

RELAZIONE ELABORATO

INGEGNERIA DEL SOFTWARE 2017-2018

ESERCIZIO 3 – Articoli Sportivi

Elia Piacentini – VR409352
Federico Boschi – VR408846

SOMMARIO

INTRODUZIONE	3
REQUISITI	3
ASSUNZIONI	4
DOCUMENTAZIONE.....	5
USE CASE	5
USE CASE MAGAZZINIERE.....	5
USE CASE RESPONSABILE.....	5
USE CASE SEGRETERIA.....	5
ACTIVITY DIAGRAMS	6
ACTIVITY DIAGRAM LOGIN	6
ACTIVITY DIAGRAM REGISTRAZIONE INGRESSO – MAGAZZINIERE.....	7
ACTIVITY DIAGRAM REGISTRAZIONE USCITA – MAGAZZINIERE	8
ACTIVITY DIAGRAM SPOSTA ARTICOLO – MAGAZZINIERE.....	9
ACTIVITY DIAGRAM ORDINA ARTICOLI – RESPONSABILE NEGOZIO.....	10
ACTIVITY DIAGRAM VISUALIZZA ORDINI – RESPONSABILE NEGOZIO	11
ACTIVITY DIAGRAM VISUALIZZA MOVIMENTI – SEGRETERIA	12
ACTIVITY DIAGRAM INSERISCI ARTICOLI IN CATALOGO – SEGRETERIA	13
SEQUENCE DIAGRAMS	14
SEQUENCE DIAGRAM LOGIN	14
SEQUENCE DIAGRAM INGRESSO IN MAGAZZINO.....	15
SEQUENCE DIAGRAM USCITA DAL MAGAZZINO.....	16
SEQUENCE DIAGRAM SPOSTAMENTO PRODOTTO IN MAGAZZINO	17
CLASS DIAGRAM	18
METODOLOGIA DI SVILUPPO	19
PATTERN.....	19
PATTERN ARCHITETTURALI.....	19
DESIGN PATTERN.....	19
TESTING.....	19

INTRODUZIONE

L'obiettivo dell'elaborato è quello di creare un sistema gestionale per una catena di negozi di articoli sportivi. In questa relazione abbiamo raccolto la documentazione sviluppata in fase di progettazione del prototipo.

REQUISITI

Viene riportato il testo dell'elaborato, con i requisiti funzionali evidenziati.

Si vuole progettare un sistema informatico per gestire il magazzino di una catena di negozi di articoli sportivi. Il negozio vende articoli di diversa tipologia, raggruppati per sport.

Per ogni tipo articolo si registra: un nome univoco, una descrizione, lo sport, e i materiali utilizzati per produrlo.

Il sistema registra tutti gli articoli in magazzino memorizzando per ogni articolo: il tipo di articolo, un codice univoco, il prezzo e la data di produzione.

Gli articoli in magazzino vengono gestiti dal sistema che registra per ogni ingresso in magazzino: un codice interno univoco, la data e tutti articoli entrati e le loro posizioni in magazzino.

Per ogni uscita il sistema registra: la data e il numero di bolla (univoco), tutti gli articoli usciti, il negozio che li ha ordinati e lo spedizioniere che li ritira.

Per ogni negozio della catena il sistema registra: il codice fiscale, il nome, l'indirizzo e la città.

Il sistema memorizza inoltre gli ordini dei negozi registrando: il negozio che ha effettuato l'ordine, un codice ordine univoco, la data dell'ordine, i tipi di articolo ordinati e per ogni tipo di articolo la quantità ordinata e il prezzo totale.

Quando un ordine viene evaso si registra un'uscita dal magazzino che viene collegata all'ordine al quale si riferisce. Si suppone che per ogni ordine evaso si abbia una sola uscita dal magazzino.

Per ogni tipo di articolo il sistema memorizza esplicitamente alla fine di ogni mese dell'anno la quantità di articoli ricevuti in magazzino e la quantità di articoli usciti.

Il sistema deve permettere ai magazzinieri di inserire le informazioni relative ai movimenti di ingresso e uscita dal magazzino. I magazzinieri, inoltre, possono spostare un articolo da una posizione ad un'altra del magazzino, al fine di ottimizzare l'occupazione del magazzino.

La segreteria amministrativa della catena di negozi è responsabile dell'inserimento dei tipi di articolo. Essa può accedere al sistema e visualizzare i movimenti di magazzino rispetto agli ordini dei vari negozi.

Tutti gli utenti sono opportunamente autenticati dal sistema, prima che possano accedere alle funzionalità specifiche.

I responsabili dei negozi possono accedere al sistema per effettuare gli ordini e per avere un riassunto degli ordini passati.

ASSUNZIONI

Per poter svolgere l'elaborato abbiamo dovuto effettuare delle scelte progettuali necessarie per chiarire i diversi flussi di esecuzione.

Le nostre assunzioni sono state:

1. A ogni ordine corrisponde una sola uscita, e viceversa. In altre parole, un ordine ricevuto da un negozio viene effettuato tutto insieme, altrimenti non può essere effettuato.
2. Non è possibile annullare un ordine o un'uscita già creata.
3. Un responsabile può ordinare una quantità di articoli a piacere, anche se non presenti in magazzino. Una volta effettuato, l'ordine rimarrà quindi pendente fino a quando non vi saranno disponibili abbastanza articoli per poterlo completare.
4. Il magazzino è diviso in scaffali. Ogni scaffale ha un numero identificativo e 10 ripiani su cui possono essere posizionati gli articoli.
5. Ogni piano può contenere solo un articolo, a prescindere dalla sua dimensione.

