PROGRAMMAZIONE • SEP. 16, 2024
INGEGNERIA DEL SOFTWARE • SEP. 16, 2024

2024

NEGOZIO DI ABBIGLIAMENTO

Ceriachi Chiara

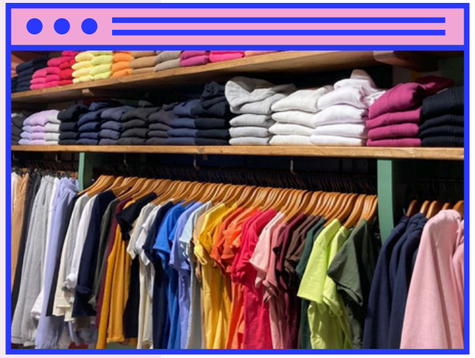
FASI DEL PROGETTO

SOFTWARE PER LA GESTIONE DELL'INVENTARIO DI UN NEGOZIO DI ABBIGLIAMENTO

1. Descrizione del progetto
2. Analisi dei requisiti
3. Diagramma dei casi d'uso
4. Matrice di mapping
5. Package di analisi
6. Diagramma di sequenza
7. Diagrammi di attività
8. Classi di progettazione
9. Macchine a stati
10. Mockup
11. Implementazione
12. Unit Testing

Obiettivi del progetto

Il progetto consiste nella realizzazione di un sistema informatico per la gestione di articoli, vendite e performance finanziarie di un negozio di abbigliamento. Il sistema offre una gestione efficiente delle operazioni fondamentali di gestione, semplificando le attività di registrazione e monitoraggio.

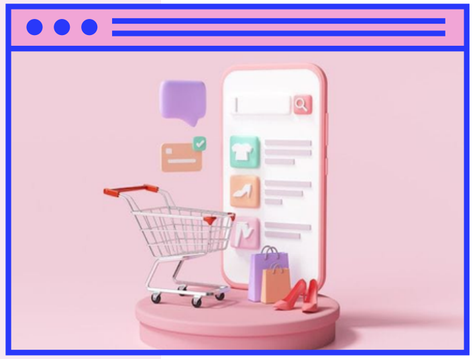


1. GESTIONE DEGLI ARTICOLI

2. GESTIONE DELLE VENDITE

3. REPORT DI VENDITA

4. SALVATAGGIO E CARICAMENTO DEI DATI



5. INTERFACCIA GRAFICA INTUITIVA

FUNZIONALITÀ PRINCIPALI:

GESTIONE ARTICOLI:

Creare, inserire, modificare e eliminare articoli

GESTIONE VENDITE:

- Registrare e tracciare le transazioni di vendita
- Gestione flessibile delle date di vendita
- Analizzare i dati di vendita ordinandoli cronologicamente

ANALISI DATI:

- Visualizzare e analizzare i dati di vendita
- Tracciare i dati economici e le statistiche di vendita
- Generare report e riassunti dei dati di vendita

REQUISITI PROGETTUALI

- Linguaggio di Programmazione: Python
- Framework GUI: Tkinter

FUNZIONALITÀ PRINCIPALI:

STAMPA:

- Stampare l'elenco degli articoli
- Stampare gli articoli venduti
- Stampare report di vendita e analisi grafica delle statistiche di vendita annuali

ARCHIVIAZIONE DATI:

- Archiviazione dati sicura e affidabile
- Estrazione e salvataggio dei dati su file di testo

GUI:

- Interfaccia intuitiva e user-friendly
- Navigazione facile e accesso immediato alle varie funzionalità

REQUISITI PROGETTUALI

- Linguaggio di Programmazione: Python
- Framework GUI: Tkinter

DIAGRAMMA DI CLASSI

NEGOZIO

VENDITE

ARTICOLI

ARTICOLO

DATE

FILEMANAGER

Vendite

elenco: Articoli
numerovendite: int
capiacquistati: List[Articolo]
datevendita: List[Date]

+ __init__()
+ venditaarticolo(a: Articolo, x: int): int
+ incassimensili(a: int, m: int): float
+ venditemensili(a: int, m: int): int
+ get_numerovendite(): int
+ get_elementodalistacapiacquistati(index: int): Articolo
+ get_data(index: int): Date
+ estrai_data(index: int): Tuple[Optional[int], Optional[int], Optional[int]]
+ add_data(g: Date): int

Negozio

mioarticolo: Articolo
giornovendita: Date
file_manager: FileManager
elenco: Articoli
magazzino: Vendite
articoli_caricati: List[Articolo]

+ __init__()
+ stampa_elencoarticoli(output: scrolledtext)
+ stampa_listacapiacquistati(output: scrolledtext)
+ aggiungi_articolo()
+ elimina_articolo()
+ modifica_articolo()
+ vendi_articolo()
+ calcola_incassi()
+ ordina_articoli_venduti()
+ report()
+ salva_su_file()
+ salva_vendite_su_file()
+ pulisci_output()
+ reset_campi()
+ stampa_elenco_articoli()
+ stampa_articoli_acquistati()
+ main()

Articoli

elencoarticoli: List[Articolo]
n: int
MAX: int = 1000

+ __init__()
+ get_n(): int
+ add_elemento(a: Articolo): int
+ get_elementodaelencoarticoli(p: int): Articolo
+ rimuoviarticolo(x: int): None

Date

date: List[int]

+ __init__(day: int = 1, month: int = 1, year: int = 1900)
+ is_valid(): bool
+ get_day(): int
+ set_day(day: int): None
+ get_month(): int
+ set_month(month: int): None
+ get_year() -> int
+ set_year(year: int): None
+ verifica_data(day: int, month: int, year: int): bool
+ bisestile(year: int): bool

FileManager

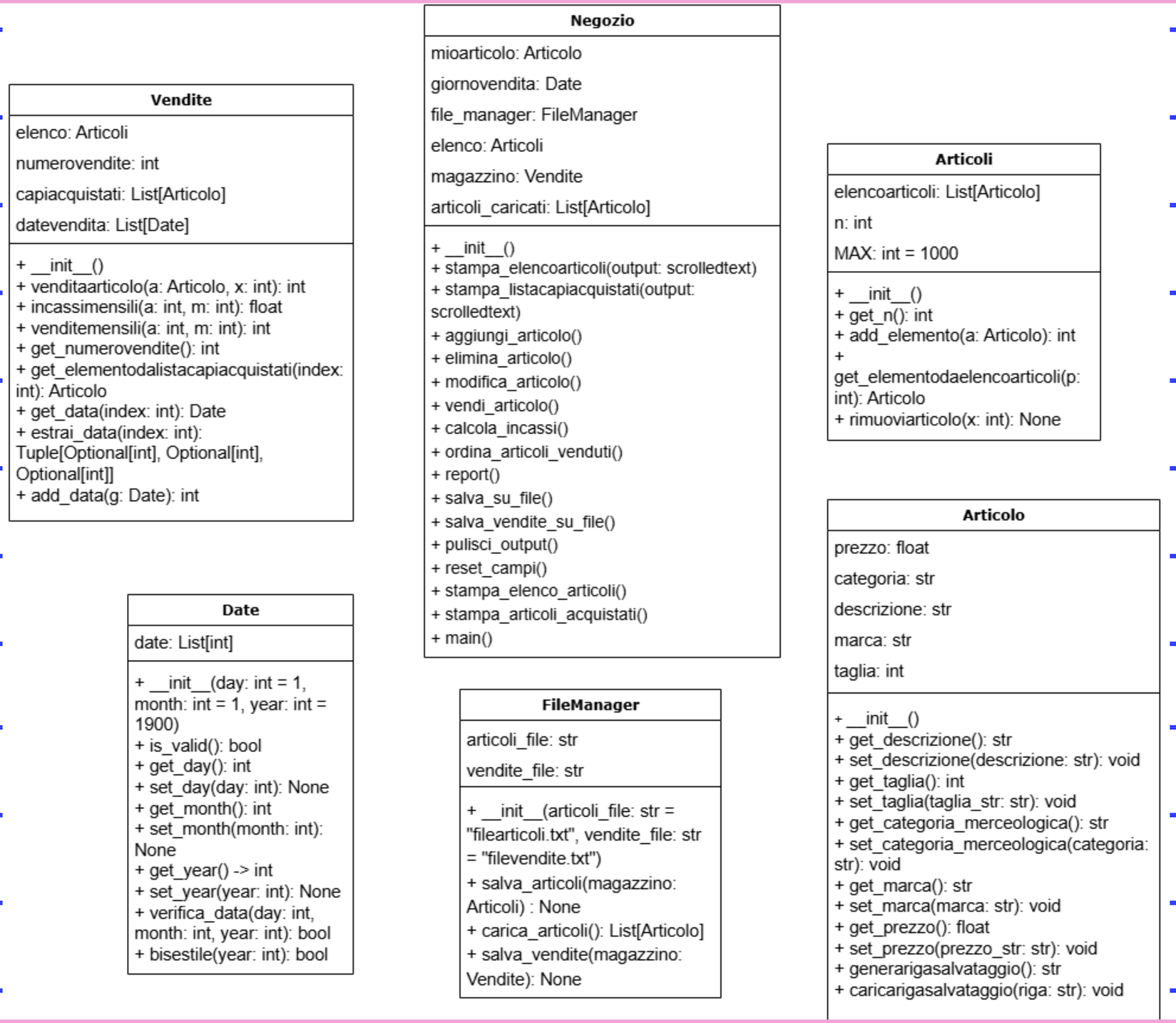
articoli_file: str
vendite_file: str

+ __init__(articoli_file: str = "filearticoli.txt", vendite_file: str = "filevendite.txt")
+ salva_articoli(magazzino: Articoli) : None
+ carica_articoli(): List[Articolo]
+ salva_vendite(magazzino: Vendite): None

