2.1.6 Regole aziendali

Regole di vincolo

RV1 Il valore di "prezzo" relativo all'entità "Prodotto" deve essere maggiore di 0.
RV2 Il valore di "scorte" relativo all'entità "Prodotto" deve essere maggiore o uguale a 0.
RV3 Il valore di "prodotti acquistati" relativo all'entità "Cliente" deve essere maggiore o uguale a 0.

Regole di derivazione

Non è presente nessuna regola di derivazione perchè lo schema E-R disegnato e descritto è minimale; quindi, nessun attributo potrà essere derivato da altri.

2.2 Progettazione logica

La *progettazione logica* è la fase del processo di sviluppo di un database in cui si traduce il modello concettuale (ad esempio, il diagramma E-R) in un modello logico che può essere implementato su un sistema di gestione di database. Durante questa fase, si definiscono le tabelle, gli attributi, le chiavi primarie e le chiavi esterne, oltre alle relazioni tra le tabelle, senza ancora preoccuparsi delle specificità del database fisico o delle tecnologie utilizzate. L'obiettivo è creare una struttura organizzata e coerente dei dati che garantisca integrità e coerenza nelle operazioni.

2.2.1 Tavola dei volumi

Nella Tabella 2.3 viene riportata la tavola dei volumi.

Concetto	Tipo	Volume
Prodotto	E	2000
Azienda	E	25
Cliente	E	400
Appartenenza	R	2000
Acquisto	R	750

Tabella 2.3: Tavola dei volumi

2.2.2 Tavola delle operazioni

Nella Tabella 2.4 viene riportata la tavola delle operazioni con frequenza mensile.

Operazione	Frequenza
Visualizza prodotti	5000
Inserisci prodotto	200
Modifica prodotto	50
Elimina prodotto	30
Visualizza clienti	800
Inserisci cliente	50
Modifica cliente	5
Elimina cliente	20
Visualizza aziende	100
Inserisci azienda	4
Modifica azienda	1
Elimina azienda	2
Registra acquisto	120

Tabella 2.4: Tavola delle operazioni

2.2.3 Elenco degli identificatori principali

Nella Tabella 2.5 viene riportato l'elenco degli identificatori principali.

Nome Entità	Identificatore
Prodotto	Codice
Cliente	Codice Fiscale
Azienda	Codice

Tabella 2.5: Elenco degli identificatori principali

2.2.4 Normalizzazione

Nella Tabella 2.6 viene riportata la tabella di normalizzazione.

Nome Entità	Commento
Prodotto	Non esistono dipendenze banali fra gli attributi
Cliente	Non esistono dipendenze banali fra gli attributi
Azienda	Non esistono dipendenze banali fra gli attributi

Tabella 2.6: Normalizzazione

2.2.5 Traduzione verso il modello relazionale

Nella Tabella 2.7 viene riportata la tabella di traduzione verso il modello relazionale.

Nome Entità	Commento
Prodotto	Prodotto (**codice**, nome, azienda, prezzo, scorte)
Cliente	Cliente (**codiceFiscale**, nome, cognome, telefono, dataDiNascita, prodottiAcquistati)
Azienda	Azienda (**codice**, nome, tipologia, localizzazione, ragioneSociale)

Tabella 2.7: Traduzione verso il modello relazionale

CAPITOLO 3

Progettazione della componente applicativa

In questi capitolo parleremo della progettazione della componente applicativa. L'obiettivo della componente applicativa è fornire un'interfaccia semplice e intuitiva, capace di gestire le operazioni principali del magazzino. Attraverso un'architettura client-server e una chiara modellazione delle interazioni e dei processi, il sistema garantisce efficienza e flessibilità. Le sezioni successive descrivono in dettaglio le scelte progettuali, partendo dalla struttura dell'architettura del sistema, fino alla definizione delle funzionalità principali attraverso i vari diagrammi.

3.1 Architettura del sistema

L'architettura client-server è un modello in cui due entità principali (il client e il server) comunicano tra loro per svolgere determinate operazioni. Il motivo dell'utilizzo di un'architettura client-server è principalmente per separare la gestione della presentazione e interazione (client) dalla logica e dal controllo dei dati (server), migliorando modularità, scalabilità e sicurezza del sistema. Più nello specifico i motivi sono i seguenti:

- *Separazione delle responsabilità*: il client si occupa solo della presentazione e dell'interazione con l'utente. Il server gestisce la logica di business e l'accesso al database. Questa separazione rende il sistema più modulare e manutenibile.
- *Scalabilità*: in futuro, potremmo avere più client che accedono al tuo sistema (ad esempio, più operatori del magazzino o utenti da località diverse). Il server può servire più client contemporaneamente senza che ciascuno debba replicare la logica di business.
- *Sicurezza e controllo*: la logica di business risiede sul server, il che significa che i dati sensibili (come le scorte o gli ordini) non sono esposti direttamente al client. Solo il server ha accesso diretto al database, riducendo i rischi di corruzione dei dati.

In questa architettura:

- *Il client* è l'interfaccia con cui interagisce l'utente, ovvero la parte che richiede informazioni e invia comandi al server. Nel nostro caso è l'interfaccia grafica (GUI), costruita con PyQt, che consente agli operatori del magazzino di gestire prodotti, ordini e scorte.
- *Il server* gestisce la logica di business e il database. Quando il client invia una richiesta (ad esempio, per visualizzare i prodotti), il server elabora la richiesta, accede al

database e restituisce le informazioni richieste al client. Il server è implementato da un'applicazione Python che gestisce tutte le operazioni legate alla logica di magazzino e all'accesso al database.

- *La comunicazione tra client e server* prevede che il client invii una richiesta al server (ad esempio, per creare un nuovo ordine o visualizzare i prodotti).

Il server riceve la richiesta, la elabora eseguendo la logica di business (che può includere operazioni sul database), e restituisce una risposta al client. Quest'ultima contiene i dati richiesti o una conferma dell'operazione (come l'aggiornamento delle scorte o la creazione di un ordine).

3.2 Diagramma delle classi

Il diagramma delle classi ci permette di vedere le possibili interazioni tra le varie classi. Nella Figura 3.1 viene riportato il diagramma delle classi del sistema.

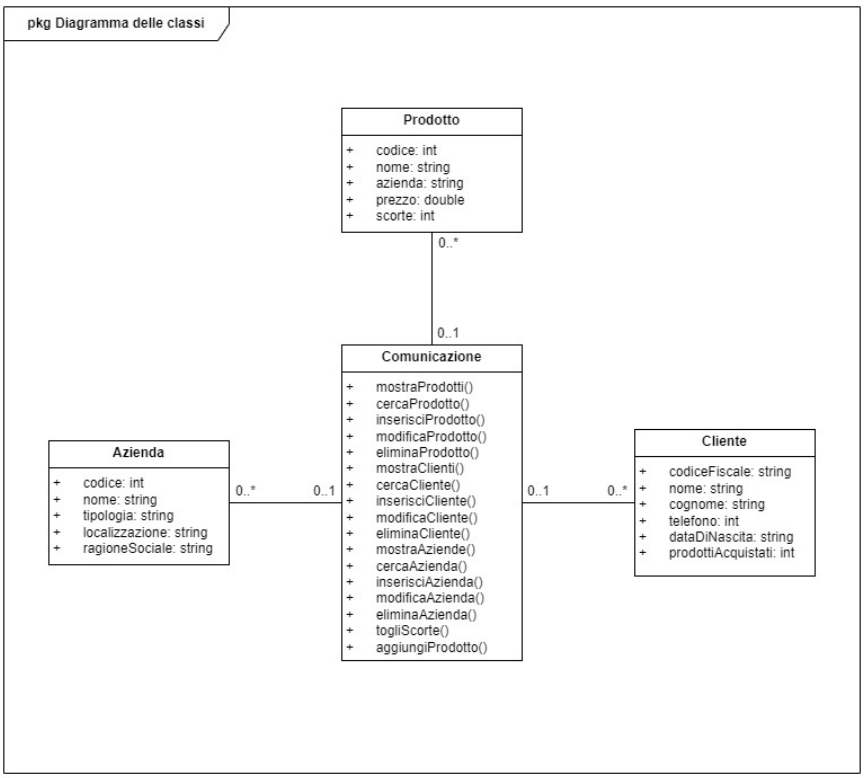

Figura 3.1: Diagramma delle classi

Nelle prossime sottosezioni illustreremo più nello specifico le varie classi che compongono il diagramma.

3.2.1 Prodotto

Nella Figura 3.2 viene riportata la classe `Prodotto`, che rappresenta un'entità con i relativi attributi. Inoltre, la classe `Prodotto` ha una relazione con la classe `Comunicazione`, che gestisce le operazioni relative ai prodotti nel database. La cardinalità indica che un'istanza della classe `Prodotto` può avere molteplici associazioni con la classe `Comunicazione`, ma quest'ultima può essere associata a un solo prodotto alla volta (0..1).

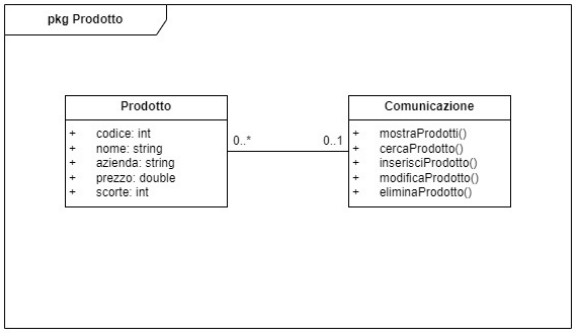

Figura 3.2: Diagramma della classe `Prodotto`

3.2.2 Cliente

Nella Figura 3.3 viene riportata la classe `Cliente`, che rappresenta un'entità con i relativi attributi. Inoltre, la classe `Cliente` ha una relazione con la classe `Comunicazione`, che gestisce le operazioni relative ai clienti nel database. La cardinalità indica che un'istanza della classe `Cliente` può avere molteplici associazioni con la classe `Comunicazione`, ma quest'ultima può essere associata a un solo cliente alla volta (0..1).

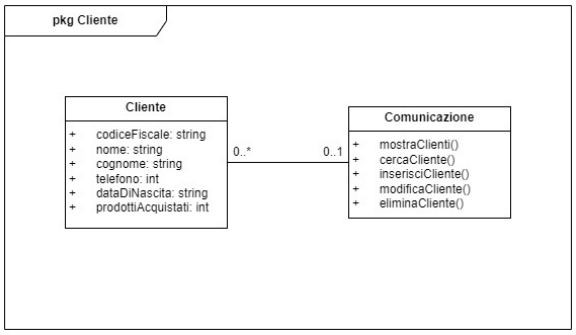

Figura 3.3: Diagramma della classe `Cliente`

3.2.3 Azienda

Nella Figura 3.4 viene riportata la classe `Azienda`, che rappresenta un'entità con i relativi attributi. Inoltre, la classe `Azienda` ha una relazione con la classe `Comunicazione`, che gestisce le operazioni relative alle aziende nel database. La cardinalità indica che un'istanza della classe `Azienda` può avere molteplici associazioni con la classe `Comunicazione`, ma quest'ultima può essere associata a una sola azienda alla volta (0..1).

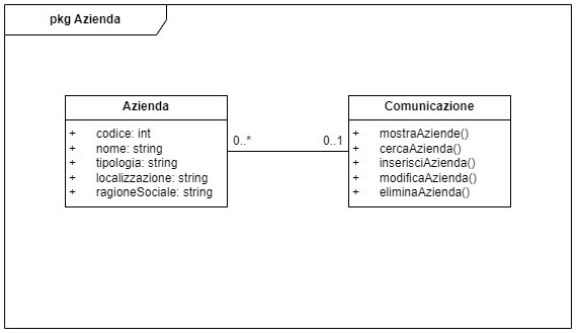

Figura 3.4: Diagramma della classe `Azienda`

3.2.4 Acquisto

Nella Figura 3.5 viene riportata la classe `Acquisto`, che rappresenta il processo di acquisto. Le classi principali coinvolte nel processo sono `Prodotto`, `Cliente` e `Comunicazione`. Le relazioni tra le classi indicano che: un cliente può essere associato a più prodotti (0..*), rappresentando i prodotti acquistati; un prodotto può essere associato a più clienti (0..*), rappresentando il fatto che un prodotto può essere acquistato da diversi clienti; la classe `Comunicazione` funge da mediatore, permettendo operazioni come la ricerca di prodotti o clienti, e la modifica delle scorte.

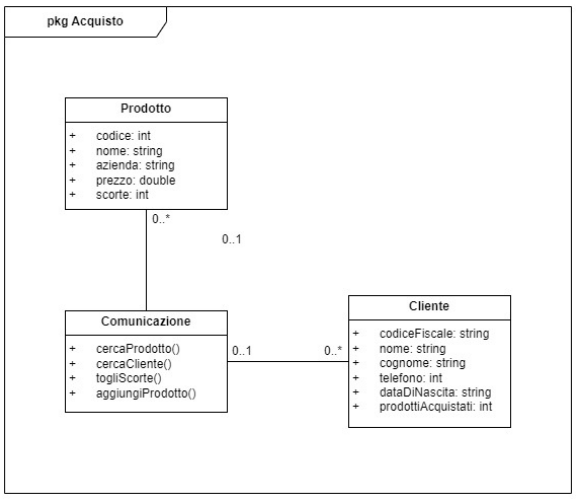

Figura 3.5: Diagramma della classe `Acquisto`

3.3 Diagrammi di sequenza

I diagrammi di sequenza sono strumenti fondamentali per rappresentare il flusso di interazioni tra gli oggetti e i componenti di un sistema. Questi diagrammi mostrano in modo sequenziale come i diversi elementi del sistema, come attori, classi e componenti software, collaborano tra loro per eseguire specifiche operazioni o processi.

Nel contesto del sistema di gestione del magazzino, i diagrammi di sequenza forniranno una chiara rappresentazione grafica di come le operazioni principali vengono eseguite. Nelle prossime sottosezioni presentiamo alcuni diagrammi di sequenza relativi al nostro sistema.

3.3.1 Gestione dell'acquisto

Nella Figura 3.6 viene riportato il diagramma di sequenza relativo alla gestione degli acquisti, in cui, specificando il prodotto e il cliente, è possibile effettuare un acquisto.