DOCUMENTAZIONE

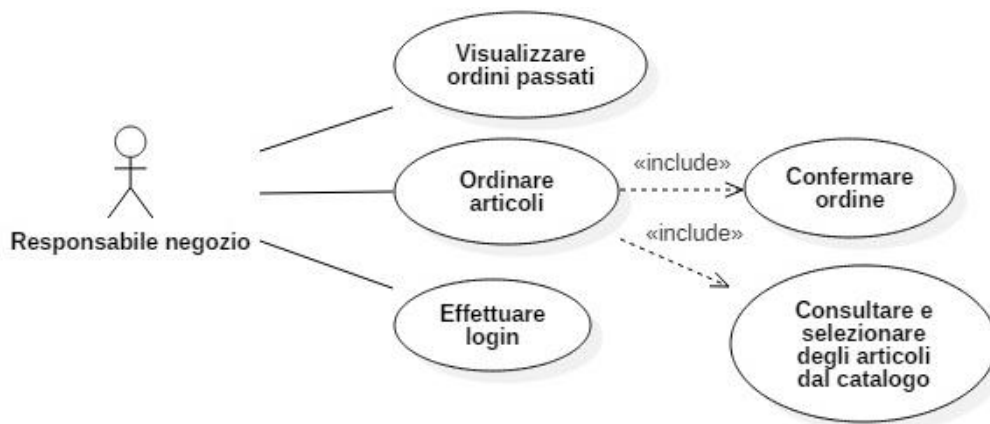
Di seguito vengono presentati tutti i documenti creati in fase di progettazione e sviluppo.