Articolo

prezzo: float
categoria: str
descrizione: str
marca: str
taglia: int

+ __init__()
+ get_descrizione(): str
+ set_descrizione(descrizione: str): void
+ get_taglia(): int
+ set_taglia(taglia_str: str): void
+ get_categoria_merceologica(): str
+ set_categoria_merceologica(categoria: str): void
+ get_marca(): str
+ set_marca(marca: str): void
+ get_prezzo(): float
+ set_prezzo(prezzo_str: str): void
+ generarigasalvataggio(): str
+ caricarigasalvataggio(riga: str): void


Vendite
elenco: Articoli
numerovendite: int
capiacquistati: List[Articolo]
datevendita: List[Date]
+ __init__()
+ venditaarticolo(a: Articolo, x: int): int
+ incassimensili(a: int, m: int): float
+ venditemensili(a: int, m: int): int
+ get_numerovendite(): int
+ get_elementodalistacapiacquistati(index: int): Articolo
+ get_data(index: int): Date
+ estrai_data(index: int): Tuple[Optional[int], Optional[int], Optional[int]]
+ add_data(g: Date): int
Negozio
mioarticolo: Articolo
giornovendita: Date
file_manager: FileManager
elenco: Articoli
magazzino: Vendite
articoli_caricati: List[Articolo]
+ __init__()
+ stampa_elencoarticoli(output: scrolledtext)
+ stampa_listacapiacquistati(output: scrolledtext)
+ aggiungi_articolo()
+ elimina_articolo()
+ modifica_articolo()
+ vendi_articolo()
+ calcola_incassi()
+ ordina_articoli_venduti()
+ report()
+ salva_su_file()
+ salva_vendite_su_file()
+ pulisci_output()
+ reset_campi()
+ stampa_elenco_articoli()
+ stampa_articoli_acquistati()
+ main()
Articoli
elencoarticoli: List[Articolo]
n: int
MAX: int = 1000
+ __init__()
+ get_n(): int
+ add_elemento(a: Articolo): int
+ get_elementodaelencoarticoli(p: int): Articolo
+ rimuoviarticolo(x: int): None
Date
date: List[int]
+ __init__(day: int = 1, month: int = 1, year: int = 1900)
+ is_valid(): bool
+ get_day(): int
+ set_day(day: int): None
+ get_month(): int
+ set_month(month: int): None
+ get_year() -> int
+ set_year(year: int): None
+ verifica_data(day: int, month: int, year: int): bool
+ bisestile(year: int): bool
FileManager
articoli_file: str
vendite_file: str
+ __init__(articoli_file: str = "filearticoli.txt", vendite_file: str = "filevendite.txt")
+ salva_articoli(magazzino: Articoli) : None
+ carica_articoli(): List[Articolo]
+ salva_vendite(magazzino: Vendite): None
Articolo
prezzo: float
categoria: str
descrizione: str
marca: str
taglia: int
+ __init__()
+ get_descrizione(): str
+ set_descrizione(descrizione: str): void
+ get_taglia(): int
+ set_taglia(taglia_str: str): void
+ get_categoria_merceologica(): str
+ set_categoria_merceologica(categoria: str): void
+ get_marca(): str
+ set_marca(marca: str): void
+ get_prezzo(): float
+ set_prezzo(prezzo_str: str): void
+ generarigasalvataggio(): str
+ caricarigasalvataggio(riga: str): void

DIAGRAMMI UML

CASO D'USO:

AGGIUNTA ARTICOLO

CASO D'USO: AGGIUNGI_ARTICOLO

DESCRIZIONE:

Il caso d'uso consente agli utenti di aggiungere nuovi articoli al magazzino. Viene verificata la validità dei dati immessi (quantità, capacità massima, ecc.) e, in caso di successo, gli articoli vengono salvati nel magazzino.

ATTORI PRINCIPALI:

Utente che gestisce il magazzino.

FLUSSO PRINCIPALE DEGLI EVENTI --->

PRECONDIZIONI:

Il sistema deve essere attivo e il magazzino deve essere accessibile.
La capacità massima del magazzino non deve essere superata.

POSTCONDIZIONI:

Gli articoli sono stati aggiunti correttamente al magazzino, o è stato mostrato un messaggio di errore se i dati erano non validi o la capacità era superata.

```
def aggiungi_articolo(self):
    self.output.delete( index1: 1.0, tk.END)
    try:
        prezzo_str = self.entry_prezzo.get().strip()
        marca = self.entry_marca.get().strip()
        taglia = self.entry_taglia.get()
        categoria = self.entry_categoria.get().strip()
        descrizione = self.entry_descrizione.get().strip()
        qty = self.entry_qty.get()
        if not qty:
            raise ValueError("La quantità non può essere vuota.")
        try:
            qty = int(self.entry_qty.get())
        except ValueError:
            raise ValueError("La quantità deve essere un numero intero val
        if qty <= 0:
            raise ValueError("La quantità deve essere maggiore di zero.")
        if self.magazzino.get_n() + qty > Articoli.MAX:
            raise ValueError(f"Non è possibile aggiungere {qty} articoli.
        else:
            for _ in range(qty):
                mioarticolo = Articolo()
                mioarticolo.set_prezzo(prezzo_str)
                mioarticolo.set_categoria_merceologica(categoria)
                mioarticolo.set_taglia(taglia)
                if self.magazzino.add_elemento(mioarticolo) != 0:
                    break
            messagebox.showinfo( title: "Successo", message: "Articoli agg
            self.reset_campi()
```