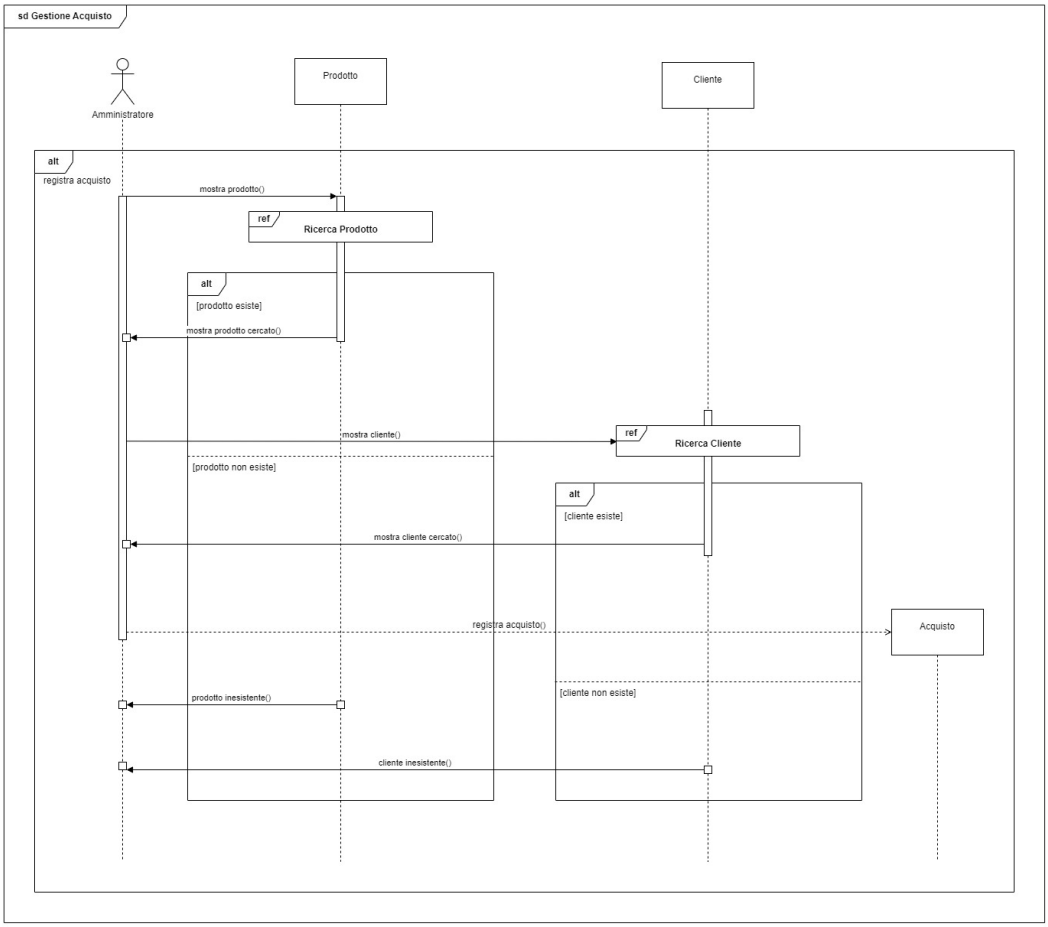

Figura 3.6: Diagramma di sequenza relativo alla gestione degli acquisti

3.3.2 Gestione dei prodotti

Nella Figura 3.7 viene riportato il diagramma di sequenza relativo alla gestione dei prodotti, in cui è possibile visualizzare, inserire, modificare o eliminare un prodotto.

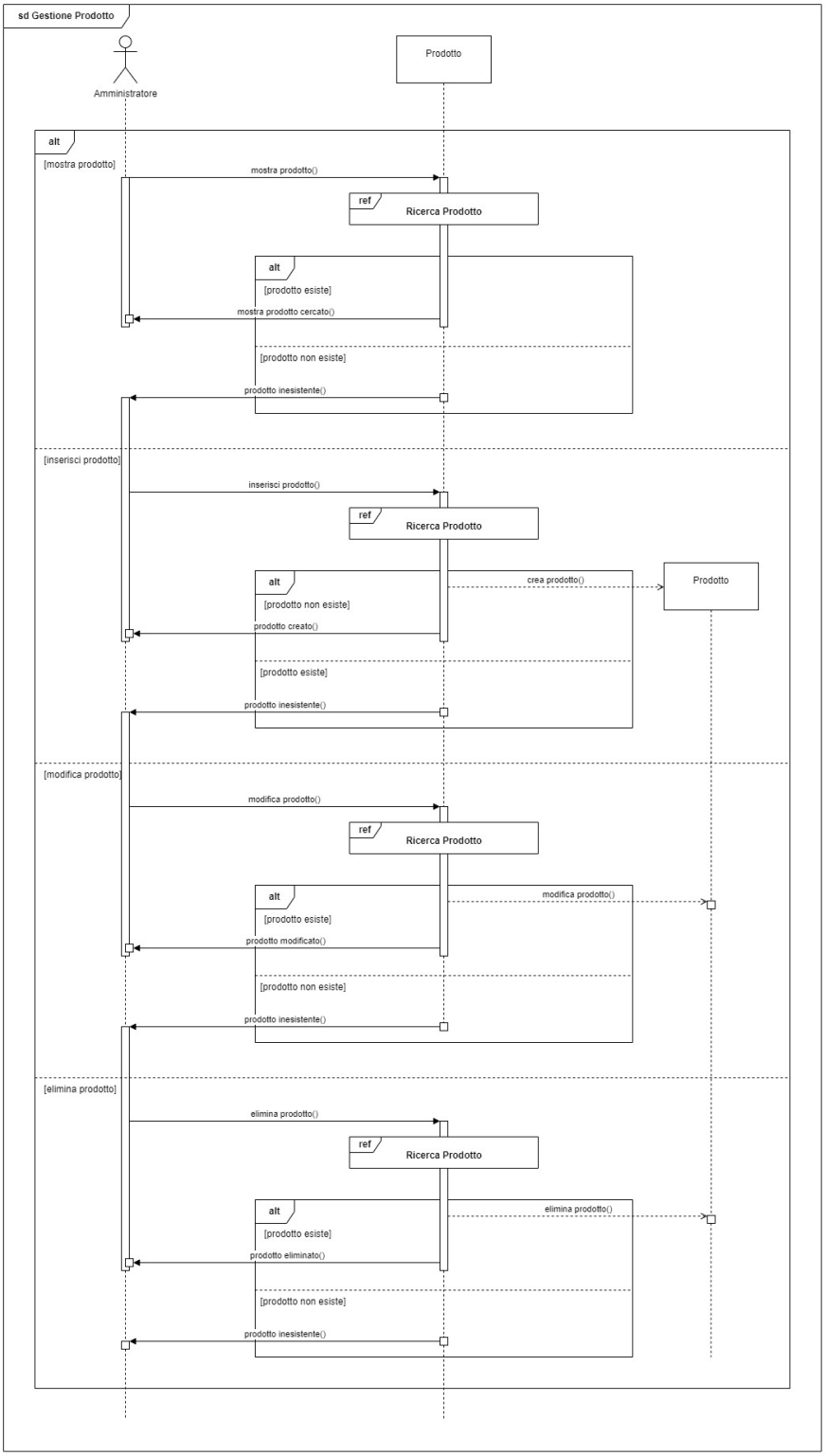

Figura 3.7: Diagramma di sequenza relativo alla gestione dei prodotti

3.3.3 Gestione del cliente

Nella Figura 3.8 viene riportato il diagramma di sequenza relativo alla gestione dei clienti, in cui è possibile visualizzare, inserire, modificare o eliminare un cliente.

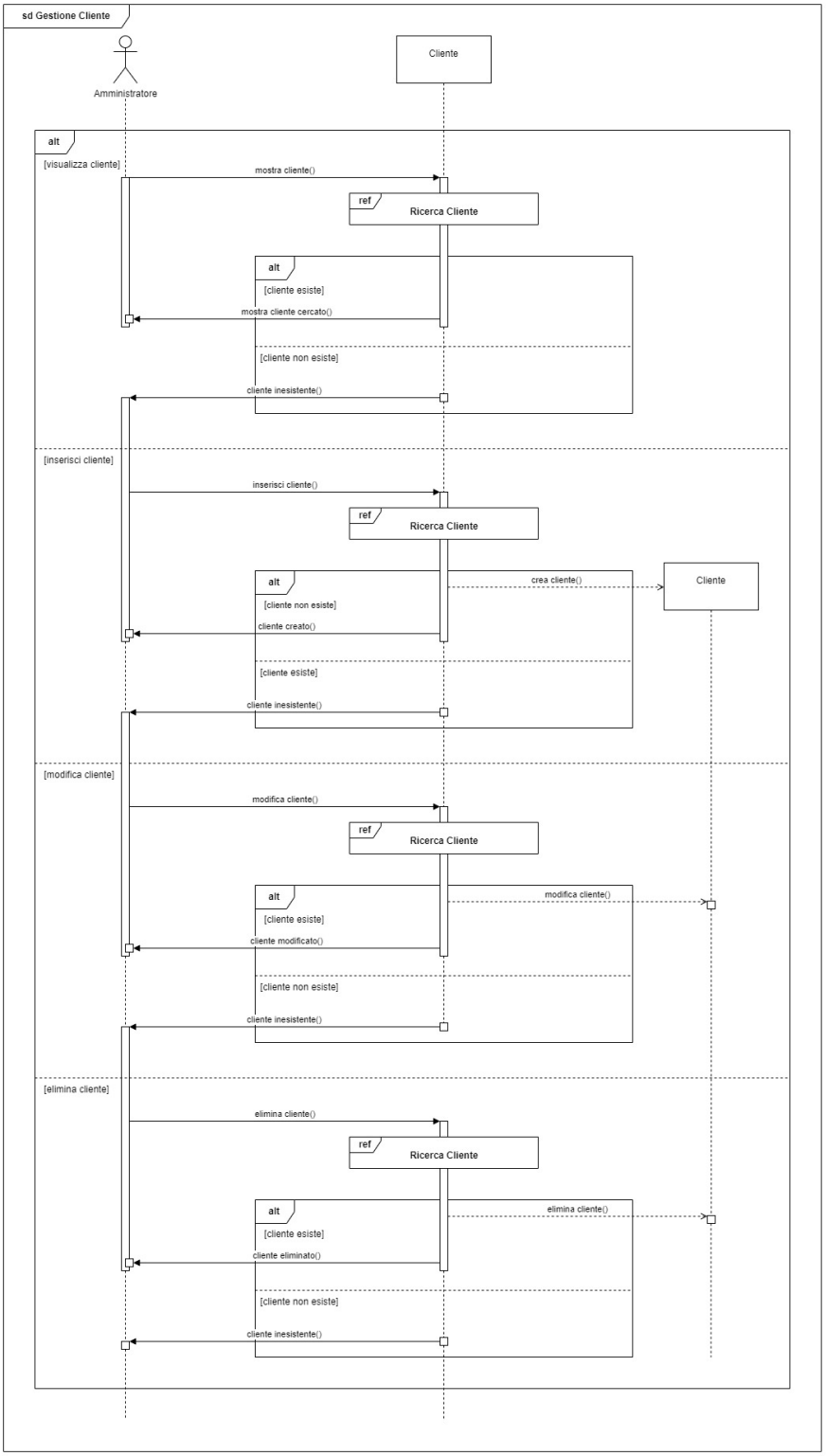

Figura 3.8: Diagramma di sequenza relativo alla gestione dei clienti

3.3.4 Gestione dell'azienda

Nella Figura 3.9 viene riportato il diagramma di sequenza relativo alla gestione delle aziende, in cui è possibile visualizzare, inserire, modificare o eliminare un'azienda.

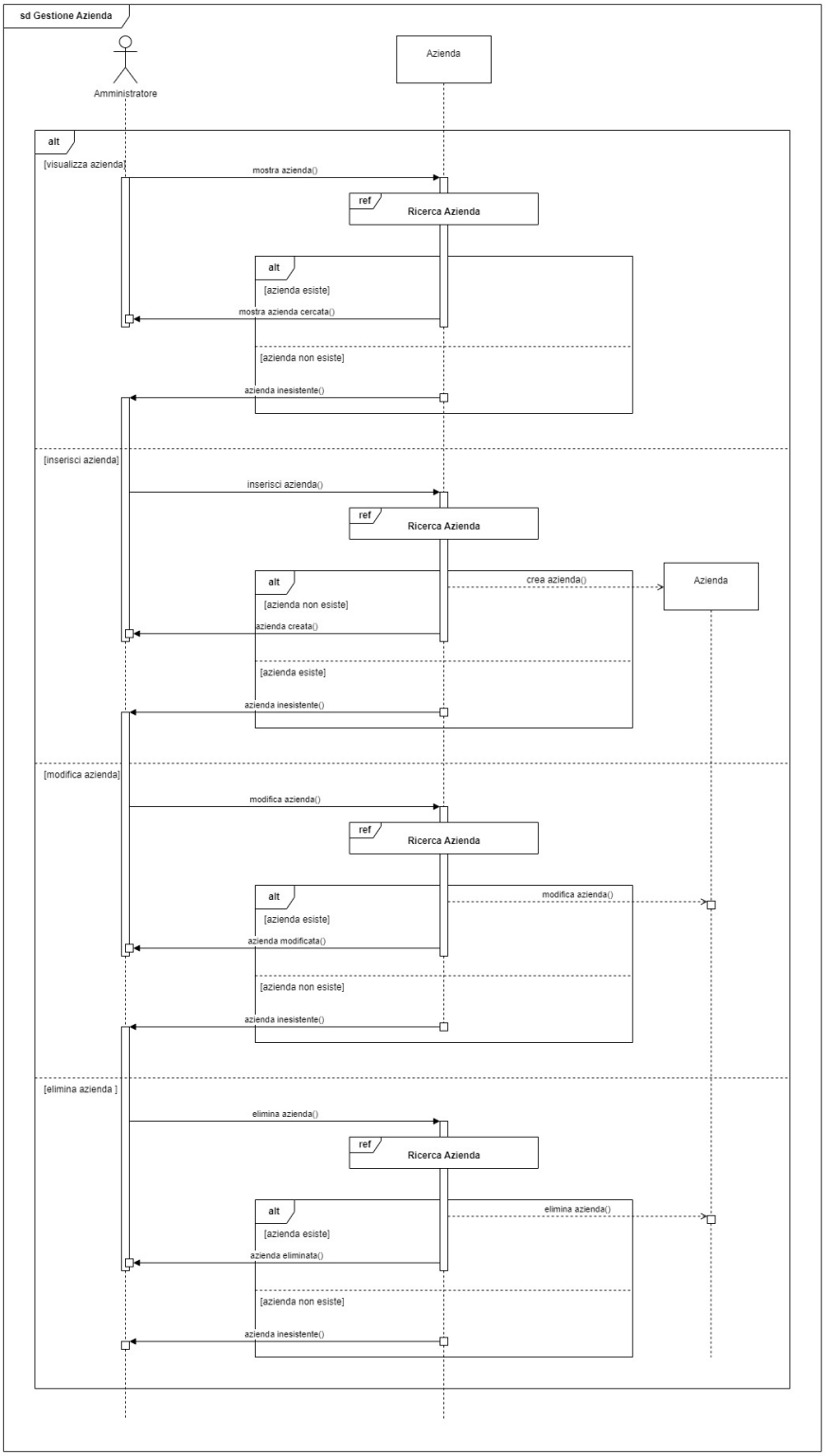

Figura 3.9: Diagramma di sequenza relativo alla gestione delle aziende

3.4 Diagramma di attività

I diagrammi di attività permettono di descrivere le varie fasi di un processo, mettendo in evidenza l'ordine delle operazioni e delle decisioni che guidano il flusso di esecuzione. Ogni diagramma rappresenta il percorso che un processo segue dall'inizio alla fine, mostrando come le attività si collegano tra loro.

Nelle prossime sottosezioni presentiamo alcuni diagrammi di attività relativi al nostro sistema.

3.4.1 Gestione dei prodotti

Nella Figura 3.10 viene riportato il diagramma di attività relativo alla gestione dei prodotti. Il diagramma descrive chiaramente il ciclo completo di gestione dei prodotti, evidenziando le diverse operazioni disponibili e gestendo le condizioni di errore.