USE CASE

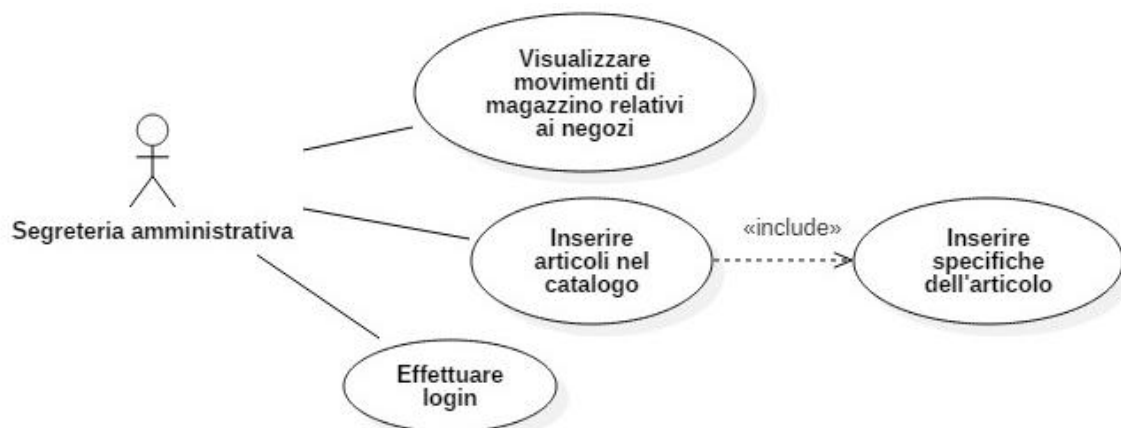
USE CASE MAGAZZINIERE



USE CASE RESPONSABILE



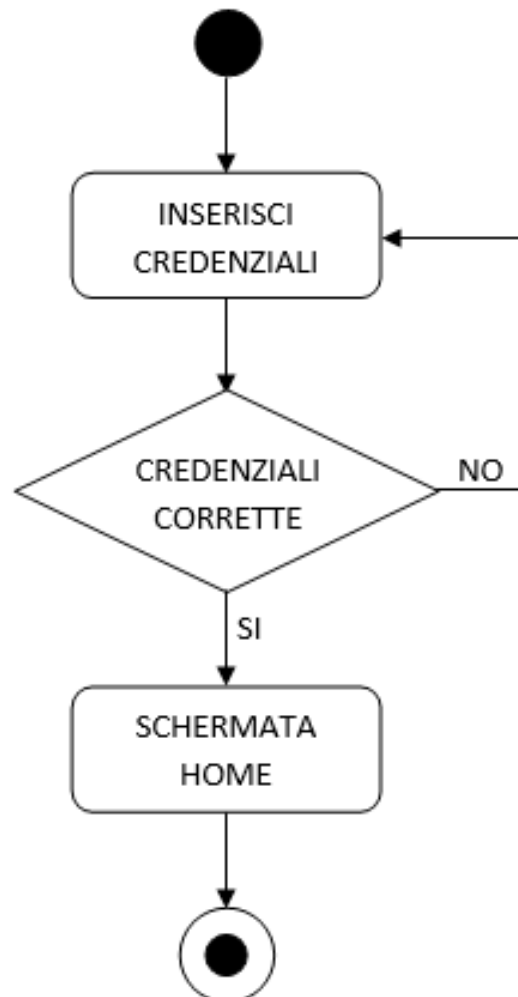
USE CASE SEGRETERIA



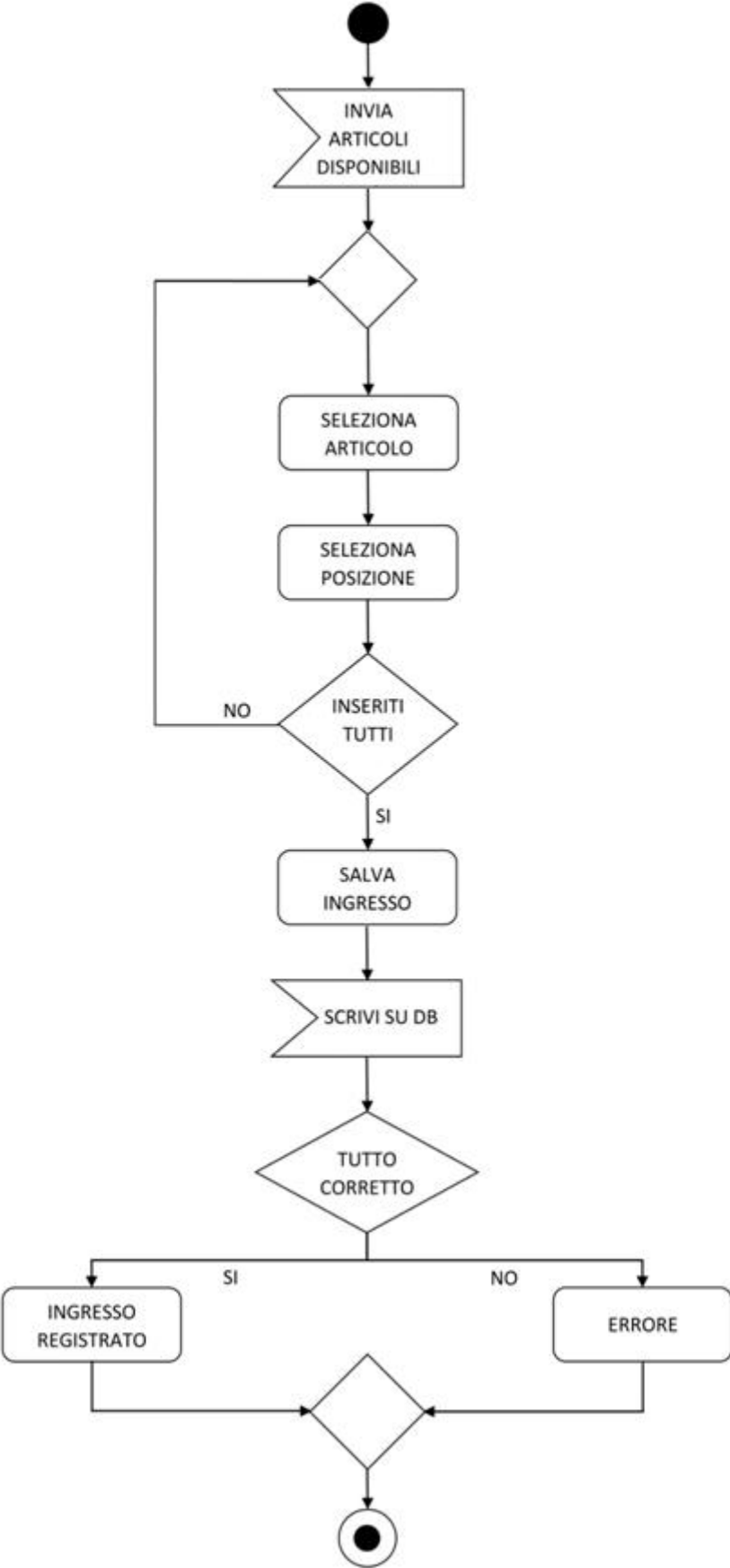
ACTIVITY DIAGRAMS

I seguenti Activity Diagrams illustrano il flusso di esecuzione dei più importanti metodi presenti nel prototipo.