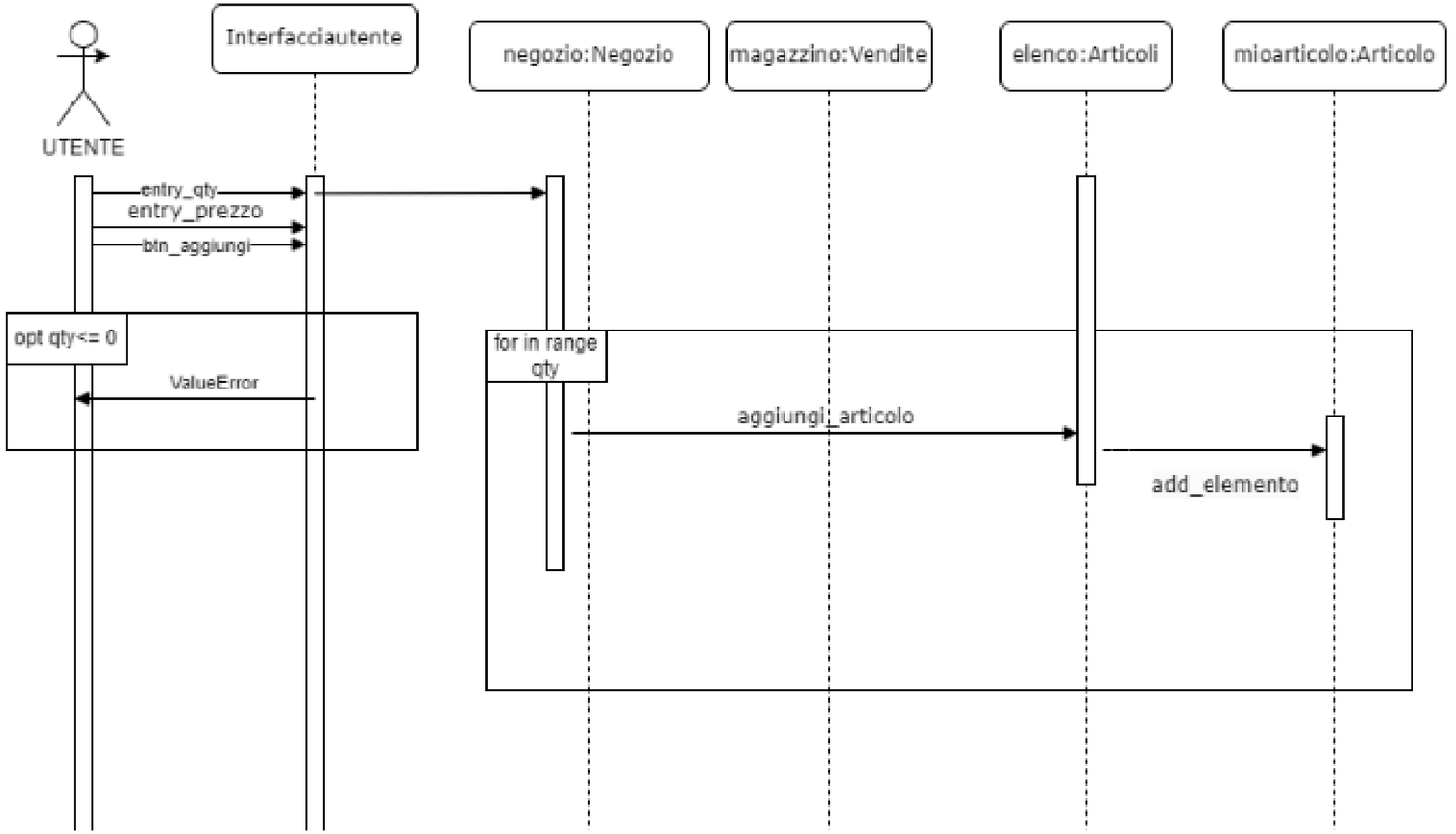
CASO D'USO: AGGIUNTA ARTICOLO

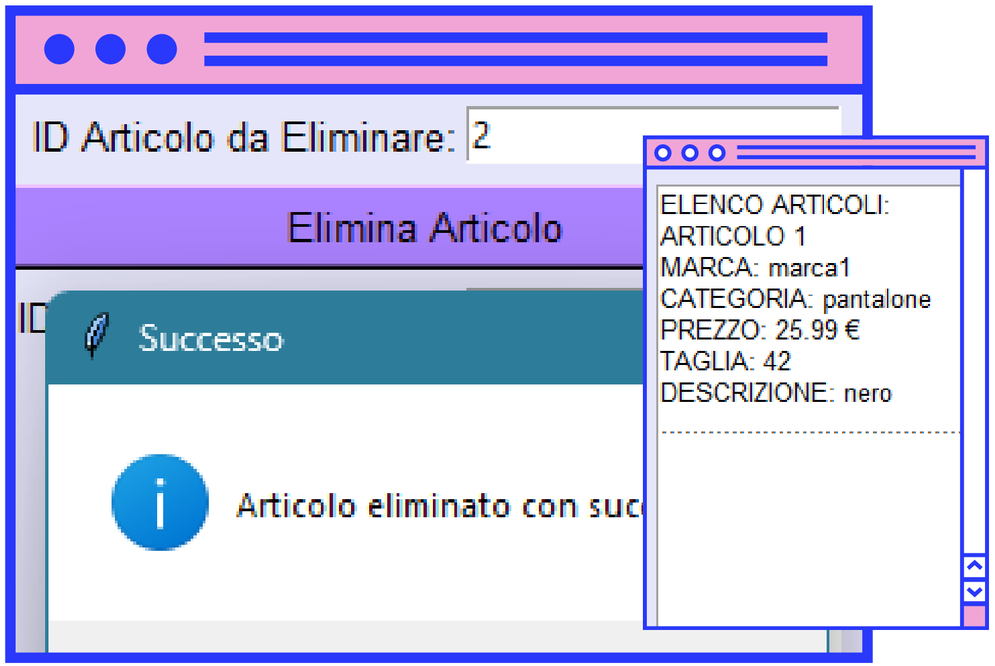
FLUSSO PRINCIPALE DEGLI EVENTI

- Avvio Funzione di Aggiunta Articolo: l'utente avvia il processo per aggiungere nuovi articoli al magazzino.
- Inserimento Dati: l'utente compila i campi richiesti, tra cui: Prezzo, Marca, Taglia, Categoria, Descrizione, Quantità
- Validazione Dati: se la quantità è vuota o non è un numero intero valido, se è $\leqslant 0$ o se supera la capacità massima del magazzino, viene mostrato un messaggio di errore.
- Creazione e Salvataggio degli Articoli: per ogni articolo nella quantità specificata:
  - Viene creato un nuovo articolo con i dati forniti.
  - L'articolo viene aggiunto al magazzino.
  - Se il magazzino è pieno, il processo si interrompe e viene mostrato un messaggio di errore.
- Conferma Operazione: se gli articoli vengono aggiunti con successo, viene mostrato un messaggio di conferma.
- Reset dei Campi.

DIAGRAMMA

AGGIUNTA ARTICOLO





CASO D'USO:

ELIMINA ARTICOLO

CASO D'USO: ELIMINA_ARTICOLO

DESCRIZIONE:

Il caso d'uso consente agli utenti di eliminare un articolo specifico dal magazzino fornendo il relativo ID. Viene verificata la validità dell'ID immesso e, se l'articolo esiste, viene rimosso con successo.

ATTORI PRINCIPALI:

Utente che gestisce il magazzino.

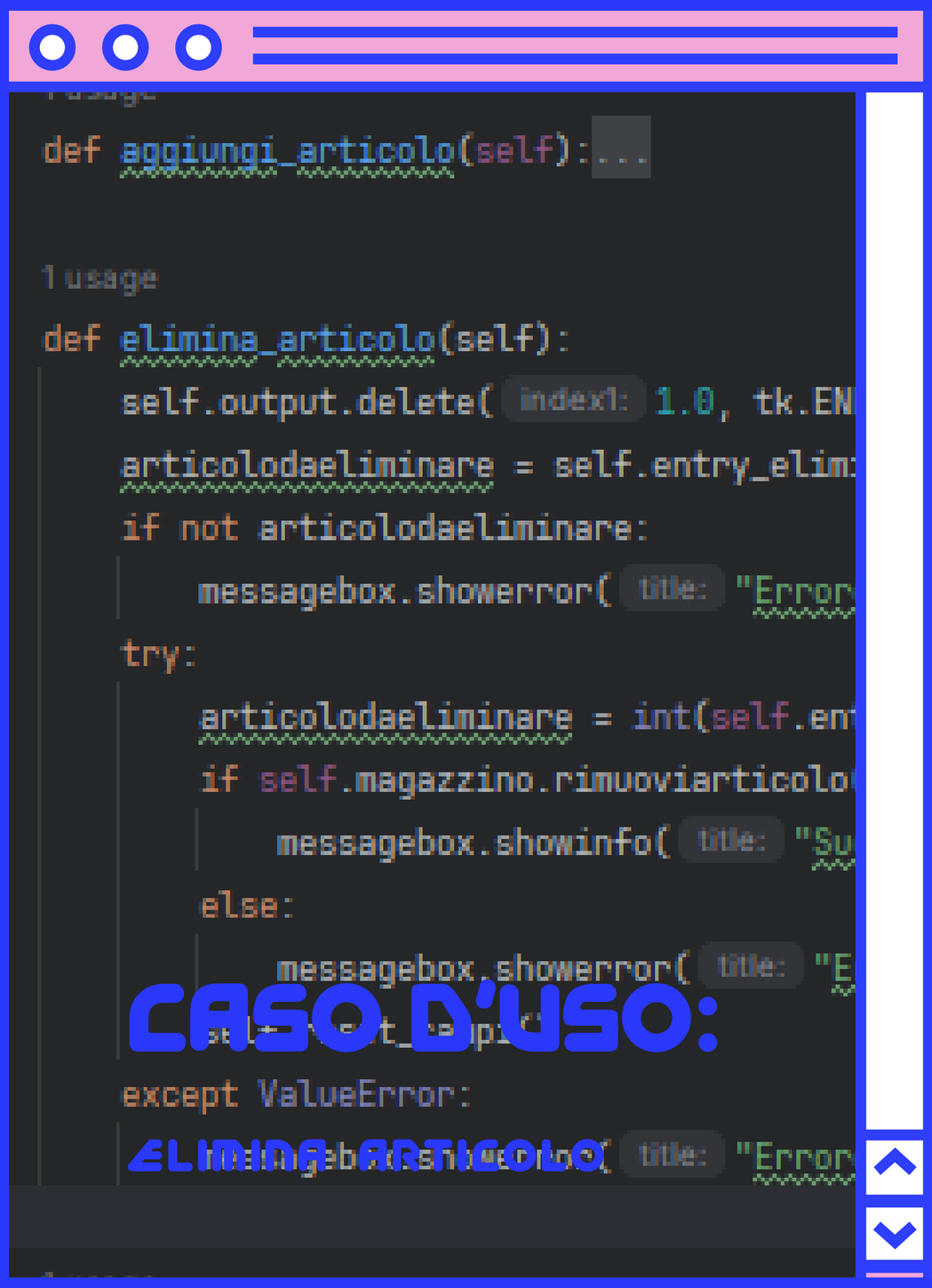
FLUSSO PRINCIPALE DEGLI EVENTI --->

PRECONDIZIONI:

Il sistema deve essere attivo e il magazzino accessibile.
Deve essere presente almeno un articolo nel magazzino.

POSTCONDIZIONI:

L'articolo specificato è stato eliminato dal magazzino, o è stato mostrato un messaggio di errore se i dati erano non validi o l'ID non corrispondeva ad alcun articolo.