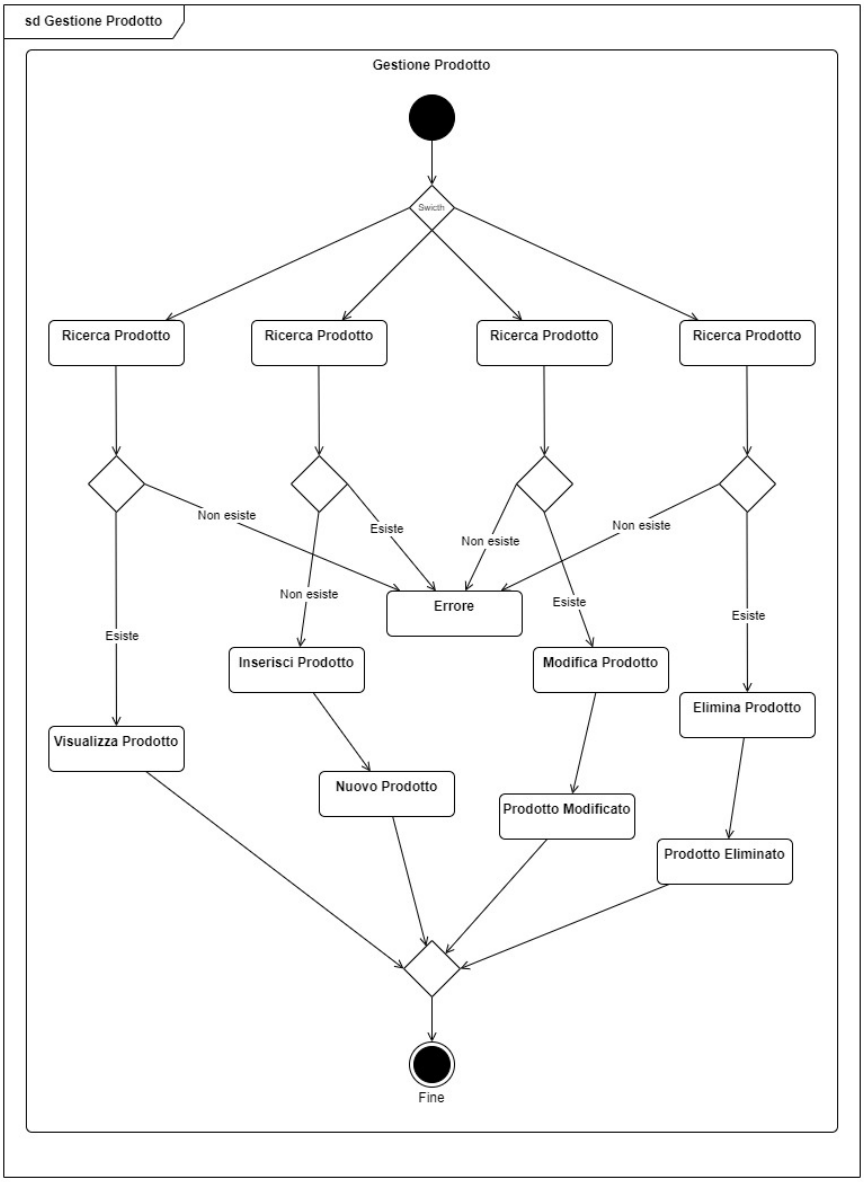

Figura 3.10: Diagramma di attività relativi alla gestione dei prodotti

3.4.2 Gestione dei clienti

Nella Figura 3.11 viene riportato il diagramma di attività relativo alla gestione dei clienti. Il diagramma descrive chiaramente il ciclo completo di gestione dei clienti, evidenziando le diverse operazioni disponibili e gestendo le condizioni di errore.

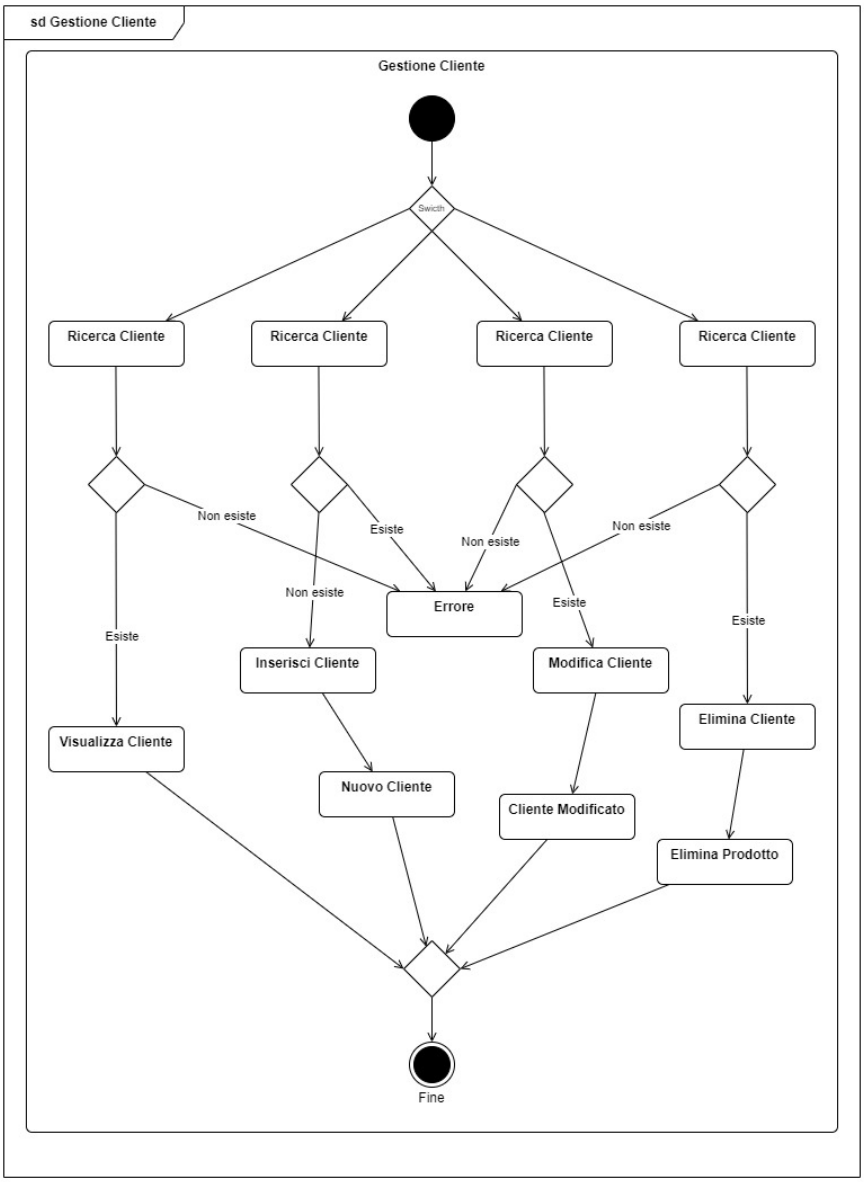

Figura 3.11: Diagramma di attività relativi alla gestione dei clienti

3.4.3 Gestione delle aziende

Nella Figura 3.12 viene riportato il diagramma di attività relativo alla gestione delle aziende. Il diagramma descrive chiaramente il ciclo completo di gestione delle aziende, evidenziando le diverse operazioni disponibili e gestendo le condizioni di errore.

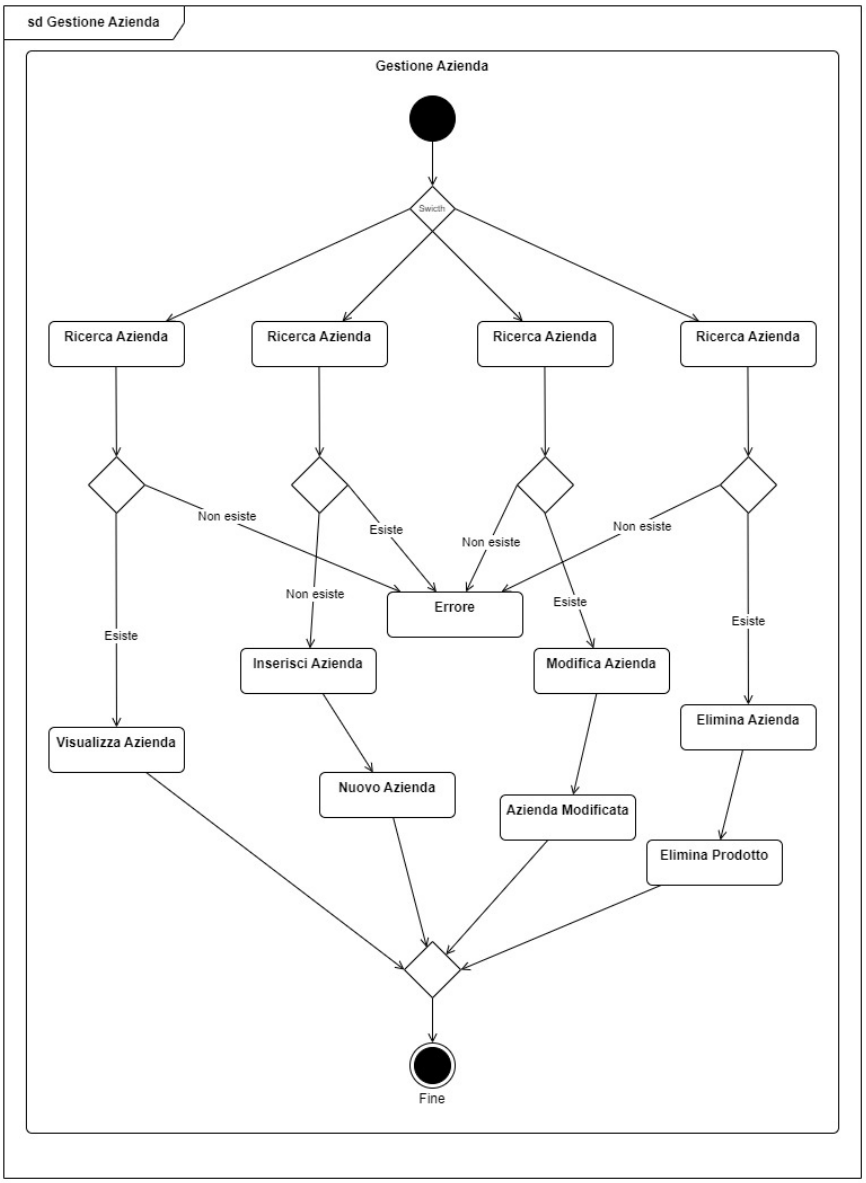

Figura 3.12: Diagramma di attività relativi alla gestione delle aziende

3.4.4 Gestione degli acquisti

Nella Figura 3.13 viene riportato il diagramma di attività relativo alla gestione degli acquisti. Il diagramma descrive chiaramente il ciclo completo di gestione degli acquisti, evidenziando le diverse operazioni disponibili e gestendo le condizioni di errore.

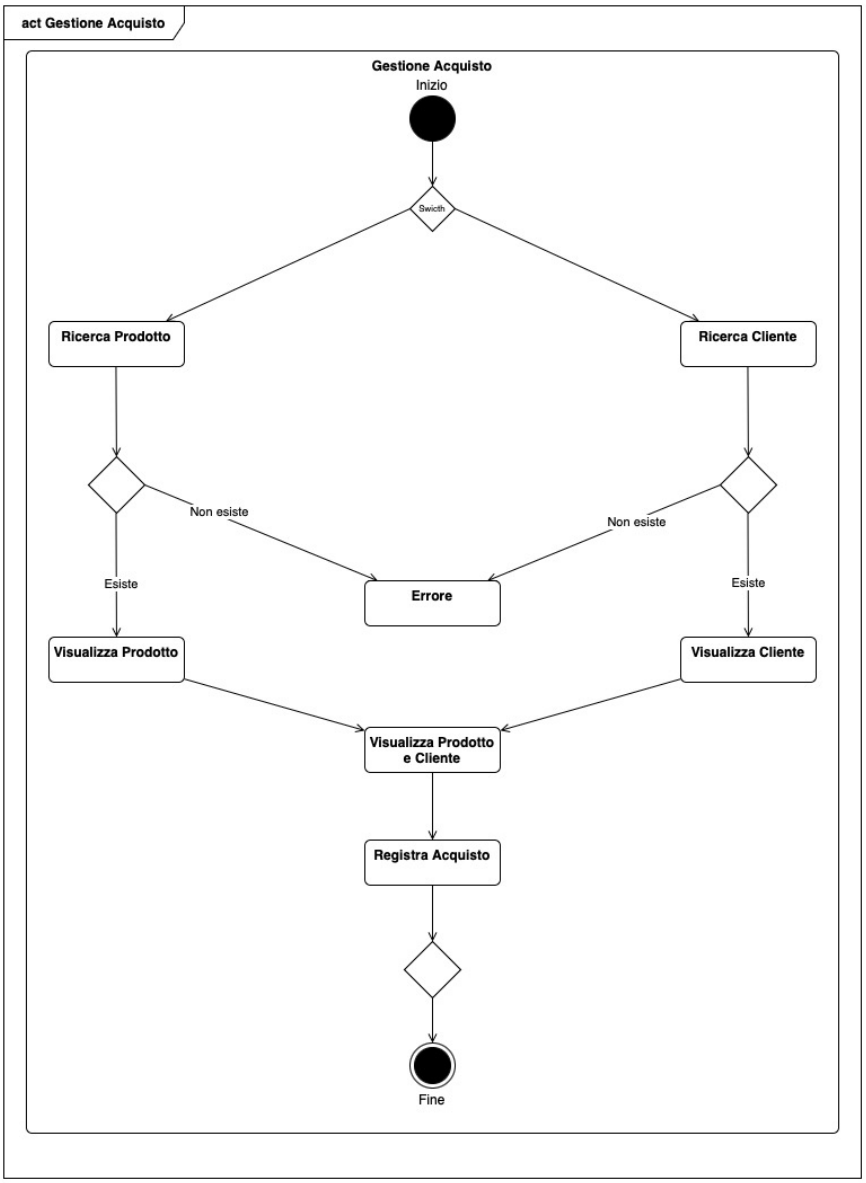

Figura 3.13: Diagramma di attività relativi alla gestione degli acquisti

3.5 Progettazione delle funzioni principali

La progettazione delle funzioni principali del sistema di gestione del magazzino è stata guidata dai requisiti funzionali e dagli obiettivi di ottimizzazione delle operazioni. Ogni funzione è stata progettata per integrarsi in modo efficiente con le altre componenti del sistema, garantendo un flusso di lavoro fluido e privo di errori. Le funzionalità principali del sistema includono la gestione dei prodotti, dei clienti, delle aziende e degli acquisti. Ciascuna di queste funzionalità viene gestita attraverso specifiche classi e flussi operativi. In particolare:

- *La gestione dei prodotti* consente di aggiungere, modificare, rimuovere e monitorare i prodotti all'interno del magazzino, tenendo traccia di tutte le informazioni rilevanti, come quantità, descrizione e prezzo.
- *La gestione dei clienti* permette di registrare, aggiornare e mantenere un database di clienti, facilitando la gestione delle informazioni per ogni cliente, come nome, contatti e la quantità dei prodotti acquistati.
- *La gestione delle aziende* è dedicata alla gestione delle aziende fornitrici, includendo le loro specifiche, tra le quali: tipologia, localizzazione e ragione sociale.
- *La gestione degli acquisti* consente di seguire l'intero processo di acquisto, verificando che siano presenti in magazzino le scorte richieste nell'acquisto e che il cliente sia presente nel database.