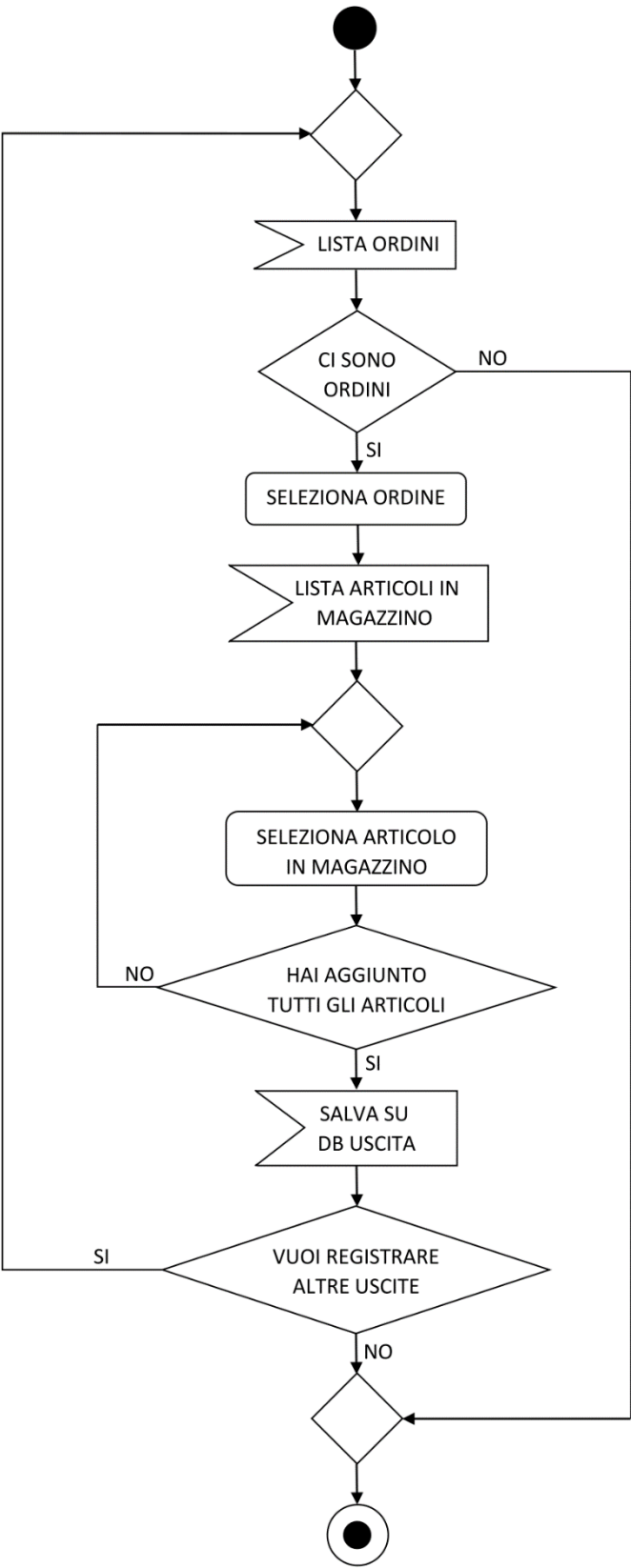
ACTIVITY DIAGRAM LOGIN



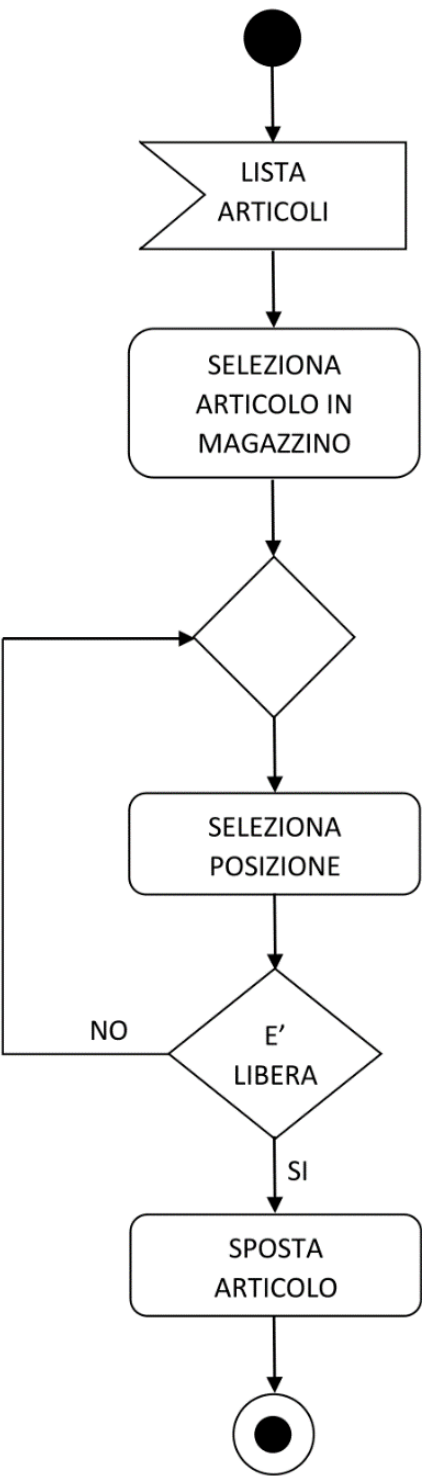
ACTIVITY DIAGRAM REGISTRAZIONE INGRESSO – MAGAZZINIERE



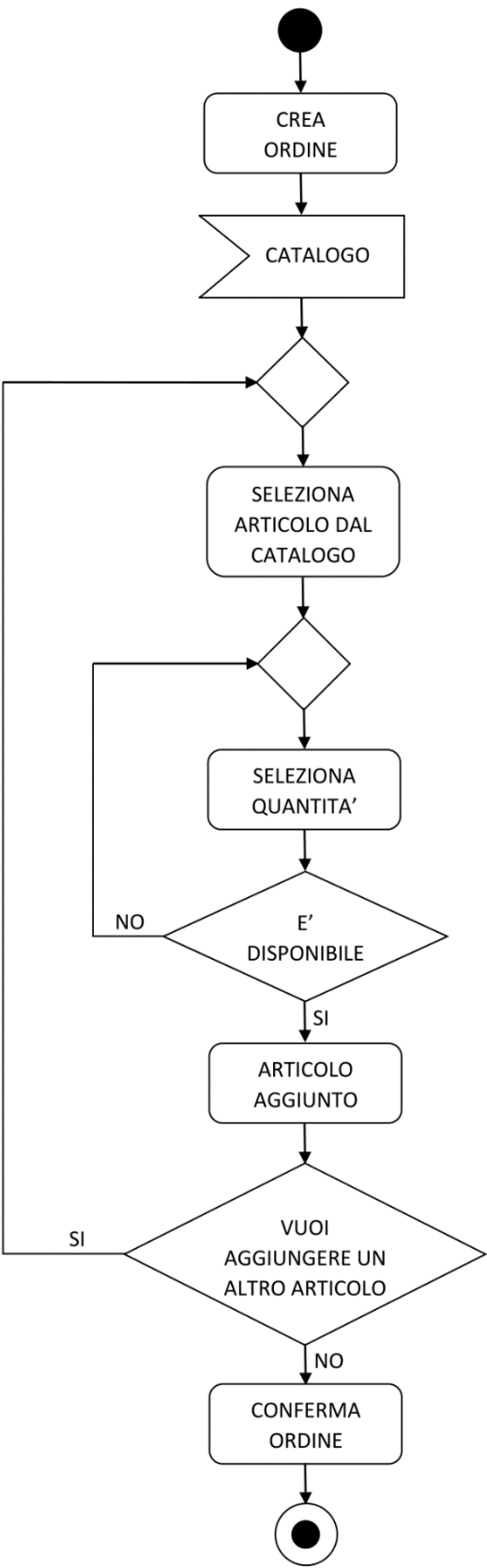
ACTIVITY DIAGRAM REGISTRAZIONE USCITA – MAGAZZINIERE