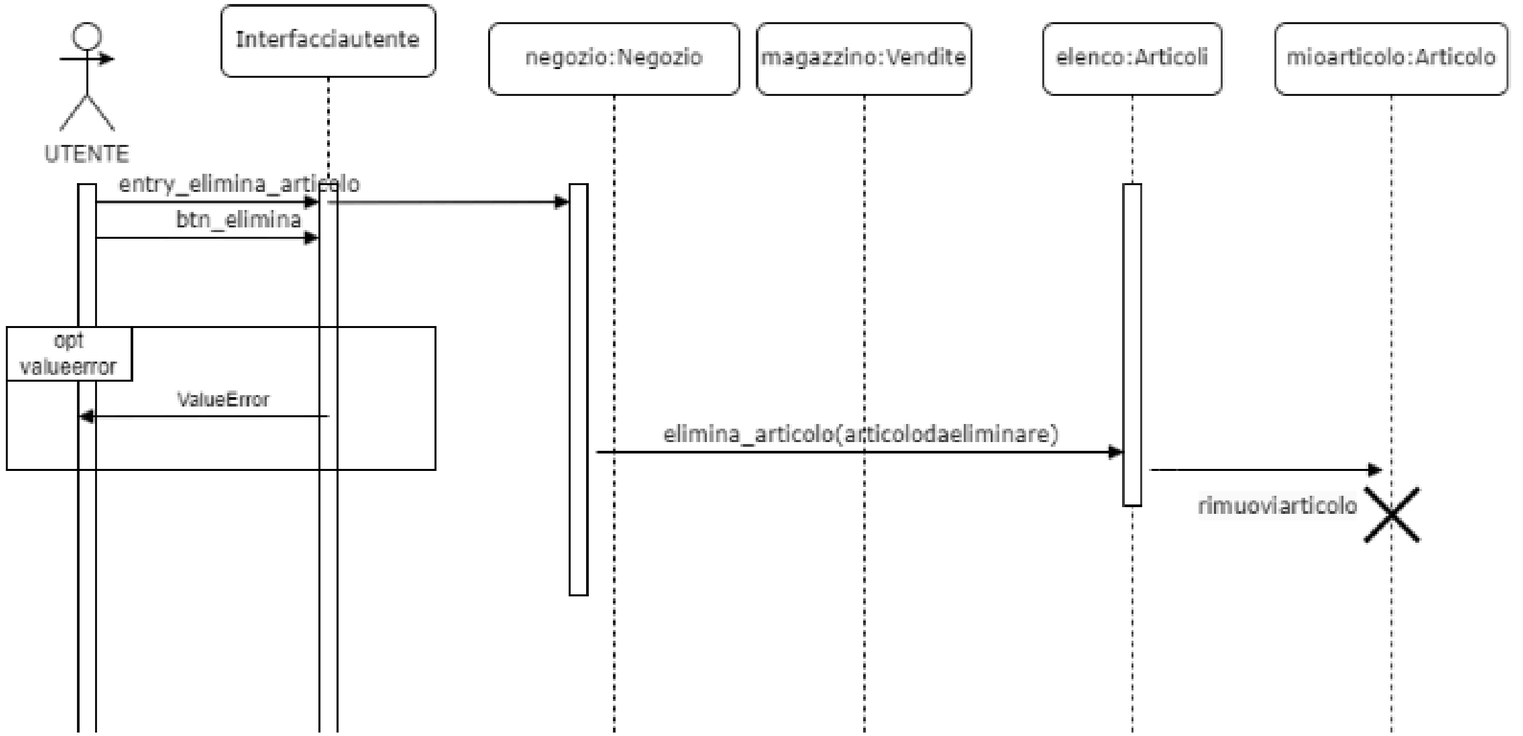
CASO D'USO: ELIMINA ARTICOLO

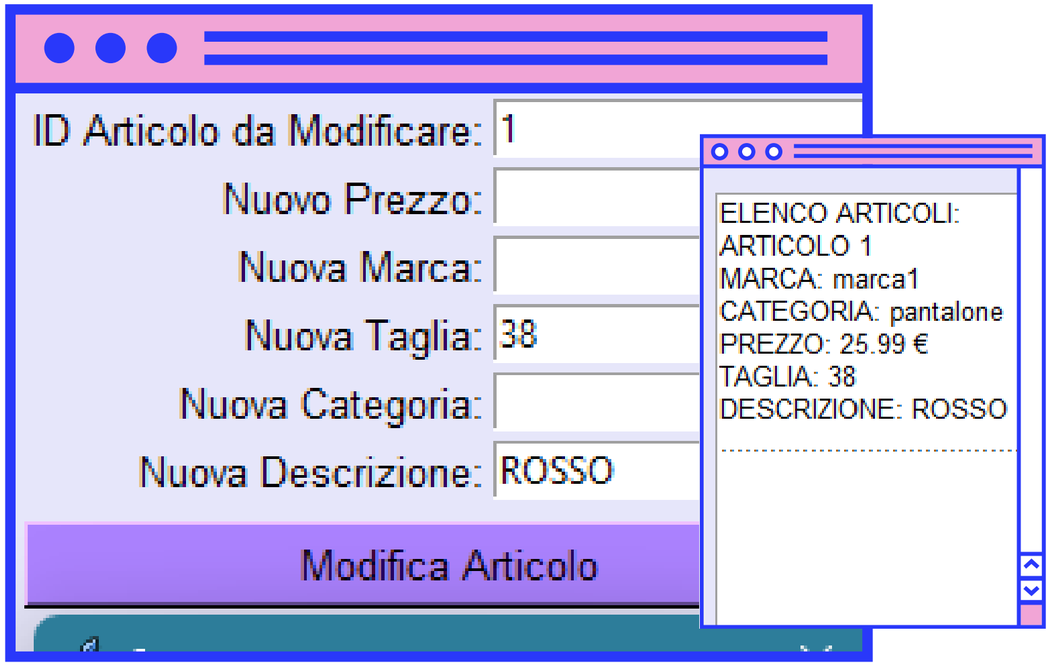
FLUSSO PRINCIPALE DEGLI EVENTI

- Avvio Funzione di Eliminazione Articolo: l'utente avvia il processo per eliminare un articolo dal magazzino.
- Inserimento Dati: l'utente specifica l'ID dell'articolo da eliminare.
- Validazione Dati: se il campo ID articolo è vuoto, non corrisponde ad un aericolo esistente o non è valido, viene mostrato un messaggio di errore.
- Eliminazione dell'Articolo: l'articolo corrispondente all'ID inserito viene rimosso dal magazzino. Se l'operazione ha successo, viene mostrato un messaggio di conferma.
- Se non è possibile eliminare l'articolo (ad esempio, ID non valido), viene mostrato un messaggio di errore.
- Reset dei Campi.

DIAGRAMMA

ELIMINA ARTICOLO





CASO D'USO:

MODIFICA ARTICOLO

CASO D'USO: MODIFICA_ARTICOLO

DESCRIZIONE:

Il caso d'uso consente agli utenti di modificare i dettagli di un articolo esistente nel magazzino fornendo il relativo ID. Viene verificata la validità dell'ID immesso, e i campi modificati vengono aggiornati nell'articolo selezionato.

ATTORI PRINCIPALI:

Utente che gestisce il magazzino.

FLUSSO PRINCIPALE DEGLI EVENTI --->

PRECONDIZIONI:

Il sistema deve essere attivo e il magazzino accessibile. Deve essere presente almeno un articolo nel magazzino. L'ID articolo deve corrispondere a un articolo esistente.

POSTCONDIZIONI:

I dati dell'articolo sono stati aggiornati correttamente, o è stato mostrato un messaggio di errore se i dati erano non validi o l'ID non corrispondeva ad alcun articolo.

```
def modifica_articolo(self):
    self.output.delete( index1: 1.0, tk.END)
    articolo_id = self.entry_aggiorna_id.get()
    if not articolo_id:
        messagebox.showerror( title: "Errore",  message: "L'ID
    try:
        articolo_id = int(self.entry_aggiorna_id.get()) - 1
        if 0 <= articolo_id < self.magazzino.get_n():
            articolo = self.magazzino.get_elementodaelencoart
            if articolo:
                if self.entry_aggiorna_prezzo.get():
                    articolo.set_prezzo(self.entry_aggiorna_p
                if self.entry_aggiorna_marca.get():
                    articolo.set_marca(self.entry_aggiorna_ma
                if self.entry_aggiorna_taglia.get():
                    articolo.set_taglia(self.entry_aggiorna_t
                if self.entry_aggiorna_categoria.get():
                    articolo.set_categoria_merceologica(self.
                if self.entry_aggiorna_descrizione.get():
                    articolo.set_descrizione(self.entry_aggio
                m                                                    messag
                s
            else:
                messagebox.showerror( title: "Errore",  message
        else:
            messagebox.showerror( title: "Errore",  message: "I
    except ValueError:
```