CAPITOLO 4

Implementazione e Testing

In questo capitolo parleremo delle fasi di implementazione e testing del nostro software, analizzando e commentando il codice delle parti più importanti del nostro sistema. Inizieremo con l'implementazione del front-end, descrivendo come è stata creata l'interfaccia utente utilizzando Qt Designer. In seguito analizzeremo l'implementazione del back-end con l'utilizzo della libreria PyQt. Inoltre, verrà anche analizzato il database utilizzato dal nostro sistema, fondamentale per la gestione dei dati; in particolare, verranno specificati nel dettaglio la sua struttura, il suo utilizzo e la sua relativa implementazione nel nostro progetto. Infine verranno riportati dei test effettuati con pytest per garantire la correttezza oltre che l'affidabilità del sistema.

4.1 Implementazione del Front-End

Lo sviluppo dell'interfaccia grafica è stata interamente effettuata su QT Designer, il quale ha generato il file "disegno.ui" dal quale vengono le schermate con cui il sistema interagisce con l'utente.

4.1.1 QT Designer

L'implementazione del front-end con Qt Designer rappresenta una fase cruciale nella creazione di un'interfaccia utente grafica (GUI) intuitiva e funzionale per il sistema di gestione del magazzino. Qt Designer è uno strumento visuale per progettare interfacce basate su widget, che facilita la creazione di layout complessi senza la necessità di scrivere codice manualmente per definire la struttura visiva. L'interfaccia è stata progettata utilizzando Qt Designer in combinazione con PyQt, un binding Python per il framework Qt, che consente di integrare facilmente il codice Python con il design visivo creato.

L'utilizzo di QT Designer e PyQt si può riassumere in 3 principali processi di implementazione:

- *Creazione del layout principale*: il layout del sistema è stato progettato per essere il più intuitivo possibile, posizionando i pulsanti per la gestione dei prodotti, dei clienti, delle aziende e degli acquisti in modo che siano facilmente accessibili. Sono stati utilizzati widget come pulsanti (QPushButton), tabelle (QTableWidget) e campi di testo (QLineEdit) per garantire un'interazione chiara e diretta con l'utente.

- *Esportazione e collegamento*: una volta creato il design, i file `.ui` sono stati esportati e convertiti in codice Python tramite il comando `pyuic5` o caricati dinamicamente utilizzando `uic.loadUi`. Questo processo garantisce che l'interfaccia progettata possa essere gestita dal codice Python, permettendo l'integrazione con la logica del sistema.
- *Eventi e connessioni*: i vari componenti dell'interfaccia, come pulsanti e campi di testo, sono stati collegati a funzioni specifiche all'interno del codice Python tramite i segnali e slot di PyQt. Ad esempio, il pulsante "Inserisci Prodotto" è stato collegato alla funzione che gestisce l'inserimento dei nuovi prodotti nel database.

Sebbene Qt Designer offra una vasta gamma di opzioni di personalizzazione, alcune parti dell'interfaccia sono state ulteriormente migliorate direttamente nel codice Python per aggiungere comportamenti personalizzati o dinamici. Ad esempio, la visualizzazione dei prodotti nel magazzino utilizza un QTableWidget, che viene popolato dinamicamente dal database in base alle azioni eseguite dall'utente.

4.2 Implementazione Back-End

Il nostro sistema è stato implementato utilizzando il linguaggio Python 3.

4.2.1 Visual Studio Code

Si è deciso di utilizzare l'IDE Visual Studio Code per sviluppare il sistema. Attraverso l'utilizzo dell'estensione `.py` e di tutte le estensioni che permette di usare l'IDE, è stato possibile utilizzarlo al meglio, potendo fare uso anche delle sue librerie.

4.2.2 PyQt

PyQt è una delle librerie più utilizzate per lo sviluppo di GUI in Python; per l'implementazione del nostro sistema abbiamo usato PyQt5. Quest'ultimo fornisce un insieme di moduli e classi per gestire la logica della GUI e implementare più funzionalità. Nel nostro sistema i moduli e le classi più utilizzati sono stati:

- `QtCore`: contiene delle classi di base di PyQt ed è stata utilizzata principalmente per le animazioni della finestra.
- `QtGui`: è un modulo che contiene classi dedicate alla gestione di immagini ed elementi grafici. Tra le sue classi abbiamo usato, in particolare:
  - `QPixmap`: è una classe che rappresenta un'immagine ottimizzata per la visualizzazione ed è stata utilizzata per i pulsanti volti a chiudere, ingrandire, rimpicciolire e minimizzare.
  - `QIcon`: è una classe che rappresente un'icona attraverso la quale possiamo caricare un'immagine, nel nostro caso le miniature all'interno dei pulsanti nella barra laterale (in formato SVG).
- `QtWidgets`: è uno dei moduli principali di PyQt che fornisce una serie di widget per creare interfacce grafiche. I widget sono elementi visivi che compongono la GUI, come ad esempio pulsanti, etichette, caselle di testo. Il modulo `QtWidgets` ci ha permesso di gestire la struttura e il layout dell'interfaccia in modo da avere un'applicazione desktop ben organizzata e facile da usare. Tra i widget principali che sono stati utilizzati abbiamo:

- `QPushButton`: pulsante cliccabile che è stato utilizzato per eseguire delle azioni quando viene premuto,
- `QLabel`: è utilizzato per visualizzare testo ed è stato usato in particolare per descrivere campi di input nelle pagine di aggiunta e modifica.
- `QLineEdit`: è utilizzato per fornire un campo di input di testo; è stato usato per far scrivere all'utente i dati che richiede o che vuole inserire.
- `QTableWidget`: widget utilizzato per visualizzare e gestire tabelle di dati. Come, ad esempio, il database dei prodotti,
- `QMainWindow`: è la finestra principale dell'applicazione.
- `QStackedWidget`: è un widget utilizzato per gestire più pagine in una singola area visibile; nel nostro caso tutte le pagine elencate nei mockup cambiano direttamente all'interno di `QStackedWidget` in modo da fornire un'interfaccia molto comoda dato che l'utente può passare da una pagina all'altra senza che si debbano aprire e chiudere ulteriori finestre ma stando sempre nella stessa.
- `QFrame` è un widget che fa da contenitore di altri widget e facilita l'interfaccia con l'uso di bordi e linee.
- `QVBoxLayout`: è un widget che dispone i widget in verticale all'interno di un contenitore o una finestra ridimensionando i widget al suo interno in base allo spazio disponibile; è stato molto utile per realizzare un'interfaccia che rispettasse gli spazi mantenendo un design ordinato e allineato.
- `QHBoxLayout`: è un widget che dispone i widget in orizzontale all'interno di un contenitore o una finestra ridimensionando i widget al suo interno in base allo spazio disponibile; è stato utilizzato, in particolare, per le pagine di registrazione per allineare i `QLabel` e i `QLineEdit`.
- `QSize` è una classe utilizzata per rappresentare le dimensioni; è stata utilizzata ad esempio per impostare una dimensione fissa ai pulsanti per chiudere, ingrandire, rimpicciolire o minimizzare la finestra.
- `QSpacerItem`: è una classe utilizzata per creare spazi vuoti all'interno di un layout; è stata utilizzata per creare un'interfaccia più organizzata, in modo da avere degli spazi vuoti tra i pulsanti e le tabelle.

4.3 Database

Come database è stato utilizzato DB Browser for SQLite che permette una facile interazione con il linguaggio Python.

4.3.1 DB Browser for SQLite

DB Browser for SQLite è un'applicazione che fornisce un'interfaccia grafica che permette all'utente di interagire con il database in modo semplice e intuitivo. Nel nostro sistema è stato utilizzato per semplificare la gestione e la visualizzazione dei dati.

All'interno della nostra base di dati abbiamo creato la struttura delle tabelle dei dati dei prodotti, dei clienti e delle aziende.

Nel Listato 4.1 viene riportato il codice SQL utilizzato per la creazione della tabella denominata `tabella_dati`, progettata per memorizzare informazioni sui prodotti, inclusi codice, nome, azienda fornitrice, prezzo e numero di scorte. Il codice è la chiave primaria, garantendo l'unicità di ogni record prodotto. Inoltre, stabilisce una relazione con la `tabella_aziende` tramite una chiave esterna, permettendo la cancellazione automatica dei prodotti associati se l'azienda viene eliminata.

```
CREATE TABLE "tabella_dati" (
    "CODICE" INTEGER,
    "NOME" TEXT,
    "AZIENDA" TEXT,
    "PREZZO" REAL,
    "SCORTE" INTEGER,
    PRIMARY KEY("CODICE"),
    FOREIGN KEY("AZIENDA") REFERENCES "tabella_aziende"("NOME") ON
        DELETE CASCADE
)
```

Listing 4.1: Definizione della tabella relativa ai dati

Nel Listato 4.2 viene riportato il codice SQL utilizzato per la creazione della tabella denominata `tabella_clienti`, progettata per memorizzare informazioni sui clienti, inclusi codice fiscale, nome, cognome, numero di telefono, data di nascita e numero dei prodotti acquistati, che viene inizializzato a zero. Il codice fiscale è la chiave primaria, garantendo l'unicità di ogni record cliente.

```
CREATE TABLE "tabella_clienti" (
    "CODICE_FISCALE" TEXT,
    "NOME" TEXT,
    "COGNOME" TEXT,
    "TELEFONO" TEXT,
    "DATA_DI_NASCITA" TEXT,
    "PRODOTTI_ACQUISTATI" INTEGER DEFAULT 0,
    PRIMARY KEY("CODICE_FISCALE")
)
```

Listing 4.2: Definizione della tabella relativa ai clienti

Nel Listato 4.3 viene riportato il codice SQL utilizzato per la creazione della tabella denominata `tabella_aziende`, progettata per memorizzare informazioni sulle aziende, inclusi codice, nome, tipologia, localizzazione e ragione sociale. Il codice è la chiave primaria, garantendo l'unicità di ogni record azienda.

```
CREATE TABLE "tabella_aziende" (
    "CODICE"        INTEGER,
    "NOME"          TEXT,
    "TIPOLOGIA"     TEXT,
    "LOCALIZZAZIONE"    TEXT,
    "RAGIONE_SOCIALE"   TEXT,
    PRIMARY KEY("CODICE")
)
```

Listing 4.3: Definizione della tabella relativa alle aziende

DB Browser for SQLite ci permette di effettuare delle query sull'applicazione stessa per inserire, modificare o eliminare dati. Inoltre, fornisce un'interfaccia grafica per poter visualizzare le tabelle create precedentemente, con i relativi dati. L'applicazione permette, anche, di sostituire l'utilizzo delle query con delle interazioni sull'interfaccia delle tabelle stesse; in questo modo, l'interfaccia grafica fornita semplifica l'utilizzo delle query in interazioni visive più semplici per l'utente. Le tabelle presenti nel database, con i relativi dati, sono: la `tabella_dati`, la `tabella_clienti` e la `tabella_aziende`.

Tabella Dati

Nella Figura 4.1 viene riportata la schermata su DB Browser for SQLite relativa alla `tabella_dati`.

CODICE	NOME	*AZIENDA*	PREZZO	SCORTE
Filtro	Filtro	Filtro	Filtro	Filtro
1	AIR FORCE 1	NIKE	90.0	97
2	JORDAN AIR IV	JORDAN	260.0	6
3	CAMPUS	ADIDAS	80.0	70
4	THE LAST CAMPUS	ADIDAS	160.0	2
5	OLD SKOOL	VANS	70.0	50
6	KNU SKOOL	VANS	65.0	80
7	BLAZER	NIKE	60.0	50
8	SAMBA	ADIDAS	80.0	90

Figura 4.1: Schermata su DB Browser for SQLite relativa alla tabella sui dati

Tabella Clienti

Nella Figura 4.2 viene riportata la schermata su DB Browser for SQLite relativa alla `tabella_clienti`.

CODICE_FISCALE	NOME	COGNOME	TELEFONO	DATA_DI_NASCITA	PRODOTTI_ACQUISTATI
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
DCRDNR98R03G987X	DANIEL	DUBOIS	3985534734	17/10/98	0
GRVDVS96R17G100X	GERVONTA	DAVIS	3269542019	03/06/96	3
TFOLPZ98R23T333X	TEOFIMO	LOPEZ	3750162476	10/10/98	2
DVNHNY97D20Y456X	DEVIN	HANEY	3452870120	03/06/99	0
FGHJKFHGJ	GHJKGHJK	DGFHJCVHBJN	1234567890	12-02-2012	0
JU76TRFGJKJUYU8	ASDFG	GHJK	234567890	12-01-2000	0

Figura 4.2: Schermata su DB Browser for SQLite relativa alla tabella sui clienti

Tabella Aziende

Nella Figura 4.3 viene riportata la schermata su DB Browser for SQLite relativa alla `tabella_aziende`.

CODICE	NOME	TIPOLOGIA	LOCALIZZAZIONE	RAGIONE_SOCIALE
Filtro	Filtro	Filtro	Filtro	Filtro
1	JORDAN	ABBIGLIAMENTO SPORTIVO	USA	JORDAN S.R.L.
2	NIKE	ABBIGLIAMENTO SPORTIVO	USA	NIKE, INC
3	ADIDAS	ABBIGLIAMENTO SPORTIVO	GERMANIA	ADIDAS S.P.A.
4	VANS	CALZATURE	USA	VANS S.R.L.

Figura 4.3: Schermata su DB Browser for SQLite relativa alla tabella sulle aziende

4.3.2 Implementazione del database su python

Abbiamo implementato un sistema di gestione del database in Python utilizzando il modulo `sqlite3`, che permette di interagire con un database SQLite. La sua classe `Comunicazione` contiene diversi metodi per gestire le operazioni CRUD (Create, Read, Update, Delete) su tre tabelle: `tabella_dati`, `tabella_clienti` e `tabella_aziende`. Di seguito è riportata una descrizione delle funzionalità che abbiamo implementato.