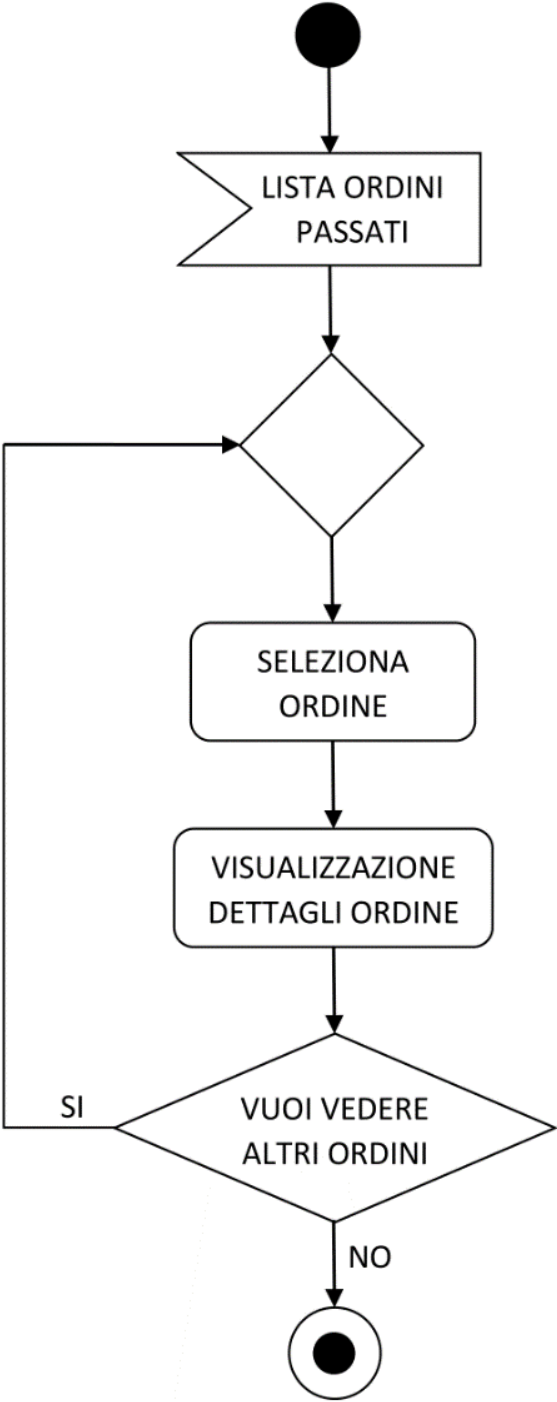
ACTIVITY DIAGRAM SPOSTA ARTICOLO – MAGAZZINIERE



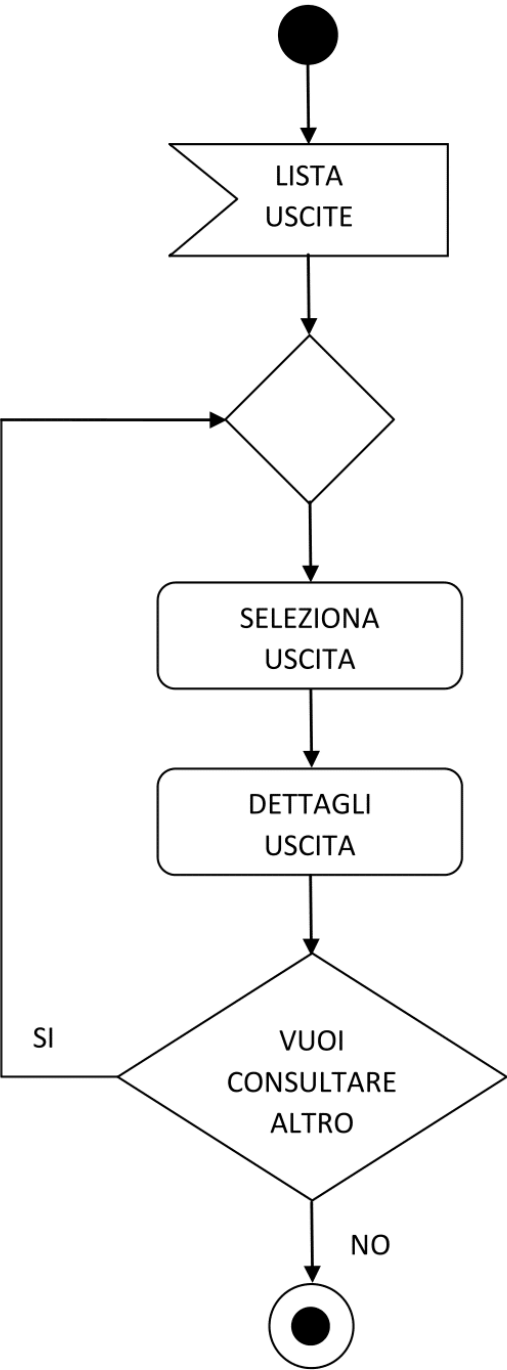
ACTIVITY DIAGRAM ORDINA ARTICOLI – RESPONSABILE NEGOZIO



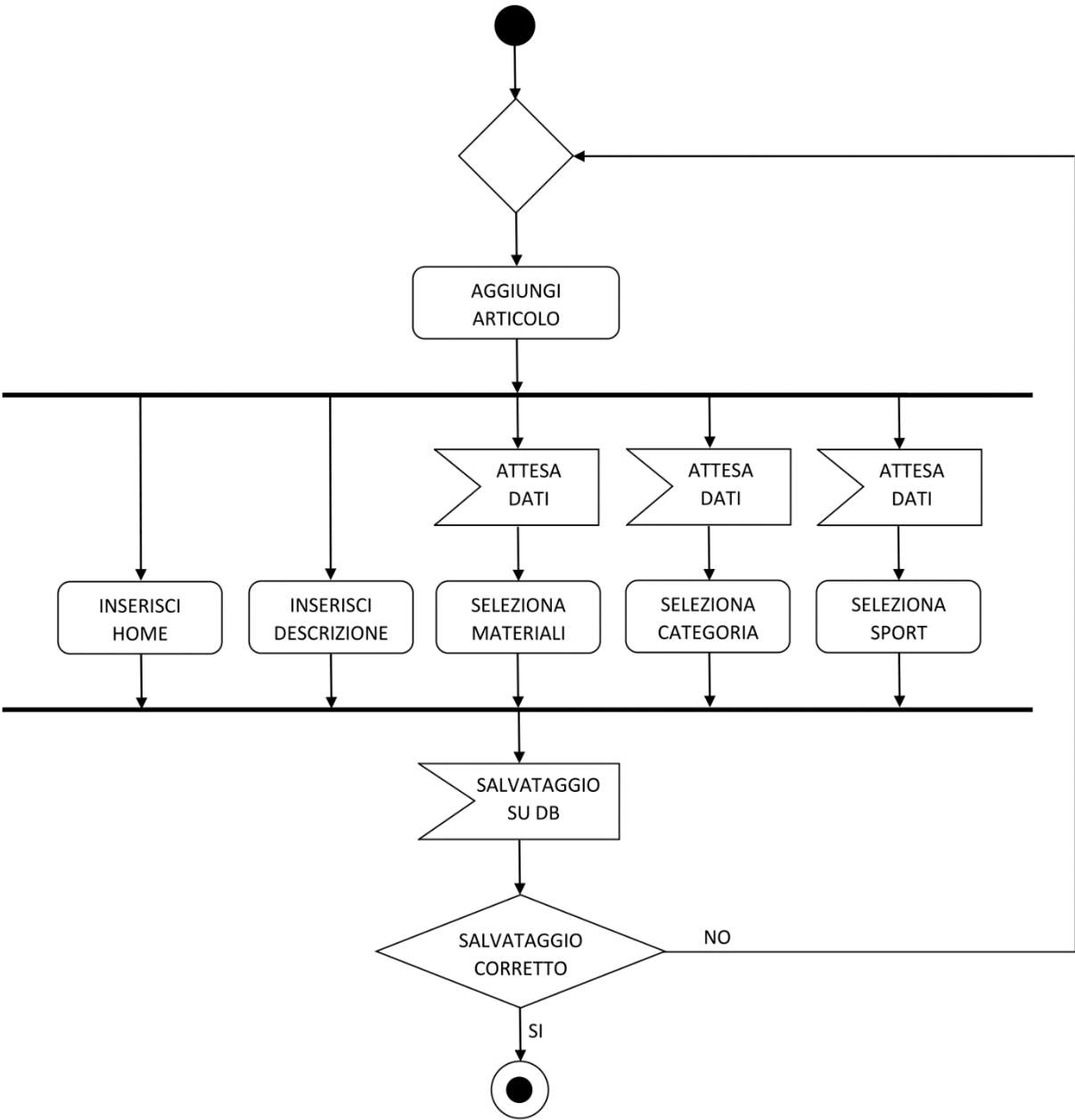