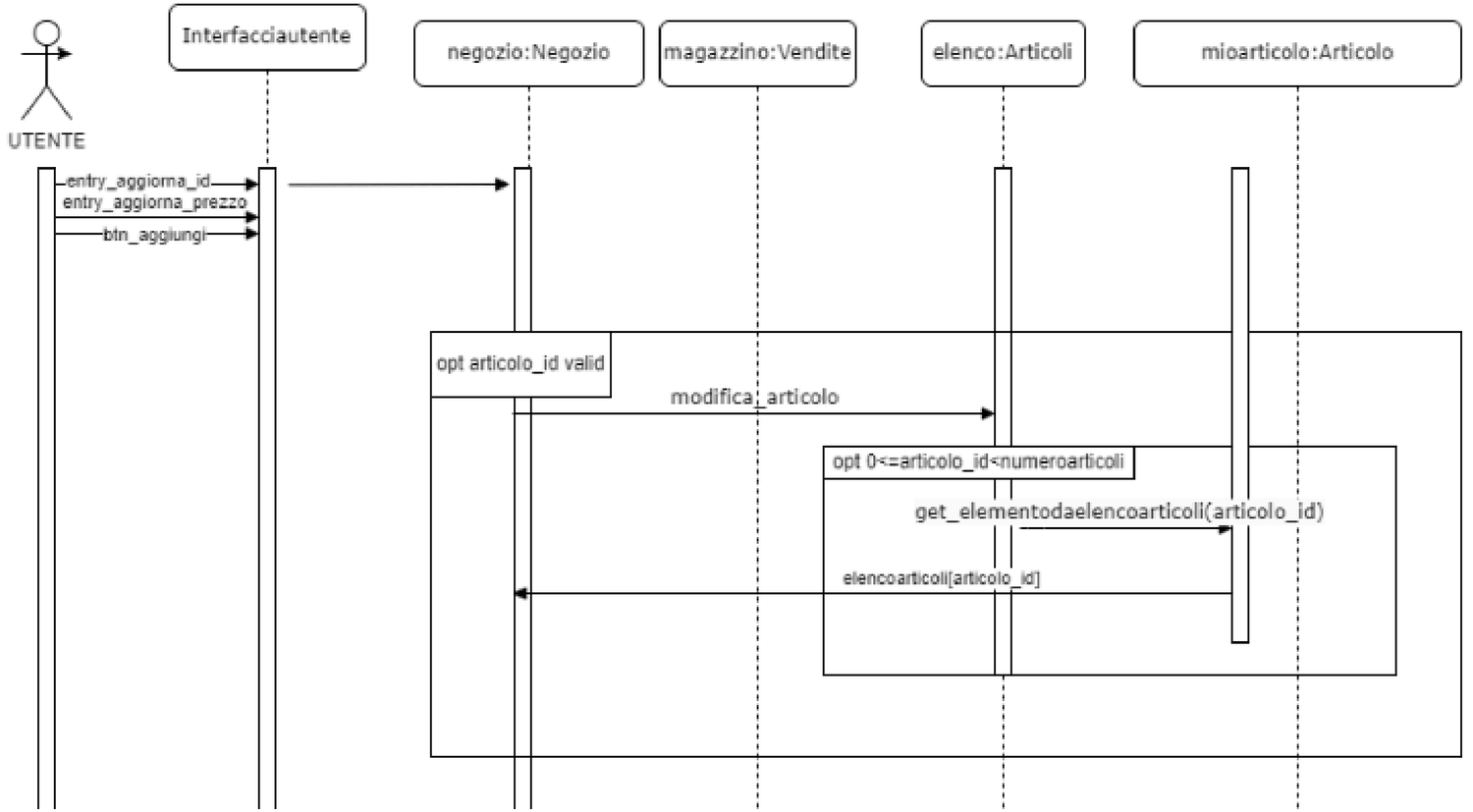
CASO D'USO: MODIFICA ARTICOLO

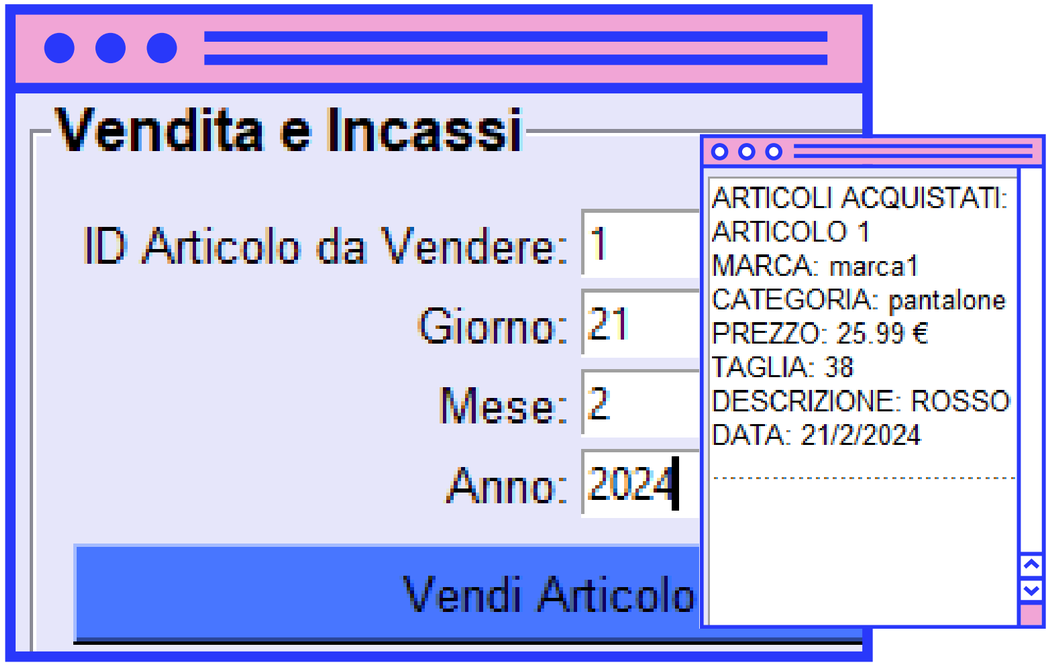
FLUSSO PRINCIPALE DEGLI EVENTI

- Avvio Funzione di Modifica Articolo: l'utente avvia il processo per modificare i dettagli di un articolo nel magazzino.
- Inserimento Dati. L'utente specifica: l'ID dell'articolo da modificare e uno o più campi da aggiornare (prezzo, marca, taglia...)
- Validazione Dati: Se il campo ID articolo è vuoto invalido o corrisponde ad un articolo inesistente, viene mostrato un messaggio di errore.
- Modifica dell'Articolo: L'articolo corrispondente all'ID viene recuperato. Per ogni campo modificato, il sistema aggiorna il valore corrispondente nell'articolo. Se l'operazione ha successo, viene mostrato un messaggio di conferma.
- Reset dei Campi.

DIAGRAMMA

MODIFICA ARTICOLO





CASO D'USO:

VENDITA ARTICOLO

CASO D'USO: VENDI_ARTICOLO

DESCRIZIONE:

Il caso d'uso consente agli utenti di registrare la vendita di un articolo specifico presente nel magazzino. È necessario specificare l'ID dell'articolo da vendere e la data di vendita. Dopo la vendita, l'articolo viene rimosso dal magazzino.

ATTORI PRINCIPALI:

Utente che gestisce il magazzino.

FLUSSO PRINCIPALE DEGLI EVENTI --->

PRECONDIZIONI:

Il sistema deve essere attivo e il magazzino accessibile. Deve essere presente almeno un articolo nel magazzino. L'ID articolo deve corrispondere a un articolo esistente.

POSTCONDIZIONI:

La vendita dell'articolo è stata registrata correttamente, la data di vendita è stata aggiunta e l'articolo è stato rimosso dal magazzino, oppure è stato mostrato un messaggio di errore in caso di dati non validi.

```
def vendi_articolo(self):
    self.output.delete( index1: 1.0, tk.END)
    try:
        articolovenduto_index = int(self.entry
        if 0 <= articolovenduto_index < self.m
            gg = int(self.entry_giorno.get())
            mm = int(self.entry_mese.get())
            aaaa = int(self.entry_anno.get())
            giornovendita = Date(gg, mm, aaaa)
            if not giornovendita.is_valid():
                messagebox.showerror( title: "Er
                return
            self.magazzino.venditaarticolo(sel
            self.magazzino.add_data(giornovend
            self.magazzino.rimuoviarticolo(sel
            messagebox.showinfo( title: "Success
        else:
            messagebox.showerror( title: "Errore
        self.reset_campi()
    except ValueError:
```

CASO D'USO:

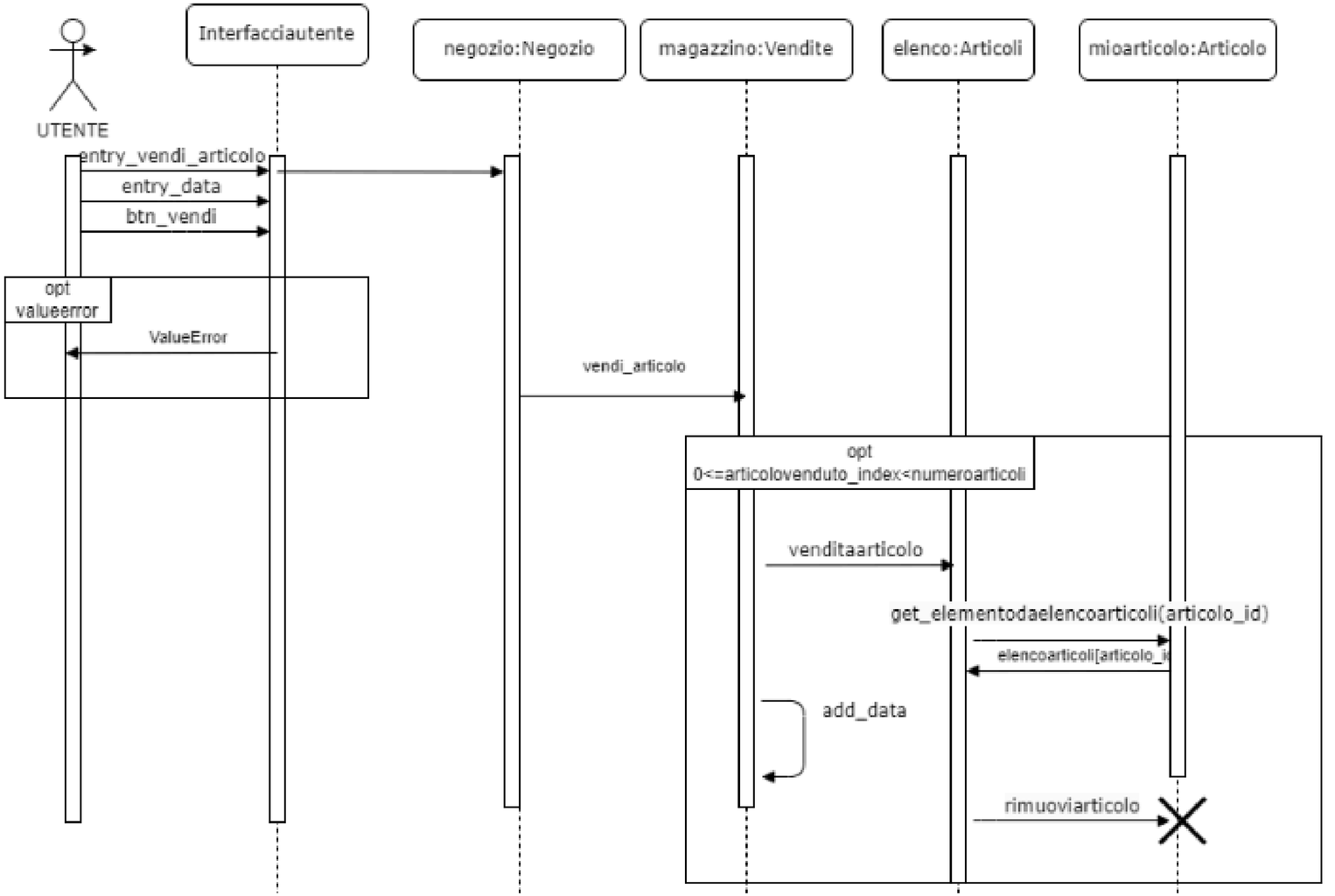
VENDITA ARTICOLO

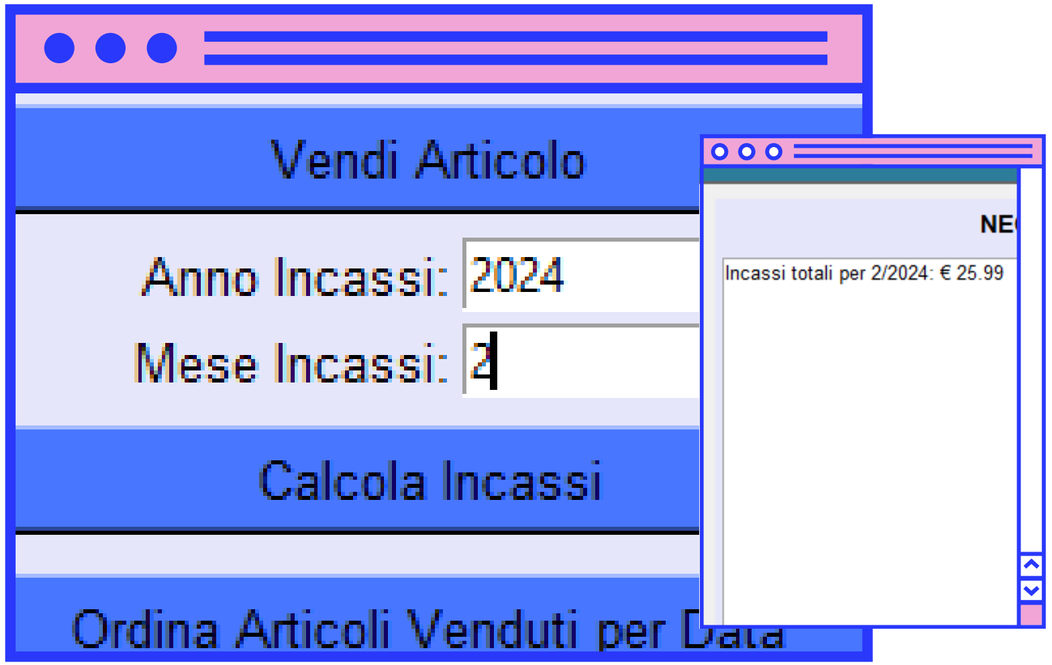
FLUSSO PRINCIPALE DEGLI EVENTI

- Avvio Funzione di Vendita Articolo: l'utente avvia il processo per registrare la vendita di un articolo.
- Inserimento Dati. Ll'utente specifica: l'ID dell'articolo da vendere, la data di vendita (giorno, mese, anno).
- Validazione Dati: se l'ID articolo è vuoto o non valido (non è un numero intero o è fuori dai limiti del magazzino), viene mostrato un messaggio di errore. Se la data di vendita non è valida (ad esempio, il formato non è corretto o la data è inesistente), viene mostrato un messaggio di errore.
- Elaborazione della Vendita:
  - L'articolo corrispondente all'ID viene identificato e copiato sulla lista contenente gli articoli venduti.
  - La data di vendita viene validata e registrata.
  - L'articolo viene rimosso dal magazzino.
- Se l'operazione ha successo, viene mostrato un messaggio di conferma.
- Reset dei Campi.

DIAGRAMMA

VENDITA ARTICOLO





CASO D'USO:

CALCOLO INCASSI

CASO D'USO: CALCOLO_INCASSI

DESCRIZIONE:

Il caso d'uso consente agli utenti di calcolare gli incassi totali registrati per un determinato mese e anno. L'utente inserisce i dati necessari, e il sistema restituisce il risultato.

ATTORI PRINCIPALI:

Utente che gestisce il magazzino.

FLUSSO PRINCIPALE DEGLI EVENTI --->

PRECONDIZIONI:

Il sistema deve essere attivo e il magazzino accessibile. Devono essere presenti dati relativi a vendite registrate per il periodo specificato.

POSTCONDIZIONI:

Gli incassi totali per il periodo specificato sono stati calcolati e mostrati all'utente, oppure è stato mostrato un messaggio di errore in caso di dati non validi.

```
            self.reset_campi()
        except ValueError:
            messagebox.showerror( title: "Errore",  message:

1 usage
def calcola_incassi(self):
    self.output.delete( index1: 1.0, tk.END)
    try:
        anno = int(self.entry_anno_incassi.get())
        mese = int(self.entry_mese_incassi.get())
        incassi = (self.magazzino.incassimensili(anno,
        self.output.insert(tk.END,  chars: f"Incassi to
        self.reset_campi()
    except ValueError as ve:
        messagebox.showerror( title: "Errore",  message:

1 usage
def ordina_articoli_venduti(self):...

1 usag
def
    self.output.delete( index1: 1.0, tk.END)
    try:
        a = int(self.entry_anno_report.get())
        venditeannuali = 0
```

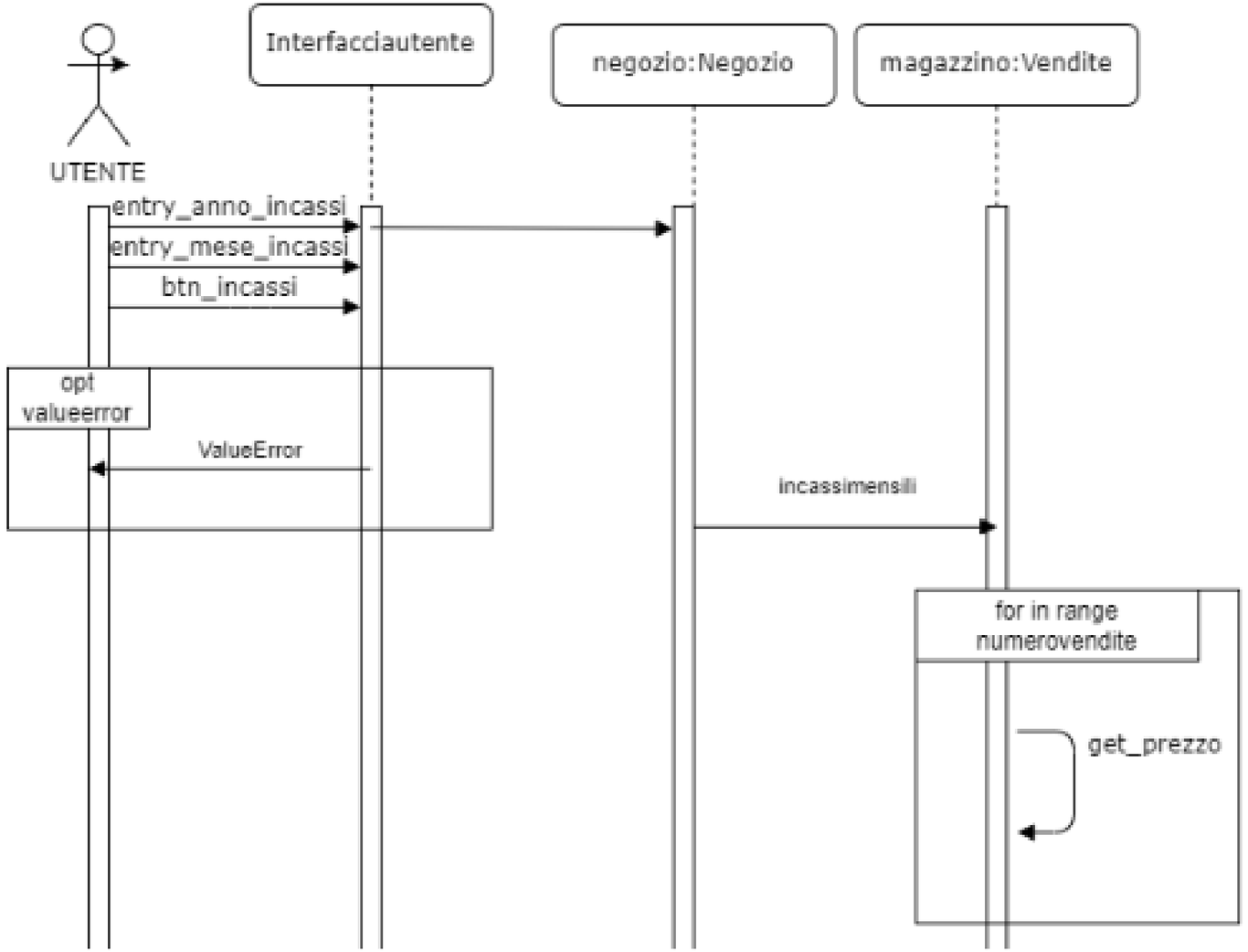
CASO D'USO: CALCOLO INCASSI

FLUSSO PRINCIPALE DEGLI EVENTI

- Avvio Funzione di Calcolo Incassi: l'utente avvia il processo per calcolare gli incassi totali di un mese e anno specifici.
- Inserimento Dati. L'utente specifica: l'anno e il mese.
- Validazione Dati: se i dati inseriti sono vuoti o non validi (ad esempio, il mese non è compreso tra 1 e 12 o l'anno non è un numero intero), viene mostrato un messaggio di errore.
- Calcolo degli Incassi: se i dati sono validi, il sistema calcola gli incassi totali per il mese e l'anno specificati.
- Il risultato viene mostrato all'utente nel formato: “Incassi totali per Mese/Anno: € <importo>.”
- Reset dei Campi.