Inserimento Prodotto

Nel Listato 4.4 viene riportato il codice Python che definisce la funzione `inserisci_prodotto`, che permette di inserire un nuovo prodotto nel database.

```
def inserisci_prodotto(self, codice, nome, azienda, prezzo, scorte)
    :
    cursor = self.connessione.cursor()
    bd = '''INSERT INTO tabella_dati (CODICE, NOME, AZIENDA, PREZZO
        , SCORTE)
            VALUES (?, ?, ?, ?, ?)'''
    cursor.execute(bd, (codice, nome, azienda, prezzo, scorte))
    self.connessione.commit()
    cursor.close()
```

Listing 4.4: Funzione per inserire un prodotto nel database

Mostra Prodotti

Nel Listato 4.5 viene riportato il codice Python che defnisce la funzione `mostra_prodotti`, che permette di mostrare tutti i prodotti del database.

```
def mostra_prodotti(self):
    cursor = self.connessione.cursor()
    bd = 'SELECT * FROM tabella_dati'
    cursor.execute(bd)
    registro = cursor.fetchall()
    return registro
```

Listing 4.5: Funzione per mostrare tutti i prodotti nel database

Cerca Prodotto

Nel Listato 4.6 viene riportato il codice Python che definisce la funzione `cerca_prodotto`, che permette di cercare un prodotto nel database tramite il nome del prodotto stesso.

```
def cerca_prodotto(self, nome_prodotto):
    cursor = self.connessione.cursor()
    bd = '''SELECT * FROM tabella_dati WHERE NOME = ?'''
    cursor.execute(bd, (nome_prodotto,))
    risultati = cursor.fetchall()
    cursor.close()
    return risultati
```

Listing 4.6: Funzione per cercare un prodotto nel database tramite il nome

Elimina Prodotti

Nel Listato 4.7 viene riportato il codice Python che definisce la funzione `elimina_prodotti`, che permette di eliminare un prodotto dal database.

```
def elimina_prodotti(self, nome):
    cursor = self.connessione.cursor()
    bd = '''DELETE FROM tabella_dati WHERE NOME = ?'''
    cursor.execute(bd, (nome,))
    self.connessione.commit()
    cursor.close()
```

Listing 4.7: Funzione per eliminare un prodotto dal database

Modifica Prodotti

Nel Listato 4.8 viene riportato il codice Python che definisce la funzione `modifica_prodotti`, che permette di modificare un prodotto presente nel database.

```
def modifica_prodotti(self, codice, nome, azienda, prezzo, scorte):
    cursor = self.connessione.cursor()
    bd = '''UPDATE tabella_dati
            SET NOME = ?, AZIENDA = ?, PREZZO = ?, SCORTE = ?
            WHERE CODICE = ?'''
    cursor.execute(bd, (nome, azienda, prezzo, scorte, codice))
    a = cursor.rowcount
    self.connessione.commit()
    cursor.close()
    return a
```

Listing 4.8: Funzione per modificare un prodotto esistente nel database

Inserisci Cliente

Nel Listato 4.9 viene riportato il codice Python che definisce la funzione `inserisci_cliente`, che permette di inserire un cliente nel database.

```
def inserisci_cliente(self, codiceFiscale, nome, cognome, telefono,
    dataDiNascita, prodottiAcquistati):
    cursor = self.connessione.cursor()
    bd = '''INSERT INTO tabella_clienti (CODICE_FISCALE, NOME,
       COGNOME, TELEFONO, DATA_DI_NASCITA, PRODOTTI_ACQUISTATI)
            VALUES (?, ?, ?, ?, ?, ?)'''
    cursor.execute(bd, (codiceFiscale, nome, cognome, telefono,
       dataDiNascita, 0))
    self.connessione.commit()
    cursor.close()
```

Listing 4.9: Funzione per inserire un cliente nel database

Mostra Clienti

Nel Listato 4.10 viene riportato il codice Python che definisce la funzione `mostra_clienti`, che permette di mostrare tutti i clienti del database.

```
def mostra_clienti(self):
    cursor = self.connessione.cursor()
    bd = 'SELECT * FROM tabella_clienti'
    cursor.execute(bd)
    registro = cursor.fetchall()
    return registro
```

Listing 4.10: Funzione per mostrare tutti i clienti nel database

Cerca Cliente

Nel Listato 4.11 viene riportato il codice Python che definisce la funzione `cerca_cliente`, che permette di cercare un cliente nel database tramite il nome del cliente stesso.

```
def cerca_cliente(self, nome_cliente):
    cursor = self.connessione.cursor()
    bd = '''SELECT * FROM tabella_clienti WHERE NOME = ?'''
    cursor.execute(bd, (nome_cliente,))
    return cursor.fetchall()
```

Listing 4.11: Funzione per cercare un cliente nel database tramite il suo nome

Elimina Clienti

Nel Listato 4.12 viene riportato il codice Python che definisce la funzione `elimina_clienti`, che permette di eliminare un cliente dal database.

```
def elimina_clienti(self, nome_cliente):
    cursor = self.connessione.cursor()
    bd = '''DELETE FROM tabella_clienti WHERE NOME = ?'''
    cursor.execute(bd, (nome_cliente,))
    self.connessione.commit()
```

Listing 4.12: Funzione per eliminare un cliente dal database

Modifica Clienti

Nel Listato 4.13 viene riportato il codice Python che definisce la funzione `modifica_clienti`, che permette di modificare un cliente presente nel database.

```
def modifica_clienti(self, codiceFiscale, nome, cognome, telefono,
    dataDiNascita):
    cursor = self.connessione.cursor()
    bd = '''UPDATE tabella_clienti
            SET NOME = ?, COGNOME = ?, TELEFONO = ?,
                DATA_DI_NASCITA = ?
            WHERE CODICE_FISCALE = ?'''
    cursor.execute(bd, (nome, cognome, telefono, dataDiNascita,
        codiceFiscale))
    a = cursor.rowcount
    self.connessione.commit()
    cursor.close()
    return a
```

Listing 4.13: Funzione per modificare un cliente esistente nel database

Togli Scorte

Nel Listato 4.14 viene riportato il codice Python che definisce la funzione `togli_scorte`, che permette di togliere delle scorte da un prodotto presente nel database.

```
def togli_scorte(self, codice, nuove_scorte):
    cursor = self.connessione.cursor()
    bd = '''UPDATE tabella_dati
            SET SCORTE = ?
            WHERE CODICE = ?'''
    cursor.execute(bd, (nuove_scorte, codice))
    a = cursor.rowcount
    self.connessione.commit()
    cursor.close()
    return a
```

Listing 4.14: Funzione per aggiornare le scorte di un prodotto nel database

Aggiungi Prodotti

Nel Listato 4.15 viene riportato il codice Python che definisce la funzione `aggiungi_prodotti`, che permette di aggiungere dei prodotti a un cliente presente nel database.

```
def aggiungi_prodotti(self, codiceFiscale, nuovi_prodottiAcquistati
    ):
    cursor = self.connessione.cursor()
    bd = '''UPDATE tabella_clienti SET PRODOTTI_ACQUISTATI = ?
            WHERE CODICE_FISCALE = ?'''
    cursor.execute(bd, (nuovi_prodottiAcquistati, codiceFiscale))
    self.connessione.commit()
```

Listing 4.15: Funzione per aggiungere prodotti acquistati da un cliente

Inserisci Aziende

Nel Listato 4.16 viene riportato il codice Python che definisce la funzione `inserisci_aziende`, che permette di inserire un'azienda nel database.

```
def inserisci_aziende(self, codice, nome, tipologia, localizzazione
    , ragioneSociale):
    cursor = self.connessione.cursor()
    bd = '''INSERT INTO tabella_aziende (CODICE, NOME, TIPOLOGIA,
       LOCALIZZAZIONE, RAGIONE_SOCIALE)
            VALUES (?, ?, ?, ?, ?)'''
    cursor.execute(bd, (codice, nome, tipologia, localizzazione,
       ragioneSociale))
    self.connessione.commit()
    cursor.close()
```

Listing 4.16: Funzione per inserire un'azienda nel database

Mostra Aziende

Nel Listato 4.17 viene riportato il codice Python che definisce la funzione `mostra_aziende`, che permette di mostrare tutte le aziende del database.

```
def mostra_aziende(self):
    cursor = self.connessione.cursor()
    bd = 'SELECT * FROM tabella_aziende'
    cursor.execute(bd)
    registro = cursor.fetchall()
    return registro
```

Listing 4.17: Funzione per mostrare tutte le aziende nel database

Cerca Azienda

Nel Listato 4.18 viene riportato il codice Python che definisce la funzione `cerca_azienda`, che permette di cercare un'azienda nel database tramite il nome dell'azienda stessa.

```
def cerca_azienda(self, nome):
    cursor = self.connessione.cursor()
    bd = '''SELECT * FROM tabella_aziende WHERE NOME = ?'''
    cursor.execute(bd, (nome,))
    return cursor.fetchall()
```

Listing 4.18: Funzione per cercare un'azienda nel database tramite il corrispettivo nome

Elimina Aziende

Nel Listato 4.19 viene riportato il codice Python che definisce la funzione `elimina_aziende`, che permette di eliminare un'azienda dal database.

```
def elimina_aziende(self, nome):
    cursor = self.connessione.cursor()
    bd = '''DELETE FROM tabella_aziende WHERE NOME = ?'''
    cursor.execute(bd, (nome,))
    self.connessione.commit()
```

Listing 4.19: Funzione per eliminare un'azienda dal database

Modifica Aziende

Nel Listato 4.20 viene riportato il codice Python che definisce la funzione `modifica_aziende`, che permette di modificare un'azienda presente nel database.

```
def modifica_aziende(self, codice, nome, tipologia, localizzazione,
    ragioneSociale):
    cursor = self.connessione.cursor()
    bd = '''UPDATE tabella_aziende SET CODICE = ?, NOME = ?,
       TIPOLOGIA = ?, LOCALIZZAZIONE = ?, RAGIONE_SOCIALE = ?
            WHERE CODICE = ?'''
    cursor.execute(bd, (codice, nome, tipologia, localizzazione,
       ragioneSociale, codice))
    self.connessione.commit()
```

Listing 4.20: Funzione per modificare un'azienda esistente nel database

4.4 Testing

In questa sezione, presentiamo il codice per testare la funzionalità del sistema di gestione del magazzino. Di seguito sono riportate le istruzioni su come eseguire i test utilizzando il framework `pytest`.

4.4.1 Istruzioni per l'Esecuzione dei Test

Per eseguire i test, bisogna aver installato `pytest`. Nel Listato 4.21 viene riportato il codice bash da inserire nel terminale per installarlo correttamente.

```
pipx install pytest
```

Listing 4.21: Comando per installare `pytest`

Una volta installato `pytest`, è possibile eseguire i test dal terminale dopo essere entrati nella directory del proprio progetto. Nel Listato 4.22 viene rappresentato il codice bash da inserire nel terminale per effettuare il test.

```
pytest test.py
```

Listing 4.22: Comando per eseguire i test

4.4.2 Configurazione del Database

Nel Listato 4.23 viene riportato il codice Python per configurare un ambiente di test per un'applicazione utilizzando `pytest` e `SQLite`. Nel listato viene definito un fixture `db()` che crea un database temporaneo in memoria con tre tabelle: `tabella_dati` per i prodotti, `tabella_clienti` per i clienti e `tabella_aziende` per le aziende. Ogni tabella ha una

chiave primaria e contiene vari campi per memorizzare informazioni pertinenti. Alla fine dei test, la connessione al database viene chiusa per liberare risorse.

```
import pytest
import sqlite3
from connessione_sqlite import Comunicazione

@pytest.fixture
def db():
    conn = sqlite3.connect(':memory:')
    cursor = conn.cursor()

    cursor.execute('''
    CREATE TABLE tabella_dati (
        CODICE TEXT PRIMARY KEY,
        NOME TEXT,
        AZIENDA TEXT,
        PREZZO REAL,
        SCORTE INTEGER
    )''')
    cursor.execute('''
    CREATE TABLE tabella_clienti (
        CODICE_FISCALE TEXT PRIMARY KEY,
        NOME TEXT,
        COGNOME TEXT,
        TELEFONO TEXT,
        DATA_DI_NASCITA TEXT,
        PRODOTTI_ACQUISTATI INTEGER
    )''')
    cursor.execute('''
    CREATE TABLE tabella_aziende (
        CODICE TEXT PRIMARY KEY,
        NOME TEXT,
        TIPOLOGIA TEXT,
        LOCALIZZAZIONE TEXT,
        RAGIONE_SOCIALE TEXT
    )''')

    conn.commit()
    yield conn
    conn.close()
```

Listing 4.23: Configurazione del database per i test

4.4.3 Fixture per Comunicazione

Nel Listato 4.24 viene riportato il codice Python che definisce un fixture di comunicazione per i test utilizzando `pytest`. La funzione crea un'istanza della classe `Comunicazione` e assegna ad essa una connessione al database, passata dal fixture `db` . A questo punto restituisce l'istanza configurata, consentendo ai test di utilizzare la comunicazione con il database senza doverla configurare ripetutamente.

```
@pytest.fixture
def comunicazione(db):
    com = Comunicazione()
    com.connessione = db
    return com
```

Listing 4.24: Fixture per la classe `Comunicazione`

4.4.4 Test per Inserire un Prodotto

Nel Listato 4.25 viene riportato il codice Python che definisce un test per verificare l'inserimento di un prodotto nella classe `Comunicazione`, assicurandosi che il prodotto stesso sia stato correttamente aggiunto e che i dettagli siano corretti.

```
def test_inserisci_prodotto(comunicazione):
    comunicazione.inserisci_prodotto("123", "ProdottoTest", "
        AziendaTest", 10.99, 100)
    prodotti = comunicazione.mostra_prodotti()
    assert len(prodotti) == 1
    assert prodotti[0][0] == "123"
    assert prodotti[0][1] == "ProdottoTest"
```

Listing 4.25: Test per inserire un prodotto

4.4.5 Test per Modificare un Prodotto

Nel Listato 4.26 viene riportato il codice Python che definisce un test per verificare la funzionalità di modifica di un prodotto, assicurandosi che i dettagli del prodotto siano stati aggiornati correttamente dopo la modifica.

```
def test_modifica_prodotto(comunicazione):
    comunicazione.inserisci_prodotto("123", "ProdottoTest", "
        AziendaTest", 10.99, 100)
    comunicazione.modifica_prodotti("123", "ProdottoModificato", "
        AziendaModificata", 15.99, 150)
    prodotti = comunicazione.mostra_prodotti()
    assert prodotti[0][1] == "ProdottoModificato"
```

Listing 4.26: Test per modificare un prodotto

4.4.6 Test per Eliminare un Prodotto

Nel Listato 4.27 viene riportato il codice Python che definisce un test per verificare la funzionalità di eliminazione di un prodotto. Dopo aver inserito un prodotto, il test controlla che la lista dei prodotti sia vuota dopo l'eliminazione.

```
def test_elimina_prodotto(comunicazione):
    comunicazione.inserisci_prodotto("123", "ProdottoTest", "
        AziendaTest", 10.99, 100)
    comunicazione.elimina_prodotti("ProdottoTest")
    prodotti = comunicazione.mostra_prodotti()
    assert len(prodotti) == 0
```

Listing 4.27: Test per eliminare un prodotto

4.4.7 Test per Cercare un Prodotto

Nel Listato 4.28 viene riportato il codice Python che definisce un test per verificare la funzionalità di ricerca di un prodotto. Il test verifica che la ricerca restituisca correttamente il prodotto inserito.

```
def test_cerca_prodotto(comunicazione):
    comunicazione.inserisci_prodotto("123", "ProdottoTest", "
        AziendaTest", 10.99, 100)
    prodotto = comunicazione.cerca_prodotto("ProdottoTest")
    assert len(prodotto) == 1
    assert prodotto[0][1] == "ProdottoTest"
```

Listing 4.28: Test per cercare un prodotto

4.4.8 Test per Togliere Scorte

Nel Listato 4.29 viene riportato il codice Python che definisce un test per verificare la funzionalità di riduzione delle scorte di un prodotto. Dopo aver ridotto le scorte del prodotto di 50 unità, il test controlla che il numero di scorte rimanenti sia corretto.

```
def test_togli_scorte(comunicazione):
    comunicazione.inserisci_prodotto("123", "ProdottoTest", "
        AziendaTest", 10.99, 100)
    comunicazione.togli_scorte("123", 50)
    prodotti = comunicazione.mostra_prodotti()
    assert prodotti[0][4] == 50
```