ACTIVITY DIAGRAM VISUALIZZA ORDINI – RESPONSABILE NEGOZIO



ACTIVITY DIAGRAM VISUALIZZA MOVIMENTI – SEGRETERIA



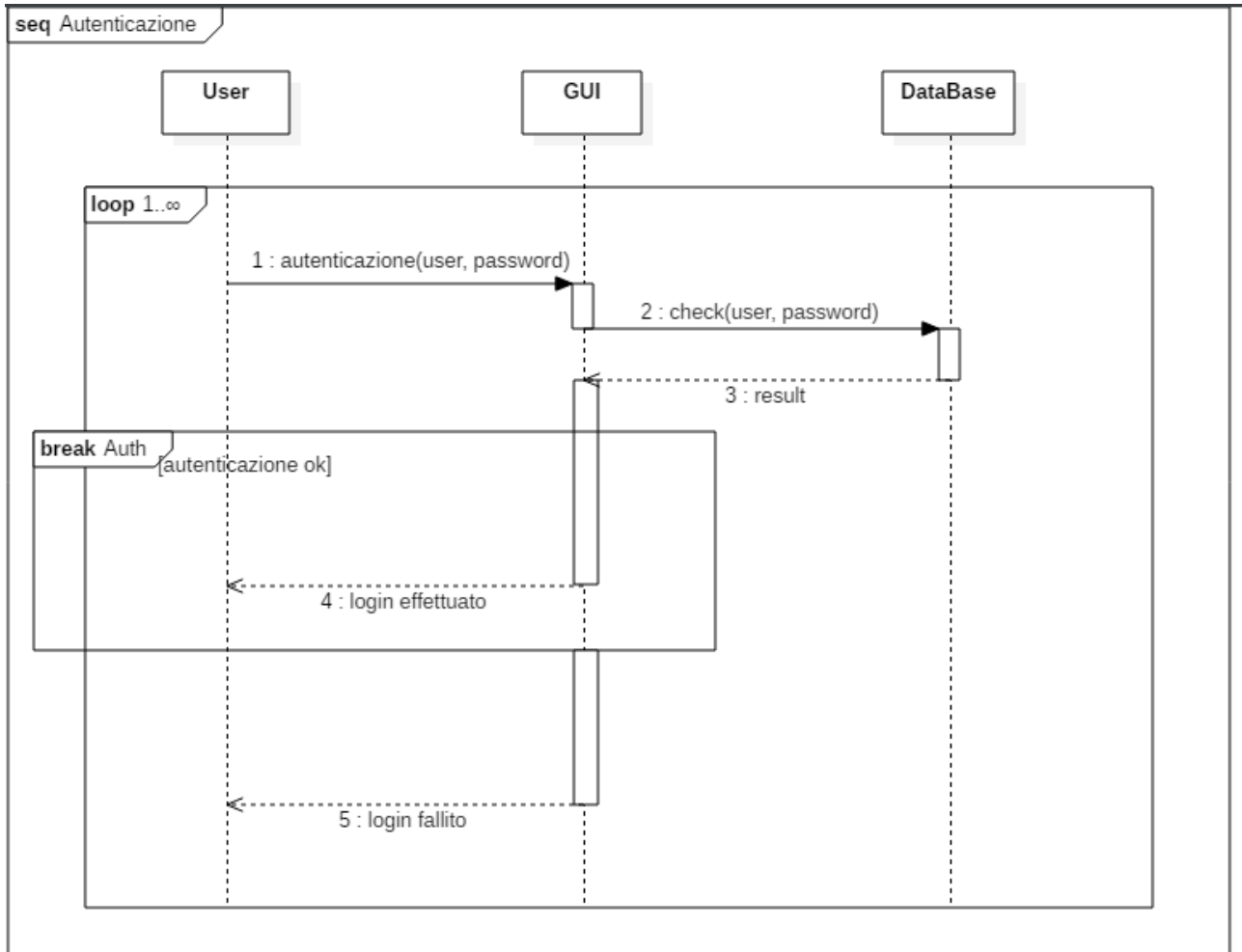
ACTIVITY DIAGRAM INSERISCI ARTICOLI IN CATALOGO – SEGRETERIA



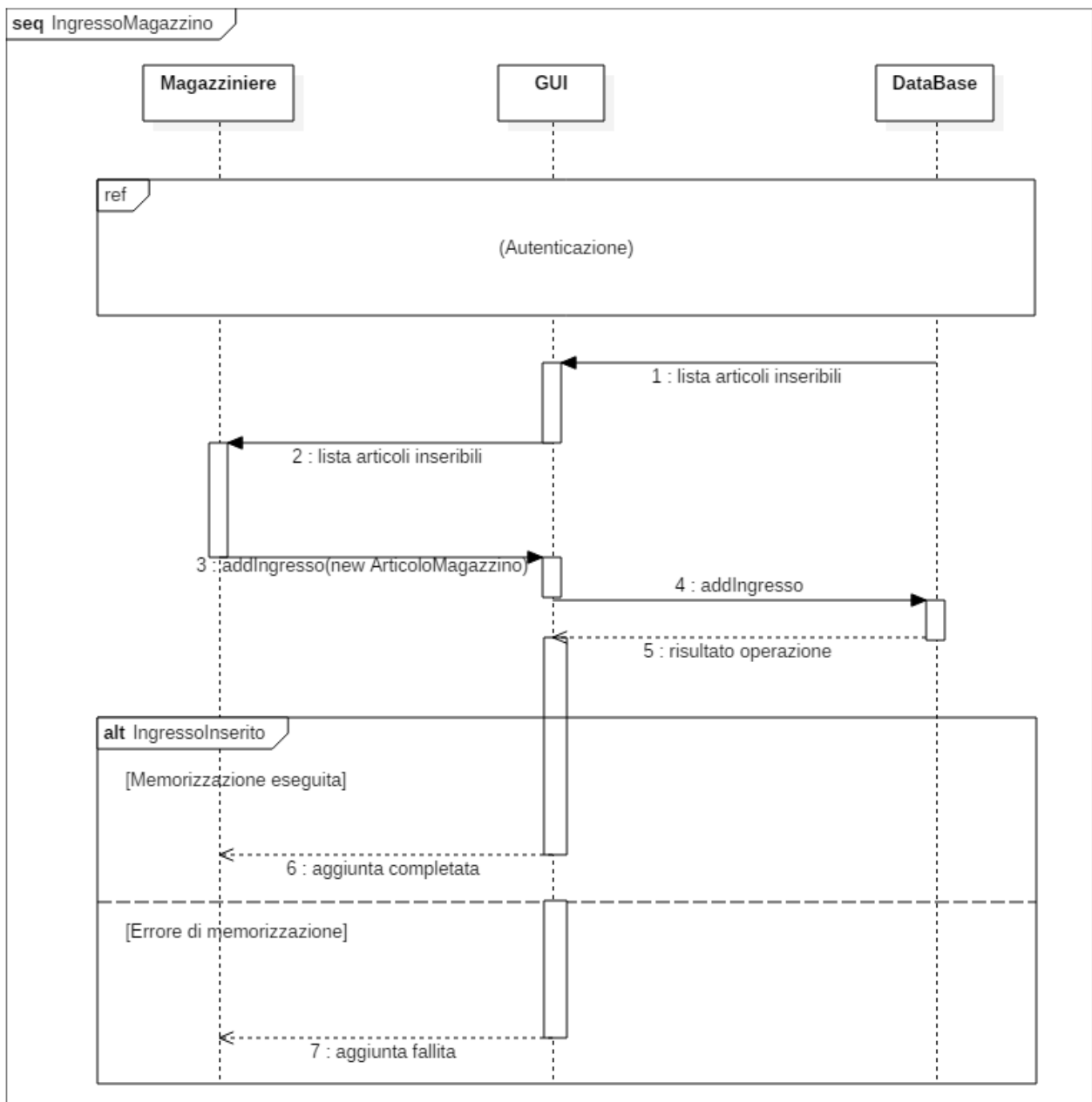
SEQUENCE DIAGRAMS

I seguenti Sequence Diagrams illustrano gli scambi di messaggi negli use case principali. Abbiamo ritenuto più interessante rispetto agli altri focalizzare l'attenzione sui compiti del magazziniere. Gli altri use case sono molto simili a questi come dinamiche.