DIAGRAMMA

CALCOLO INCASSI



ORDINA ARTICOLI VENDUTI PER DATA

IL METODO ORDINA_ARTICOLI_VENDUTI ORDINA GLI ARTICOLI VENDUTI IN BASE ALLA DATA DI VENDITA UTILIZZANDO LA FUNZIONE SORTED. IN QUESTO CASO, SI CREA UNA LISTA DI TUPLE CHE ASSOCIA OGNI ARTICOLO AI RISPETTIVI INDICI DI VENDITA, E SI UTILIZZA UNA CHIAVE DI ORDINAMENTO CHE COMBINA L'ANNO, IL MESE E IL GIORNO DELLA DATA DI VENDITA IN FORMATO STRINGA.

REPORT ANNUALE

IL METODO REPORT GENERA UN REPORT ANNUALE SU VENDITE E INCASSI PER L'ANNO FORNITO. CALCOLA E VISUALIZZA PER OGNI MESE IL NUMERO DI VENDITE, GLI INCASSI E IL PREZZO MEDIO PER ARTICOLO VENDUTO, ACCUMULANDO I TOTALI ANNUALI. ALLA FINE, MOSTRA IL PREZZO MEDIO ANNUALE PER ARTICOLO, IL GUADAGNO MENSILE MEDIO E IL GUADAGNO TOTALE ANNUALE, OLTRE A UN GRAFICO DEGLI INCASSI MENSILI.

STAMPA ELENCO ARTICOLI

IL METODO STAMPA_ELENCOARTICOLI VISUALIZZA UN ELENCO DI ARTICOLI PRESENTI NEL MAGAZZINO. DOPO AVER AZZERATO IL CONTENUTO DELL'OUTPUT, PER OGNI ARTICOLO VIENE MOSTRATA LA MARCA, LA CATEGORIA, IL PREZZO, LA TAGLIA E LA DESCRIZIONE. OGNI ARTICOLO È NUMERATO E SEPARATO DA UNA LINEA PER UNA MIGLIORE LEGGIBILITÀ.

STAMPA LISTA CAPI ACQUISTATI

IL METODO STAMPA_LISTACAPIACQUISTATI VISUALIZZA UN ELENCO DEGLI ARTICOLI ACQUISTATI. DOPO AVER AZZERATO IL CONTENUTO DELL'OUTPUT, PER OGNI ARTICOLO VENGONO MOSTRATI LA MARCA, LA CATEGORIA, IL PREZZO, LA TAGLIA, LA DESCRIZIONE E LA DATA DI ACQUISTO. OGNI ARTICOLO È NUMERATO E SEPARATO DA UNA LINEA PER UNA MIGLIORE LEGGIBILITÀ.

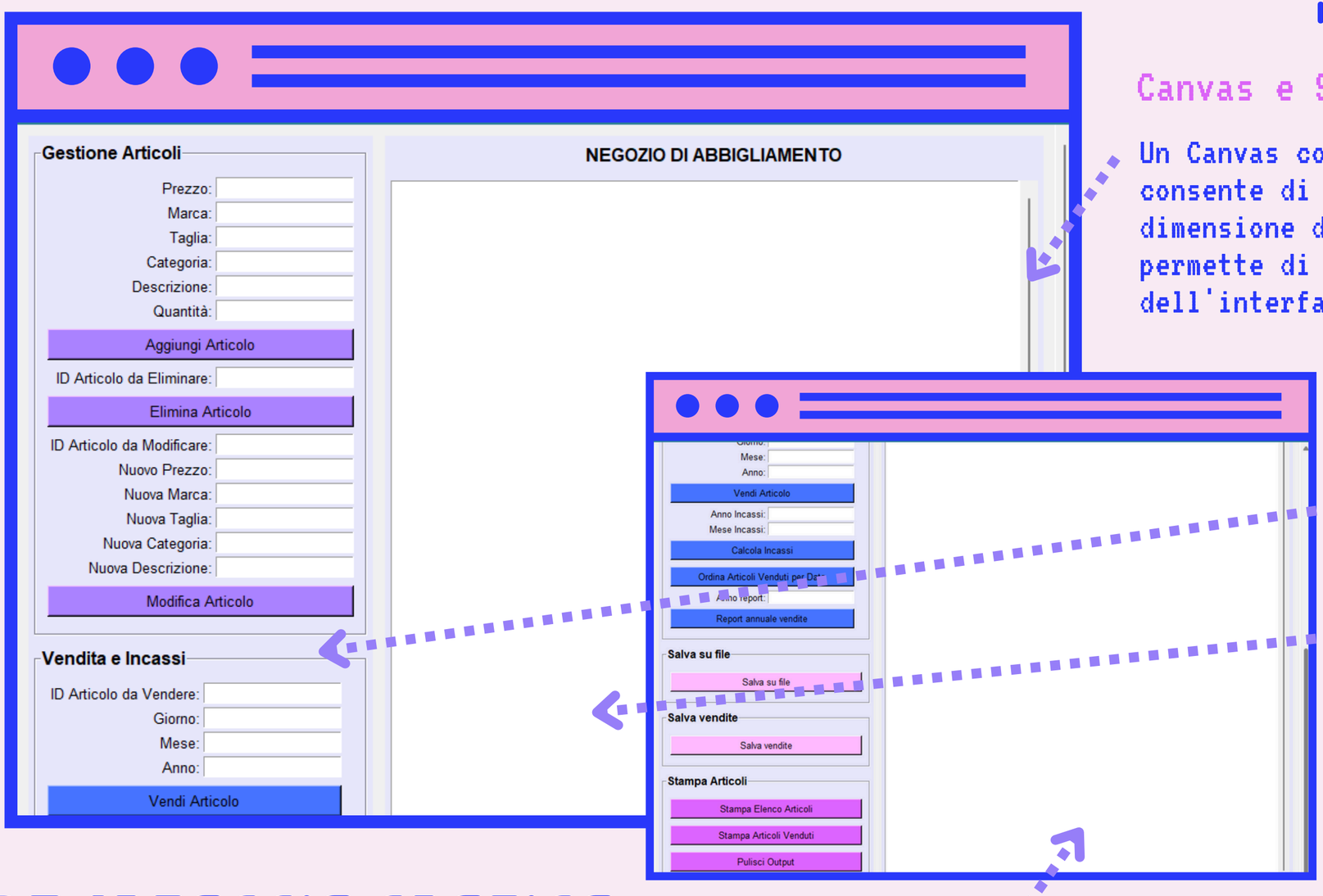
SALVA SU FILE

IL METODO SALVA_ARTICOLI SALVA GLI ARTICOLI PRESENTI NEL MAGAZZINO IN UN FILE. UTILIZZA UN BLOCCO TRY-EXCEPT PER GESTIRE EVENTUALI ERRORI DURANTE IL SALVATAGGIO, COME FILE NON TROVATO O ERRORI DI INPUT/OUTPUT. PER OGNI ARTICOLO, GENERA UNA RIGA DI SALVATAGGIO (ATTRUBUTI SEPARATI DA “|”) E LA SCRIVE NEL FILE. AL TERMINE, MOSTRA UN MESSAGGIO DI SUCCESSO O UN MESSAGGIO DI ERRORE IN CASO DI PROBLEMI.

CARICA DA FILE

IL METODO CARICA_ARTICOLI SI OCCUPA DI CARICARE ARTICOLI DA UN FILE DI TESTO SPECIFICATO. APRE IL FILE IN MODALITÀ LETTURA E PER OGNI RIGA CREA UN'ISTANZA DELLA CLASSE ARTICOLO, SFRUTTANDO IL METODO CARICARIGASALVATAGGIO. IN CASO DI ERRORI DI FORMATO, GESTISCE LE ECCEZIONI MOSTRANDO UN MESSAGGIO DI ERRORE.


Gestione Articoli
Prezzo:
Marca:
Taglia:
Categoria:
Descrizione:
Quantità:
Aggiungi Articolo
ID Articolo da Eliminare:
Elimina Articolo
ID Articolo da Modificare:
Nuovo Prezzo:
Nuova Marca:
Nuova Taglia:
Nuova Categoria:
Nuova Descrizione:
Modifica Articolo
Vendita e Incassi
ID Articolo da Vendere:
Giorno:
Mese:
Anno:
Vendi Articolo
NEGOZIO DI ABBIGLIAMENTO
INTERFACCIA GRAFICA
PANNELLO PRINCIPALE
Mese:
Anno:
Vendi Articolo
Anno Incassi:
Mese Incassi:
Calcola Incassi
Ordina Articoli Venduti per Data
Anno report:
Report annuale vendite
Salva su file
Salva su file
Salva vendite
Salva vendite
Stampa Articoli
Stampa Elenco Articoli
Stampa Articoli Venduti
Pulisci Output
Finestra Principale:
La finestra principale è
creata utilizzando
tkinter, una libreria
standard di Python per
la creazione di GUI.



Canvas e Scrollbar:

Un Canvas contiene un Frame scrollabile che consente di visualizzare contenuti oltre la dimensione della finestra. La scrollbar verticale permette di navigare tra le diverse sezioni dell'interfaccia.

Frame principali:

- Left Frame: Contiene diverse sezioni per la gestione degli articoli, le vendite, il salvataggio su file e la stampa degli articoli.
- Right Frame: Destinato a visualizzare l'output delle operazioni eseguite, come messaggi di conferma o report.