Listing 4.29: Test per togliere delle scorte

4.4.9 Test per Inserire un Cliente

Nel Listato 4.30 viene riportato il codice Python che definisce un test per verificare l'inserimento di un cliente tramite la classe `Comunicazione`, assicurandosi che il cliente sia stato aggiunto correttamente e che i dettagli siano corretti.

```
def test_inserisci_cliente(comunicazione):
    comunicazione.inserisci_cliente("ABC123", "NomeTest", "
        CognomeTest", "1234567890", "01-01-2000", 0)
    clienti = comunicazione.mostra_clienti()
    assert len(clienti) == 1
    assert clienti[0][0] == "ABC123"
    assert clienti[0][5] == 0  # Assicuriamoci che prodotti
        acquistati siano inizializzati a 0
```

Listing 4.30: Test per inserire un cliente

4.4.10 Test per Modificare un Cliente

Nel Listato 4.31 viene riportato il codice Python che definisce un test per verificare la funzionalità di modifica di un cliente, assicurandosi che i dettagli del cliente siano stati aggiornati correttamente dopo la modifica.

```
def test_modifica_cliente(comunicazione):
    comunicazione.inserisci_cliente("ABC123", "NomeTest", "
        CognomeTest", "1234567890", "01-01-2000", 0)
    comunicazione.modifica_clienti("ABC123", "NuovoNome", "
        CognomeModificato", "0987654321", "02-02-2000")
    clienti = comunicazione.mostra_clienti()
    assert clienti[0][1] == "NuovoNome"
```

Listing 4.31: Test per modificare un cliente

4.4.11 Test per Eliminare un Cliente

Nel Listato 4.32 viene riportato il codice Python che definisce un test per verificare la funzionalità di eliminazione di un cliente. Dopo aver specificato un cliente, si verifica che la lista dei clienti sia vuota dopo l'eliminazione.

```
def test_elimina_cliente(comunicazione):
    comunicazione.inserisci_cliente("ABC123", "NomeTest", "
        CognomeTest", "1234567890", "01-01-2000", 0)
    comunicazione.elimina_clienti("NomeTest")
    clienti = comunicazione.mostra_clienti()
    assert len(clienti) == 0
```

Listing 4.32: Test per eliminare un cliente

4.4.12 Test per Cercare un Cliente

Nel Listato 4.33 viene riportato il codice Python che definisce un test per verificare la funzionalità di ricerca di un cliente. Il test verifica che la ricerca restituisca correttamente il cliente inserito.

```
def test_cerca_cliente(comunicazione):
    comunicazione.inserisci_cliente("ABC123", "NomeTest", "
        CognomeTest", "1234567890", "01-01-2000", 0)
    cliente = comunicazione.cerca_cliente("NomeTest")
    assert len(cliente) == 1
    assert cliente[0][1] == "NomeTest"
```

Listing 4.33: Test per cercare un cliente

4.4.13 Test per Aggiungere Prodotti Acquistati a un Cliente

Nel Listato 4.34 viene riportato il codice Python che funge da test per la funzionalità di aggiunta di prodotti acquistati a un cliente. Il test verifica che il conteggio dei prodotti acquistati del cliente sia aggiornato correttamente.

```
def test_aggiungi_prodotti(comunicazione):
    comunicazione.inserisci_cliente("ABC123", "NomeTest", "
        CognomeTest", "1234567890", "01-01-2000", 0)
    comunicazione.aggiungi_prodotti("ABC123", 5)  # Aggiungiamo 5
        prodotti acquistati
    cliente = comunicazione.mostra_clienti()
```

```
    assert cliente[0][5] == 5  # Verifica che il numero di prodotti
        acquistati sia ora 5
```

Listing 4.34: Test per aggiungere prodotti acquistati a un cliente

4.4.14 Test per Inserire un'Azienda

Nel Listato 4.35 viene riportato il codice Python che definisce un test per verificare l'inserimento di un'azienda tramite la classe `Comunicazione`, assicurandosi che l'azienda sia stata aggiunta correttamente e che i dettagli siano corretti.

```
def test_inserisci_azienda(comunicazione):
    comunicazione.inserisci_aziende("XYZ123", "AziendaTest", "
        TipologiaTest", "LocalizzazioneTest", "RagioneSocialeTest")
    aziende = comunicazione.mostra_aziende()
    assert len(aziende) == 1
    assert aziende[0][0] == "XYZ123"
```

Listing 4.35: Test per inserire un'azienda

4.4.15 Test per Modificare un'Azienda

Nel Listato 4.36 viene riportato il codice Python che definisce un test per verificare la funzionalità di modifica di un'azienda, assicurandosi che i dettagli dell'azienda siano stati aggiornati correttamente dopo la modifica.

```
def test_modifica_azienda(comunicazione):
    comunicazione.inserisci_aziende("XYZ123", "AziendaTest", "
        TipologiaTest", "LocalizzazioneTest", "RagioneSocialeTest")
    comunicazione.modifica_aziende("XYZ123", "NuovaAzienda", "
        NuovaTipologia", "NuovaLocalizzazione", "NuovaRagione")
    aziende = comunicazione.mostra_aziende()
    assert aziende[0][1] == "NuovaAzienda"
```

Listing 4.36: Test per modificare un'azienda

4.4.16 Test per Eliminare un'Azienda

Nel Listato 4.37 viene riportato il codice Python che definisce un test per verificare la funzionalità di eliminazione di un'azienda. Dopo aver specificato un'azienda, si verifica che la lista delle aziende sia vuota dopo l'eliminazione.

```
def test_elimina_azienda(comunicazione):
    comunicazione.inserisci_aziende("XYZ123", "AziendaTest", "
        TipologiaTest",
    "LocalizzazioneTest", "RagioneSocialeTest")
    comunicazione.elimina_aziende("AziendaTest")
    aziende = comunicazione.mostra_aziende()
    assert len(aziende) == 0
```

Listing 4.37: Test per eliminare un'azienda

4.4.17 Test per Cercare un'Azienda

Nel Listato 4.38 viene riportato il codice Python che definisce un test per verificare la funzionalità di ricerca di un'azienda. Il test verifica che la ricerca restituisca correttamente l'azienda inserita.

```
def test_cerca_azienda(comunicazione):
    comunicazione.inserisci_aziende("XYZ123", "AziendaTest", "
        TipologiaTest", "LocalizzazioneTest", "RagioneSocialeTest")
    azienda = comunicazione.cerca_azienda("AziendaTest")
    assert len(azienda) == 1
    assert azienda[0][1] == "AziendaTest"
```

Listing 4.38: Test per cercare un'azienda

CAPITOLO 5

Manuale Utente

In questo capitolo forniremo una guida completa per l'utilizzo del sistema di gestione del magazzino. Verranno illustrate le principali funzionalità dell'applicazione, offrendo una panoramica sull'interfaccia utente, sulle operazioni che possono essere eseguite e sulle modalità di interazione con il software. Inoltre, verranno descritti i passaggi necessari per l'installazione e la configurazione dell'applicazione, garantendo che gli utenti possano avviare il sistema con successo. L'obiettivo è quello di rendere il processo di utilizzo il più semplice possibile, assicurando un'esperienza positiva per gli utenti, sia tecnici che non.

5.1 Visualizzazione dell'interfaccia utente

Le schermate dell'interfaccia utente del sistema di gestione del magazzino forniscono una rappresentazione visiva delle principali funzionalità implementate, permettendo di vedere direttamente l'aspetto finale del sistema. Queste schermate sono uno strumento cruciale nella fase di sviluppo, poiché mostrano come sarà il layout e come l'utente interagirà con il sistema.

Ogni schermata è stata progettata per gestire operazioni come la gestione dei prodotti, dei clienti, delle aziende fornitrici e degli acquisti. L'obiettivo è quello di offrire un'interfaccia chiara, intuitiva e funzionale, semplificando l'utilizzo grazie a un'organizzazione efficace di elementi come campi di input, pulsanti e tabelle. Le schermate permettono di migliorare l'usabilità, consentendo di ottimizzare il flusso di lavoro dell'utente durante l'interazione con il sistema.

5.1.1 Gestione dei prodotti

Le schermate relative alla gestione dei prodotti nel sistema gestionale del magazzino svolgono un ruolo importante nella progettazione dell'interfaccia utente, poiché mostrano in modo diretto le funzionalità e l'organizzazione del layout. Attraverso queste schermate, è possibile visualizzare come gli utenti interagiranno con il sistema per eseguire operazioni come la visualizzazione, l'inserimento, la modifica e l'eliminazione dei prodotti.

Inoltre la finestra "Mostra Prodotti" rappresenta la schermata iniziale del sistema quando viene avviato, offrendo un accesso immediato alle principali funzionalità.

Database

Nella Figura 5.1 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Database`, che rappresente la pagina iniziale del sistema. Questa pagina consente di visualizzare i prodotti nel database ed è accessibile anche cliccando sul pulsante `DATABASE`.

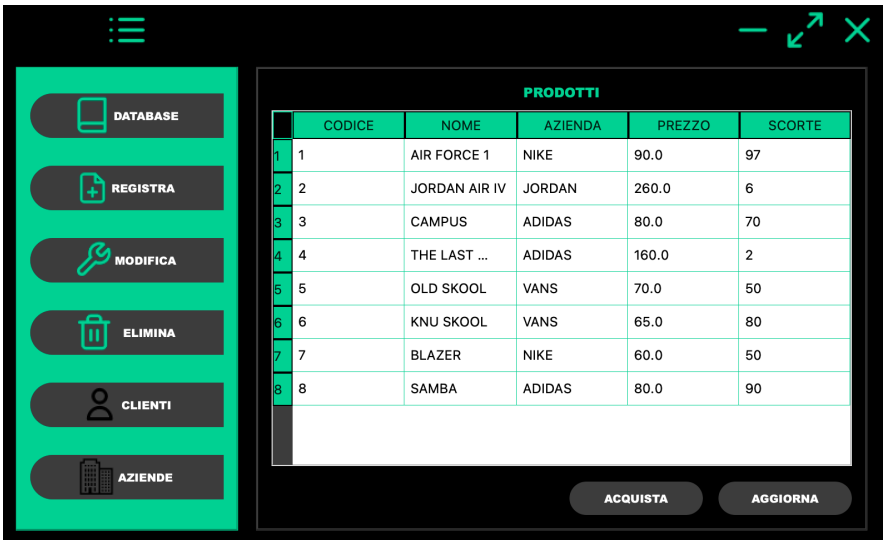

Figura 5.1: Schermata relativa alla gestione dei prodotti: Database

Registra

Nella Figura 5.2 è mostra la schermata dell'interfaccia utente relativa alla pagina `Registra`, che consente di inserire un nuovo prodotto nel database. Questa pagina è accessibile cliccando sul pulsante `REGISTRA`.

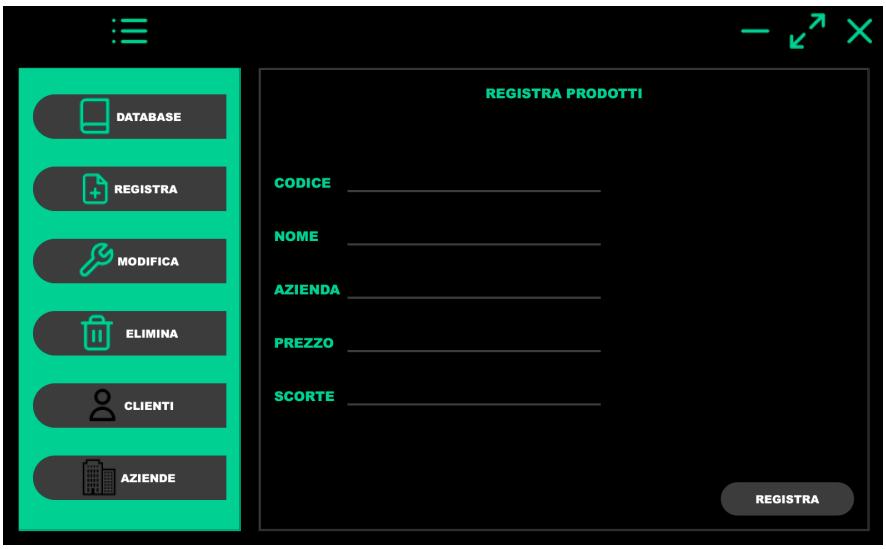

Figura 5.2: Schermata relativa alla gestione dei prodotti: Registra

Modifica

Nella Figura 5.3 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Modifica`, che consente di modificare un prodotto già presente nel database. Questa pagina è accessibile cliccando sul pulsante `MODIFICA`.

Figura 5.3: Schermata relativa alla gestione dei prodotti: Modifica

Elimina

Nella Figura 5.4 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Elimina`, che consente di eliminare un prodotto già presente nel database. Questa pagina è accessibile cliccando sul pulsante `ELIMINA`.

Figura 5.4: Schermata relativa alla gestione dei prodotti: Elimina

5.1.2 Gestione dei clienti

Clienti

Nella Figura 5.5 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Clienti`, che consente di visualizzare i clienti nel database. Questa pagina è accessibile cliccando sul pulsante `CLIENTI`.

Figura 5.5: Schermata relativa alla gestione dei clienti: Clienti

Aggiungi Cliente

Nella Figura 5.6 è mostra la schermata dell'interfaccia utente relativa alla pagina `Aggiungi Cliente`, che consente di inserire un nuovo cliente nel database. Questa pagina è accessibile cliccando sul pulsante `AGGIUNGI` dalla pagina `Clienti`.

Figura 5.6: Schermata relativa alla gestione dei clienti: Aggiungi Cliente

Modifica Cliente

Nella Figura 5.7 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Modifica Cliente`, che consente di modificare un cliente già presente nel database. Questa pagina è accessibile cliccando sul pulsante `MODIFICA` dalla pagina `Clienti`.

Figura 5.7: Schermata relativa alla gestione dei clienti: Modifica Cliente

Elimina Cliente

Nella Figura 5.8 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Elimina Cliente`, che consente di eliminare un cliente già presente nel database. Questa pagina è accessibile cliccando sul pulsante `ELIMINA` dalla pagina `Clienti`.

Figura 5.8: Schermata relativa alla gestione dei clienti: Elimina Cliente

5.1.3 Gestione delle aziende

Aziende

Nella Figura 5.9 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Aziende`, che consente di visualizzare i clienti nel database. Questa pagina è accessibile cliccando sul pulsante `AZIENDE`.

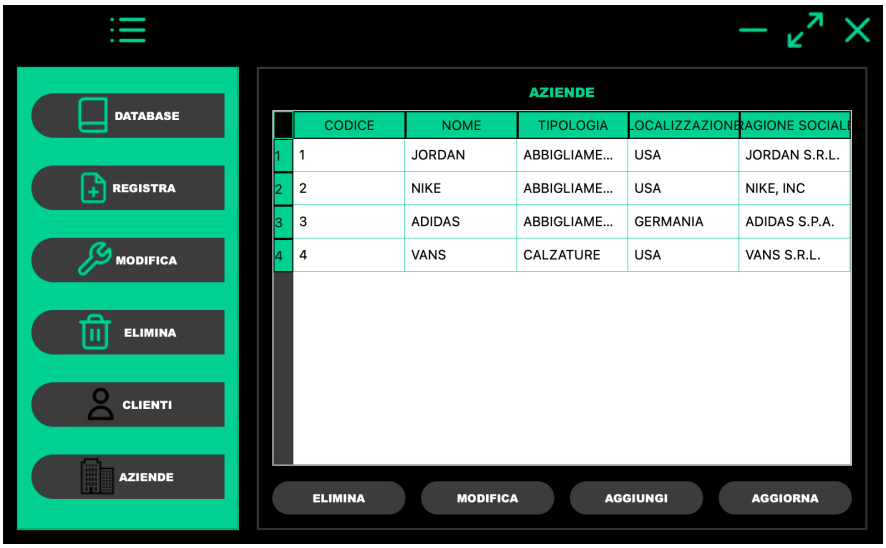

Figura 5.9: Schermata relativa alla gestione delle aziende: Aziende

Aggiungi Azienda

Nella Figura 5.10 è mostra la schermata dell'interfaccia utente relativa alla pagina `Aggiungi Azienda`, che consente di inserire un nuovo cliente nel database. Questa pagina è accessibile cliccando sul pulsante `AGGIUNGI` dalla pagina `Aziende`.

Figura 5.10: Schermata relativa alla gestione delle aziende: Aggiungi Azienda

Modifica Azienda

Nella Figura 5.11 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Modifica Azienda`, che consente di modificare un'azienda già presente nel database. Questa pagina è accessibile cliccando sul pulsante `MODIFICA` dalla pagina `Aziende`.

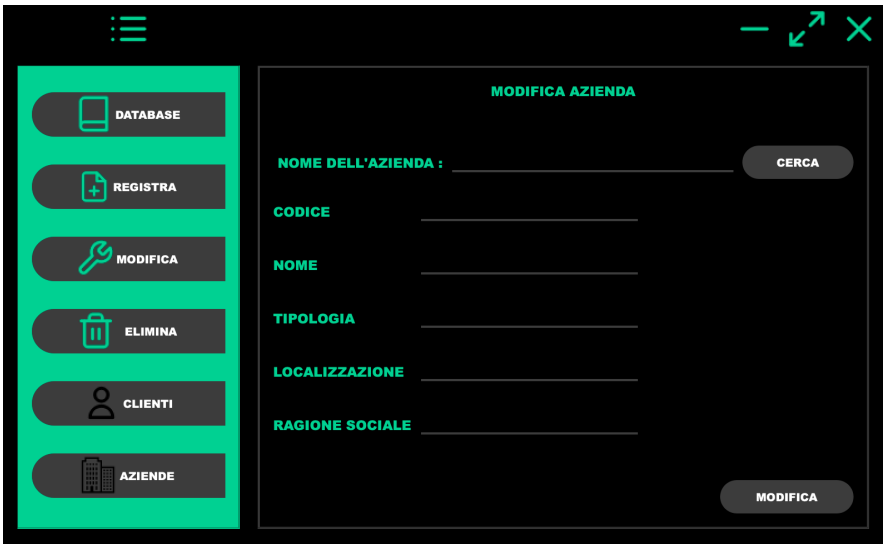

Figura 5.11: Schermata relativa alla gestione delle aziende: Modifica Azienda

Elimina Azienda

Nella Figura 5.12 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Elimina Azienda`, che consente di eliminare un'azienda già presente nel database. Questa pagina è accessibile cliccando sul pulsante `ELIMINA` dalla pagina `Aziende`.

Figura 5.12: Schermata relativa alla gestione delle aziende: Elimina Azienda

5.1.4 Gestione degli acquisti

Acquista

Nella Figura 5.13 è mostrata la schermata dell'interfaccia utente relativa alla pagina `Acquista`, che consente di effettuare un acquisto, dati un prodotto e un cliente già presenti nel database. Questa pagina è accessibile cliccando sul pulsante `ACQUISTA` dalla pagina `Database`.

Figura 5.13: Schermata relativa alla gestione degli acquisti: Acquista

5.2 Test dell'applicazione

Abbiamo effettuato dei test funzionali sul software per verificare che si comporti come previsto in situazioni reali di utilizzo da parte dell'utente. Abbiamo considerato le funzioni di inserimento, modifica ed eliminazione di un prodotto (i test sono relativi soltanto alla gestione del prodotto perché le funzioni hanno la stessa logica per i clienti e per le aziende). Inoltre abbiamo anche effettuato il test sulla funzione di acquisto di un prodotto da parte di un cliente.

5.2.1 Test relativi all'inserimento

Abbiamo effettuato dei test per verificare che le funzioni di inserimento si comportino come previsto, con i relativi messaggi da parte del sistema all'utente nei casi in cui egli ha eseguito una determinata funzione erroneamente o con successo.

Nel caso in cui l'utente reegistri un prodotto lasciando degli spazi vuoti il sistema deve impedire l'operazione e avvisare l'utente del suo errore. Invece, nel caso in cui egli abbia inserito i dati correttamente, il sistema accetterà l'operazione e lo avviserà che il prodotto è stato registrato con successo.

Registra Errore

Nella Figura 5.14 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di aggiunta dei prodotti, nel caso in cui vengono lasciati alcuni campi vuoti. Il sistema, in questo caso, restituisce un messaggio di errore per comunicare all'utente che ci sono degli spazi vuoti.

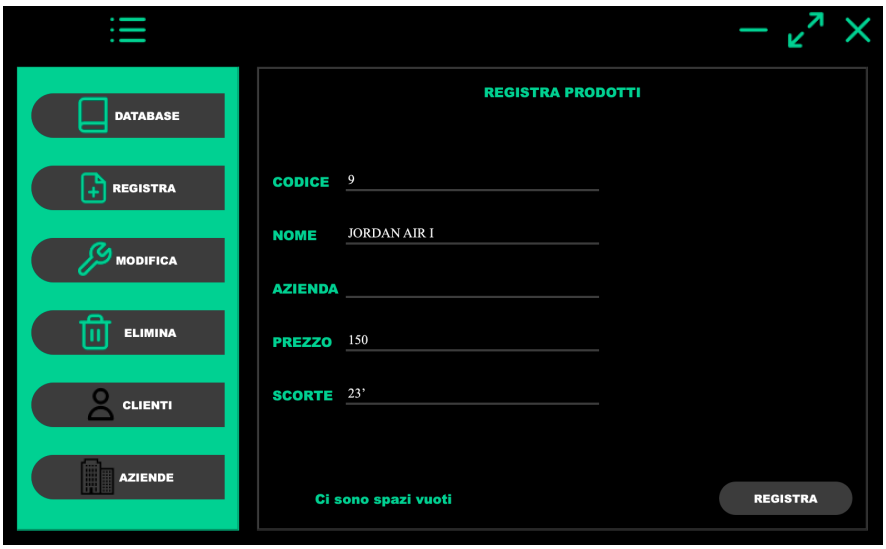

Figura 5.14: Schermata relativa ai test sull'inserimento: Registra Errore

Registra Corretto

Nella Figura 5.15 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di aggiunta dei prodotti nel caso in cui l'utente abbia inserito i dati correttamente prima di premere il pulsante `REGISTRA`.

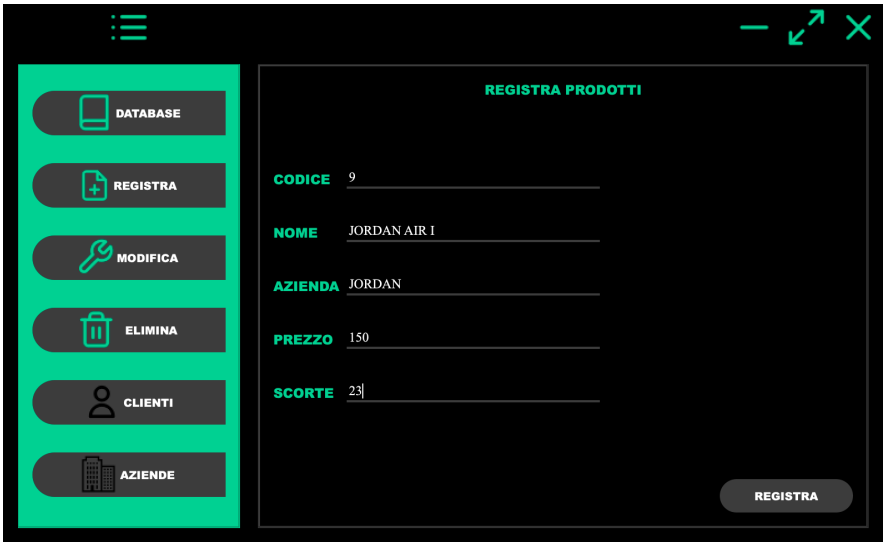

Figura 5.15: Schermata relativa ai test sull'inserimento: Registra Corretto

Registra Registrato

Nella Figura 5.16 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di aggiunta dei prodotti nel caso in cui l'utente abbia inserito i dati correttamente dopo aver premuto il pulsante `REGISTRA`. Il sistema in questo caso, restituisce un messaggio per comunicare all'utente che il prodotto è stato registrato con successo.

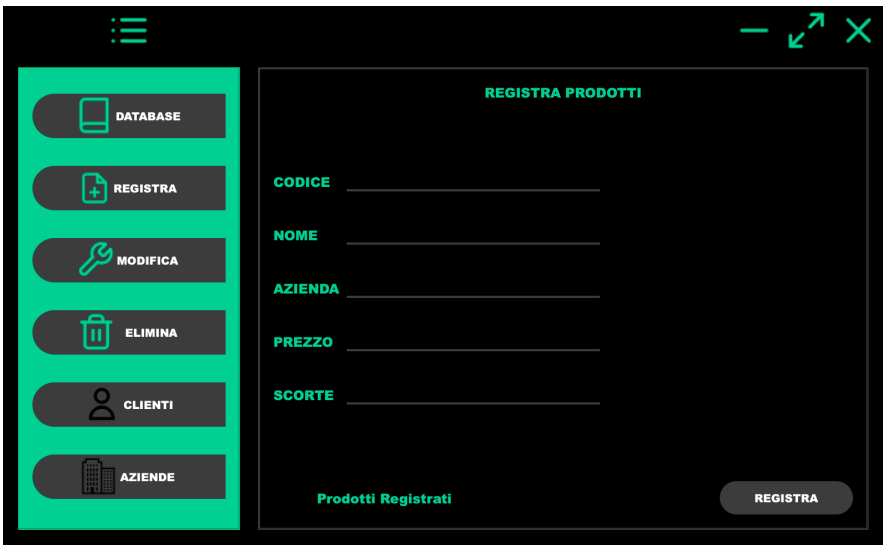

Figura 5.16: Schermata relativa ai test sull'inserimento: Registra Registrato

5.2.2 Test relativi alla modifica

Abbiamo effettuato dei test per verificare che le funzioni di modifica si comportino come previsto, con i relativi messaggi da parte del sistema all'utente nel caso in cui egli abbia eseguito una determinata funzione erroneamente o con successo.

Nel caso in cui l'utente vuole modificare un prodotto e, cercandolo, digiti male il nome, il sistema dovrà avvisare l'utente del suo errore. Invece, nel caso in cui egli abbia inserito i dati correttamente il sistema accetterà l'operazione e farà vedere i dati del prodotto rendendoli modificabili. Dopo aver modificato il prodotto correttamente e premuto il relativo tasto di modifica, il sistema dovrà avvisare l'utente che il prodotto è stato modificato con successo.

Modifica Errore

Nella Figura 5.17 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di modifica dei prodotti, nel caso in cui venga inserito il nome di un prodotto non presente nel database, dopo aver premuto il tasto `CERCA`. Il sistema, in questo caso, restituisce un messaggio di errore per comunicare all'utente che il prodotto cercato non esiste nel database del sistema.

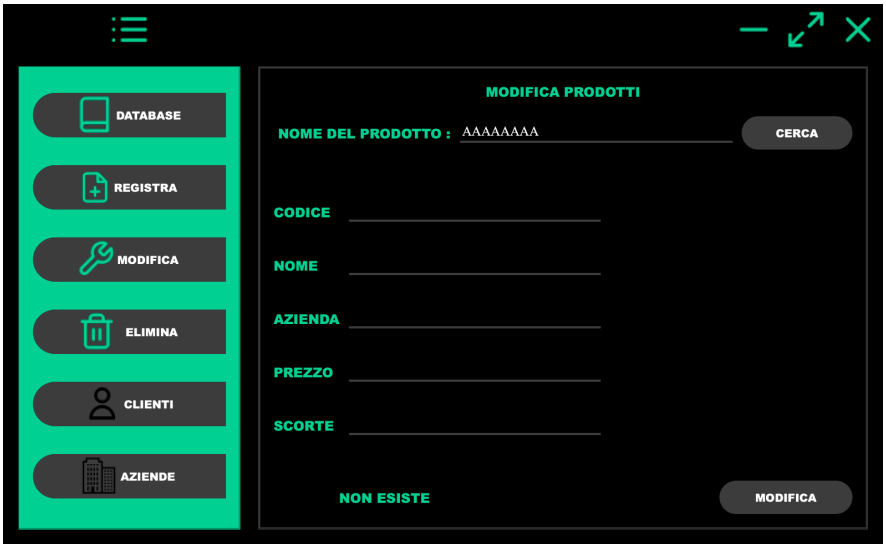

Figura 5.17: Schermata relativa ai test sulla modifica: Modifica Errore

Modifica Corretto

Nella Figura 5.18 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di modifica dei prodotti, nel caso in cui venga inserito il nome di un prodotto presente nel database, dopo aver premuto il pulsante `CERCA`.

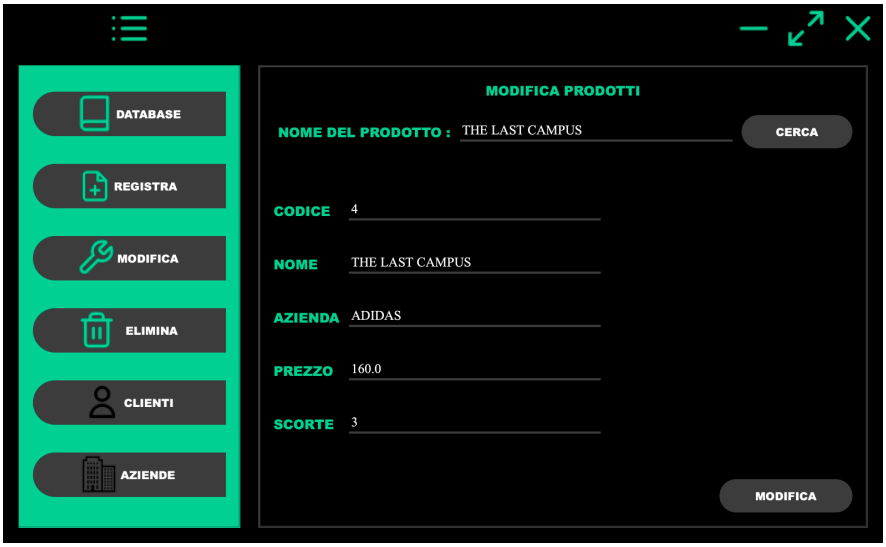

Figura 5.18: Schermata relativa ai test sulla modifica: Modifica Corretto

Modifica Modificato

Nella Figura 5.19 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di modifica dei prodotti, nel caso in cui l'utente abbia modificato i dati correttamente dopo aver premuto il pulsante `MODIFICA`. Il sistema, in questo caso, restituisce un messaggio per comunicare all'utente che il prodotto è stato modificato con successo.