INTERFACCIA GRAFICA

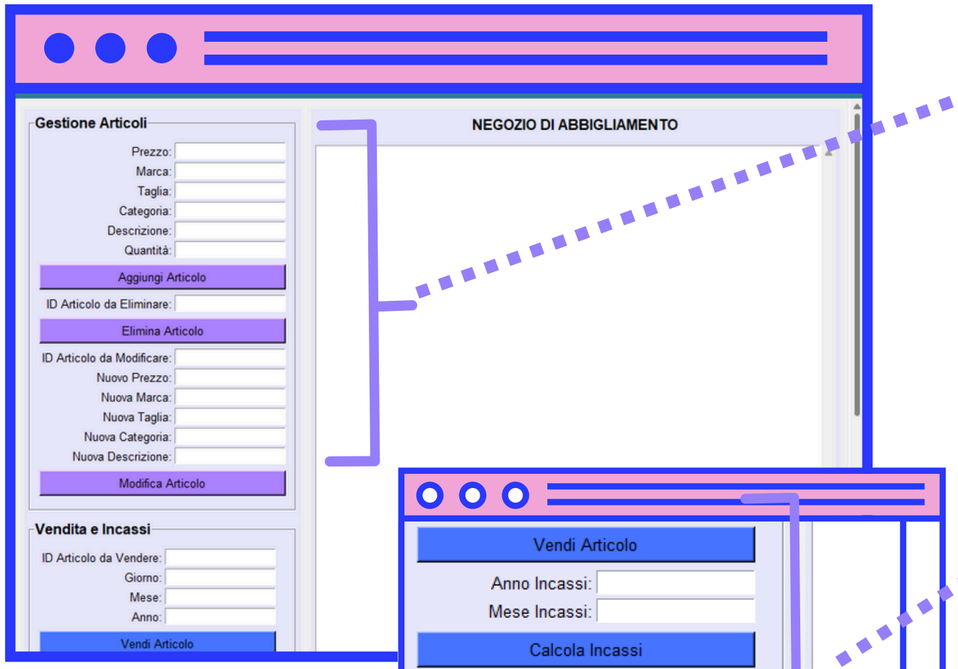
PANNELLO PRINCIPALE

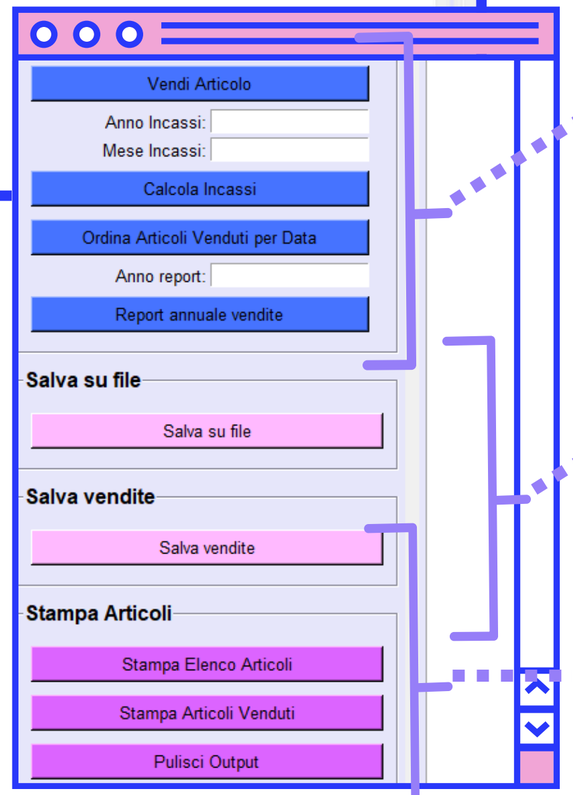
Output:

Un'area di output (utilizzando ScrolledText) mostra i messaggi e i report generati dalle operazioni eseguite, come conferme di aggiunta o eliminazione di articoli.

GUI

SEZIONI DI GESTIONE:





GESTIONE ARTICOLI:

Permette di aggiungere, eliminare e modificare articoli nel magazzino, tramite specifici pulsanti. Include campi di input per il prezzo, la marca, la taglia, la categoria, la descrizione e la quantità.

VENDITA E INCASSI:

Consente di registrare le vendite degli articoli, calcolare gli incassi e generare report. Include campi per inserire l'ID dell'articolo da vendere e la data della vendita.

SALVA SU FILE:

Sezione per salvare i dati degli articoli e delle vendite su file.

STAMPA ARTICOLI:

Permette di stampare l'elenco degli articoli e degli articoli venduti.

GUI

REPORT ANNUALE:

Funzione report:

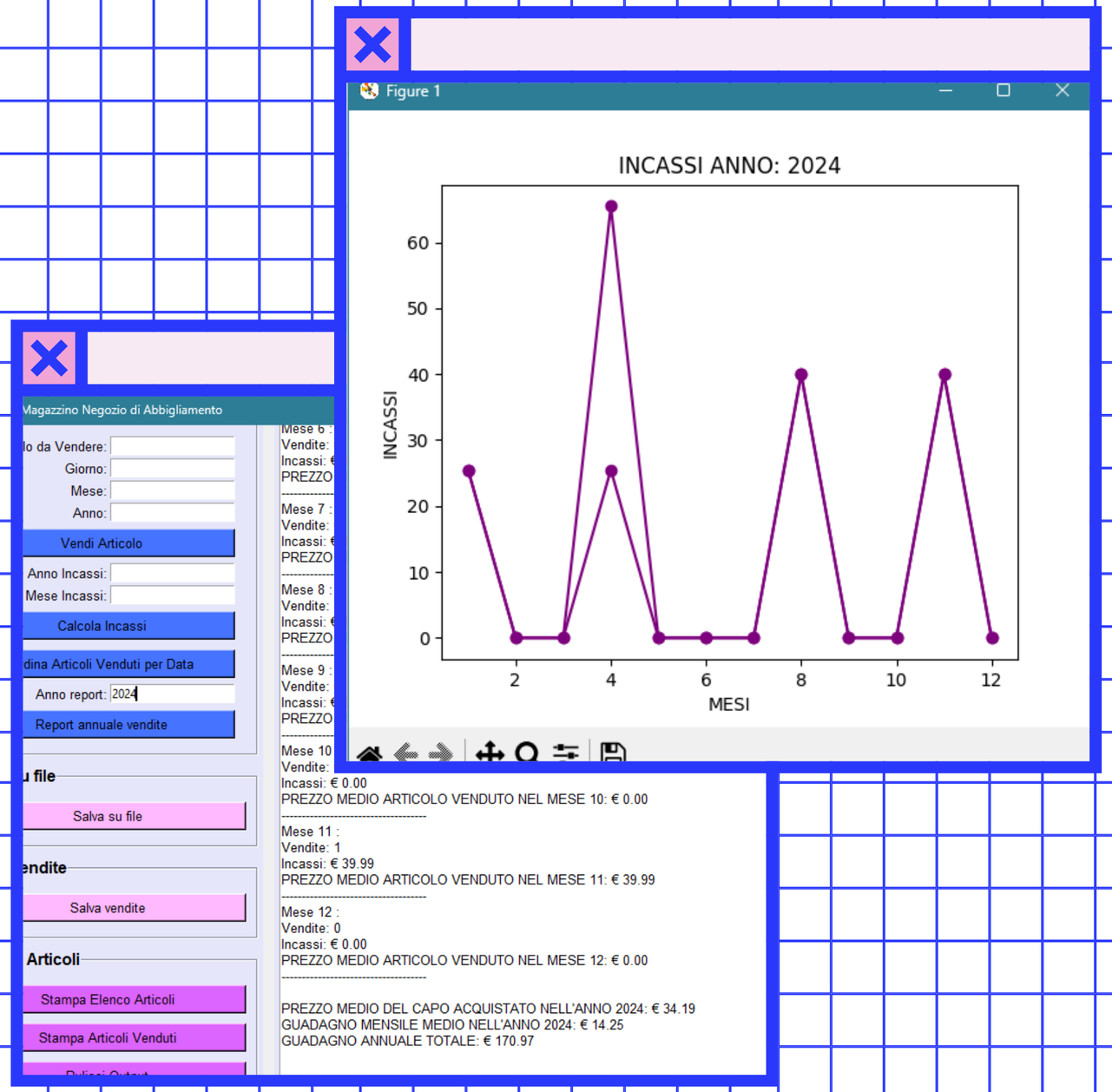
- Genera un report dettagliato delle vendite e degli incassi per un anno specificato dall'utente.

Dettagli del Report:

- Per ogni mese dell'anno, vengono calcolati: vendite mensili, incassi mensili, prezzo medio per articolo venduto, prezzo medio annuale per articolo, guadagno mensile medio, guadagno annuale totale

Grafico degli Incassi:

- Utilizza matplotlib per visualizzare gli incassi mensili in un grafico a linee.
- L'asse x rappresenta i mesi dell'anno, mentre l'asse y mostra gli incassi corrispondenti.



LIBRERIE

UTILIZZATE:

"TKINTER"

Per l'interfaccia grafica con l'utente.

"MATPLOTLIB"

Per visualizzare gli incassi mensili di un anno.

STRUMENTI UTILIZZATI

DRAW.IO

Per la creazione dei diagrammi UML digitali, ho utilizzato il tool online “draw.io”.

PYCHARM

Per la scrittura del codice del software, ho scelto “PyCharm”, uno degli ambienti di sviluppo più utilizzati per il linguaggio Python.

CANVA

La presentazione è stata realizzata utilizzando “Canva Premium”, un tool che offre una vasta gamma di modelli e opzioni di design.