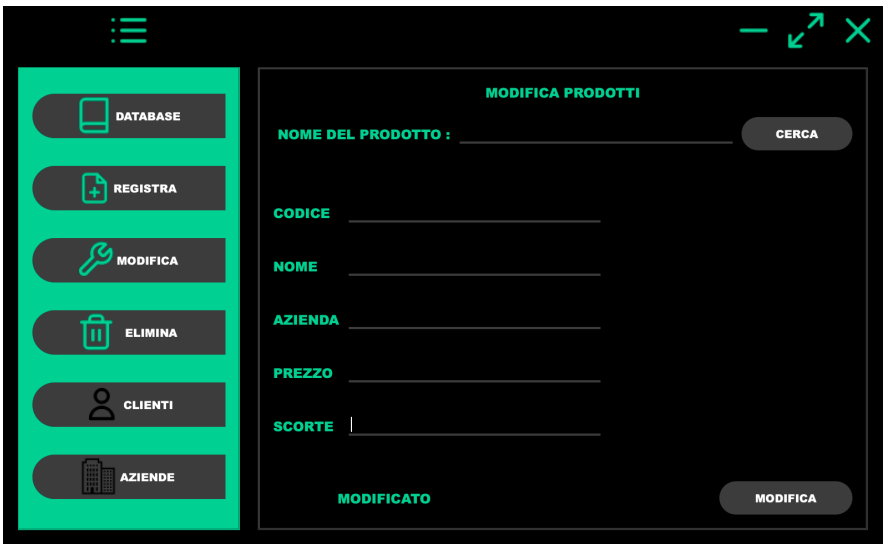

Figura 5.19: Schermata relativa ai test sulla modifica: Modifica Modificato

5.2.3 Test relativi all'eliminazione

Abbiamo effettuato dei test per verificare che le funzioni di eliminazione si comportino come previsto, con i relativi messaggi da parte del sistema all'utente nel caso in cui quest'ultimo abbia eseguito una determinata funzione erroneamente o con successo.

Nel caso in cui l'utente voglia eliminare un prodotto e, cercandolo digiti male il nome, il sistema dovrà avvisare l'utente del suo errore. Invece, nel caso in cui egli abbia inserito correttamente il nome, il sistema accetterà l'operazione e farà vedere i dati del prodotto rendendo quest'ultimo eliminabile. Dopo aver selezionato il prodotto e aver premuto il relativo tasto di eliminazione il sistema dovrà avvisare l'utente che il prodotto è stato eliminato con successo.

Elimina Errore

Nella Figura 5.20 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di eliminazione dei prodotti, nel caso in cui venga inserito il nome di un prodotto non presente nel database, dopo aver premuto il tasto `CERCA`. Il sistema, in questo caso, restituisce un messaggio di errore per comunicare all'utente che il prodotto cercato non esiste nel database.

Figura 5.20: Schermata relativa ai test sull'eliminazione: Elimina Errore

Elimina Corretto

Nella Figura 5.21 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di eliminazione dei prodotti, nel caso in cui venga inserito il nome di un prodotto presente nel database, dopo aver premuto il pulsante `CERCA`.

Figura 5.21: Schermata relativa ai test sull'eliminazione: Elimina Corretto

Elimina Eliminato

Nella Figura 5.22 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di eliminazione dei prodotti, nel caso in cui l'utente abbia selezionato il prodotto, dopo aver premuto il pulsante `ELIMINARE`. Il sistema, in questo caso, restituisce un messaggio per comunicare all'utente che il prodotto è stato eliminato con successo.

Figura 5.22: Schermata relativa ai test sull'eliminazione: Elimina Eliminato

5.2.4 Test relativi all'acquisto

Abbiamo effettuato dei test per verificare che le funzioni di acquisto si comportino come previsto, con i relativi messaggi da parte del sistema all'utente nel caso in cui quest'ultimo abbia eseguito una determinata funzione erroneamente o con successo.

Nel caso in cui l'utente voglia acquistare un prodotto e, cercandolo, digiti male il nome del prodotto o del cliente, il sistema dovrà avvisare l'utente del suo errore. Invece, nel caso in cui abbia inserito i dati correttamente, il sistema accetterà l'operazione e farà vedere i dati del prodotto e del cliente consentendo di inserire la quantità acquistata di quel prodotto da parte del cliente. Dopo aver inserito la quantità correttamente e aver premuto il tasto di acquisto, il sistema dovrà avvisare l'utente che il prodotto è stato acquistato con successo.

Acquista Errore

Nella Figura 5.23 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di acquisto, nel caso in cui venga inserito il nome di un cliente non presente nel database, dopo aver premuto il tasto `CERCA`. Il sistema, in questo caso, restituisce un messaggio di errore per comunicare all'utente che il cliente cercato non esiste nel database.

Figura 5.23: Schermata relativa ai test sull'acquisto: Acquisto Errore

Acquista Corretto

Nella Figura 5.24 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di acquisto, nel caso in cui venga inserito il nome di un prodotto e di un cliente presenti nel database e nonché una quantità, dopo aver premuto il pulsante `CERCA`.

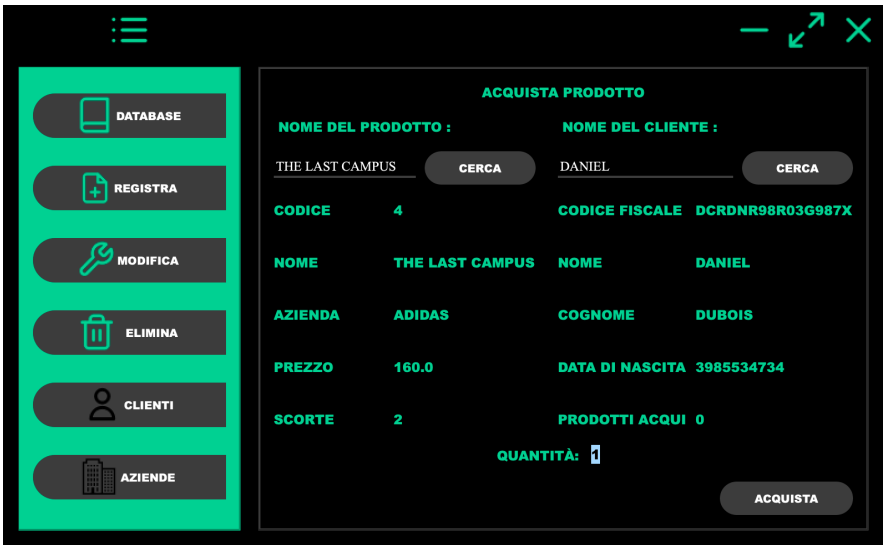

Figura 5.24: Schermata relativa ai test sull'acquisto: Acquista Corretto

Acquista Acquistato

Nella Figura 5.25 è mostrata la schermata dell'interfaccia utente relativa al test della funzione di acquisto, nel caso in cui si sia inserita correttamente la quantità, dopo aver premuto il pulsante `ACQUISTA`. Il sistema, in questo caso, restituisce un messaggio per comunicare all'utente che il prodotto è stato acquistato con successo.

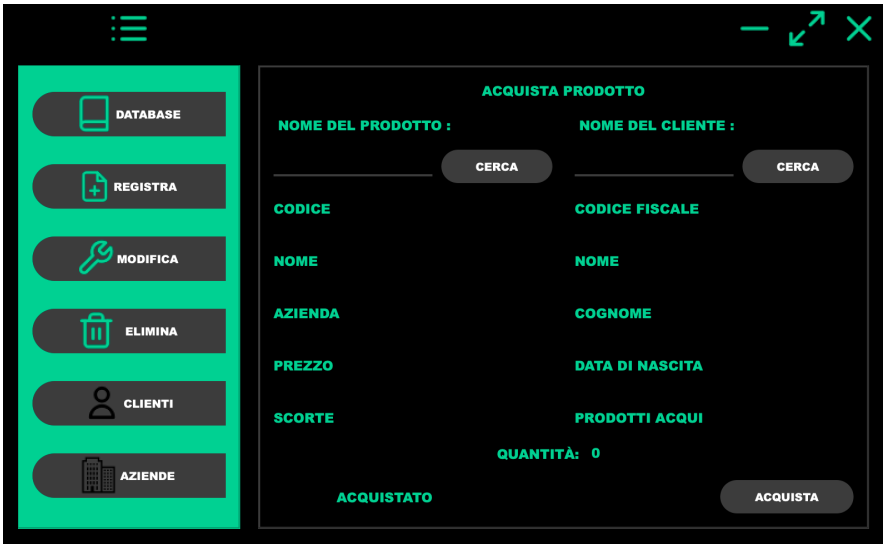

Figura 5.25: Schermata relativa ai test sull'acquisto: Acquista Acquistato

5.3 Guida all'installazione

Per installare e configurare correttamente l'applicazione di gestione del magazzino ci sono dei passaggi. In particolare, è necessario seguire le istruzioni riportate nelle seguenti sottosezioni.

5.3.1 Prerequisiti

Prima di installare l'applicazione, è necessario verificare di avere i seguenti strumenti:

- *Python3*
- *DB Browser for SQLite*
- *PyQt5*
- *Homebrew (per macOS)*

5.3.2 Github

Il codice sorgente dell'applicazione può essere scaricato dal suo repository Github. Nel Listato 5.1 viene riportato il codice bash da inserire nel terminale per clonare il repository con il codice sorgente.

```
git clone https://github.com/Daniel-Diocis/Software-Gestionale-
    Magazzino.git
```

Listing 5.1: Comando per clonare il repository

5.3.3 Installazione delle dipendenze

Dopo aver scaricato il codice sorgente da GitHub, è necessario installare le dipendenze elencate nel file `requirements.txt`. Nel Listato 5.2 viene riportato il codice bash da inserire nel terminale per installare le dipendenze.

```
pip install -r requirements.txt
```

Listing 5.2: Comando per installare le dipendenze

5.3.4 Configurazione del database

L'applicazione utilizza *DB Browser for SQLite* per la gestione e la visualizzazione del database. Quindi è possibile aprire il database dell'applicazione direttamente da DB Browser for SQLite aprendo il file contenente il database `database.db`.

5.3.5 Esecuzione dell'applicazione

Una volta che l'ambiente è stato configurato correttamente è possibile eseguire l'applicazione. Nel Listato 5.3 viene riportato il codice bash da inserire nel terminale per eseguire l'applicazione.

```
python3 main.py
```

Listing 5.3: Comando per eseguire l'applicazione

Conclusione

Nei capitoli precedenti sono state descritte la progettazione, la realizzazione e l'implementazione di un software per la gestione di un magazzino, che permetta una gestione efficiente e automatizzata dei prodotti, dei clienti, delle aziende fornitrici e degli acquisti. Abbiamo raccolto informazioni attraverso l'intervista con il cliente, dalla quale abbiamo ricavato i requisiti. Durante lo sviluppo del sistema, una delle principali sfide è stata quella di garantire un'integrazione efficace tra il front-end, realizzato con QT Designer e PyQt, e il database, gestito con DB Browser for SQLite. Grazie a un'architettura client-server, è stato possibile separare la logica di business dall'interfaccia utente, garantendo una migliore manutenibilità e flessibilità. Il sistema proposto offre funzionalità che consentono una gestione completa e automatizzata di un magazzino, migliorando l'efficienza e riducendo gli errori umani. Le funzionalità principali, come la gestione dell'inventario e la registrazione automatica delle transazioni, forniscono un controllo puntuale sui processi. I risultati ottenuti dimostrano che il sistema è in grado di gestire efficacemente il ciclo di vita dei prodotti e delle transazioni all'interno di un magazzino. I test effettuati su tutte le funzionalità principali hanno confermato l'affidabilità del sistema e la sua capacità di rispondere alle esigenze operative.

In futuro, il sistema potrebbe essere ulteriormente migliorato con l'integrazione di funzionalità avanzate, come la gestione cloud, per consentire l'accesso remoto, e la possibilità di generare report dettagliati e statistiche per l'analisi delle performance.

Bibliografia

ATZENI, P., CERI, S. e FRATERNALI, P. (2018), *Basi di dati*, McGraw-Hill Education.

BOSCAINI, M. (2017), *Imparare a programmare con Python*, Apogeo Education.

BUTTU, M. (2014), *Programmare con Python, Guida Completa*, Lswr.

DESIO, A. (2003), *Geologia applicata all'ingegneria*, Hoepli.

FITZPATRICK, M. (2020), *Create GUI Applications with Python Qt5 (PyQt5 Edition)*, Fitzpatrick.

GHEZZI, C., JAZAYERI, M. e MANDRIOLI, D. (2004), *Ingnegneria del software, fondamenti e principi*, Pearson.

HABSI, J. A., KALBANI, A. A. e ULLAH, A. (2023), «Identification of the Benefits of the Usage of Information Technology in Managing Warehouses in Supply Chain», *Journal of Economics and Management Sciences*. (Cited at page 1)

LAMBERT, K. A. (2012), *Programmazione in Python*, Apogeo.

MOORE, A. D. (2019), *Mastering GUI Programming with Python*, Packt.

PRESSMAN, R. S. (2008), *Principi di ingnegneria del software*, Hill.

REINSEL, D., GANTZ, J. e RYDNING, J. (2017), «Data Age 2025: The Evolution of Data to Life-Critical. Don't Focus on Big Data; Focus on the Data That's Big», *IDC*. (Cited at page 1)

SOMMERVILLE, I. (2017), *Ingnegneria del software*, Pearson.

Sitografia

- DB Browser for SQLite – www.sqlitebrowser.com
- Github – www.github.com
- Html – www.html.it
- Python – www.python.org
- PyQt5 – www.pypi.org/project/PyQt5
- Pytest – www.pytest.org
- PythonGUIs – www.pythonguis.com
- QT Widgets Designer Manual – https://doc.qt.io/qt-6/qtdesigner-manual.html
- Real Python – www.realpython.com
- SpringerLink – www.springerlink.com
- Stack Overlflow – www.stackoverflow.com
- W3Schools – www.w3schools.com

Ringraziamenti

Ringrazio Tommy, mia mamma e mia zia per aver sempre creduto in me e per avermi supportato in un percorso non facile.

Ringrazio mio cugino Romel, che è stato un riferimento importante per me, e il resto della mia famiglia.

Ringrazio la mia fidanzata Michela, che mi ha aiutato tanto e mi è stata vicina in ogni momento.

Vorrei ringraziare anche il mio allenatore Mattia, che mi ha insegnato tanti valori e mi ha aiutato a cambiare significativamente il mio mindset.

Infine, ringrazio i vari amici che ho conosciuto in questi ultimi due anni in università, con cui studio in polifunzionale, che mi hanno aiutato in tanti esami e hanno reso la vita universitaria più divertente e leggera.