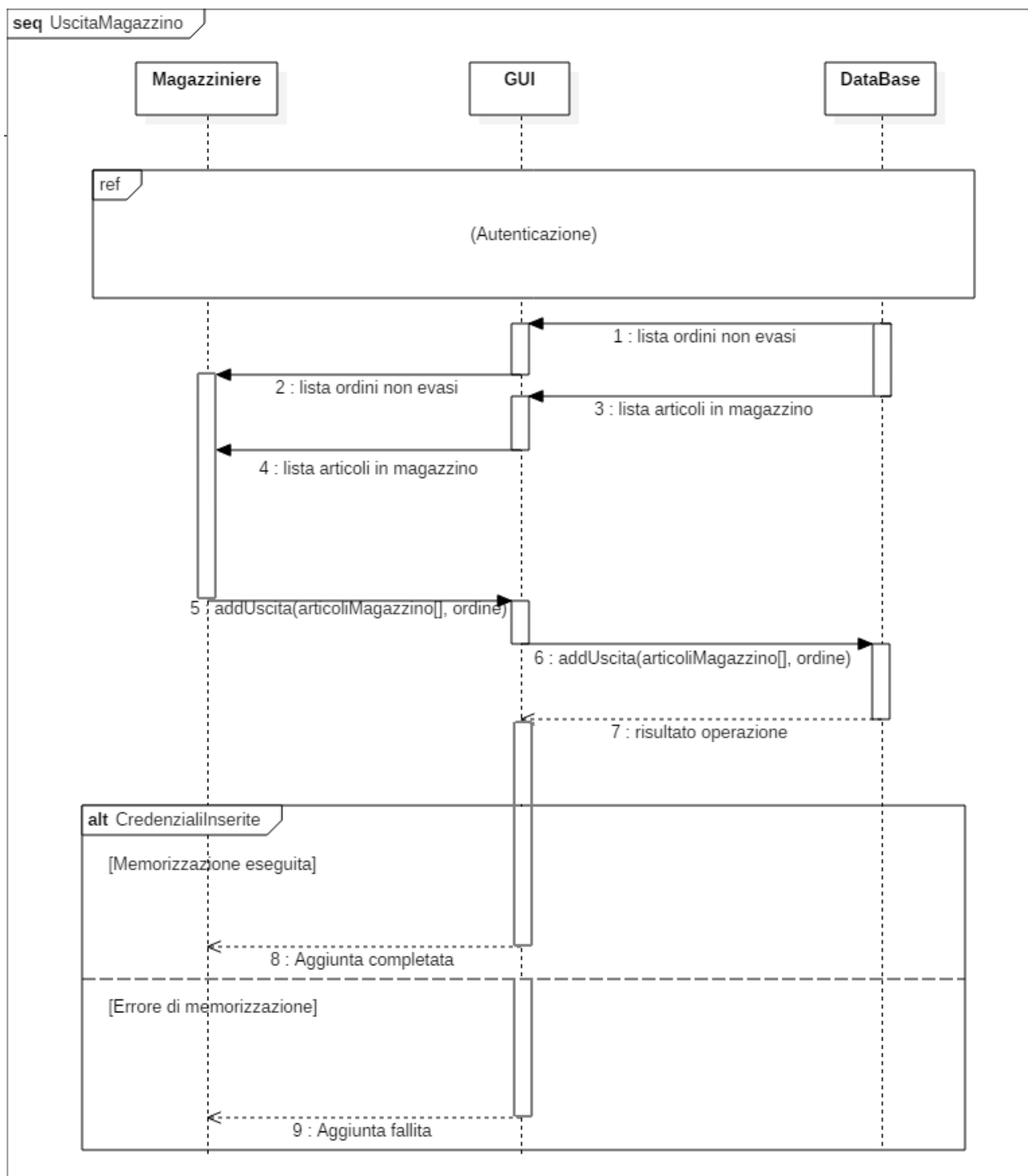
SEQUENCE DIAGRAM LOGIN



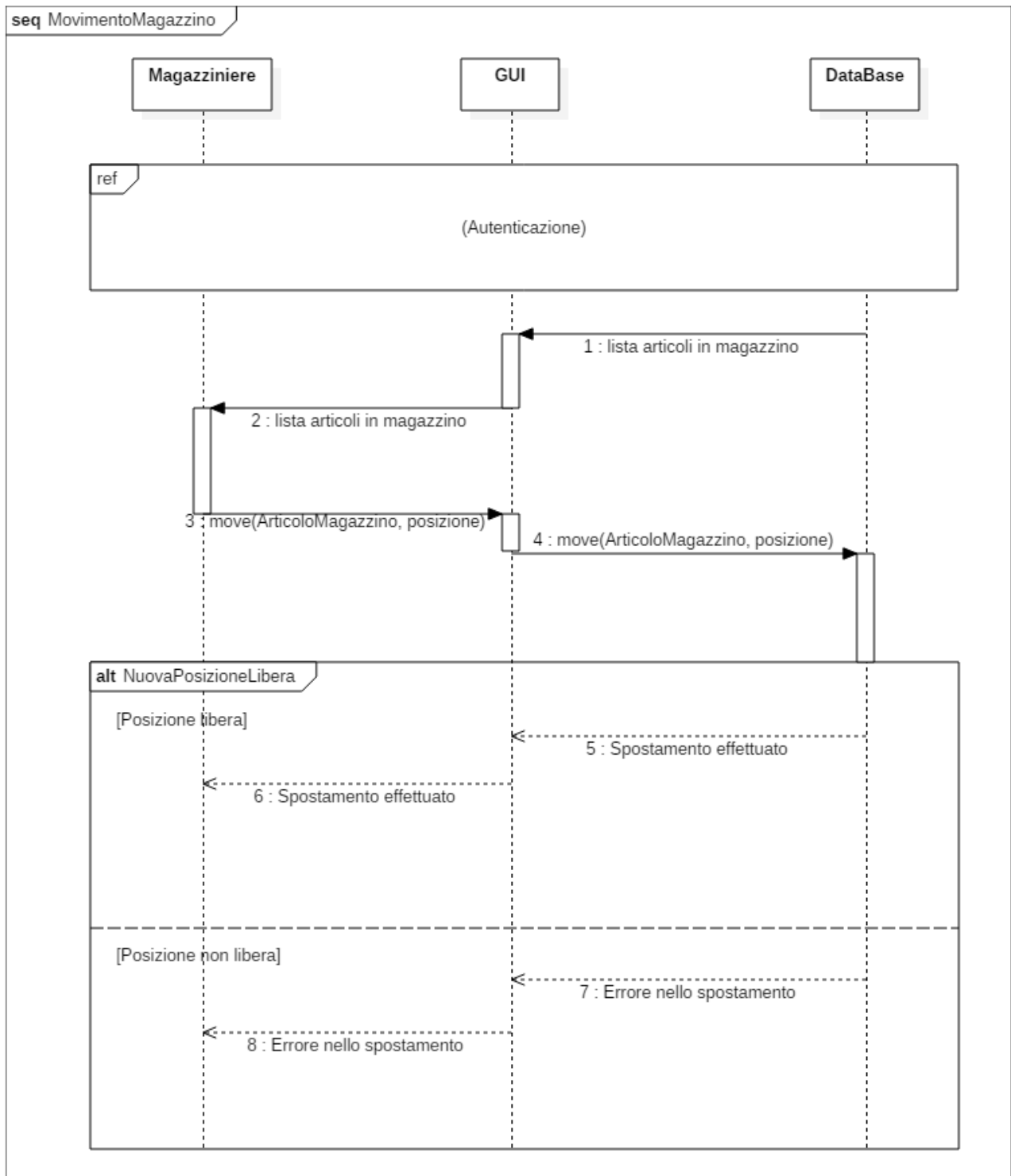
SEQUENCE DIAGRAM INGRESSO IN MAGAZZINO



SEQUENCE DIAGRAM USCITA DAL MAGAZZINO

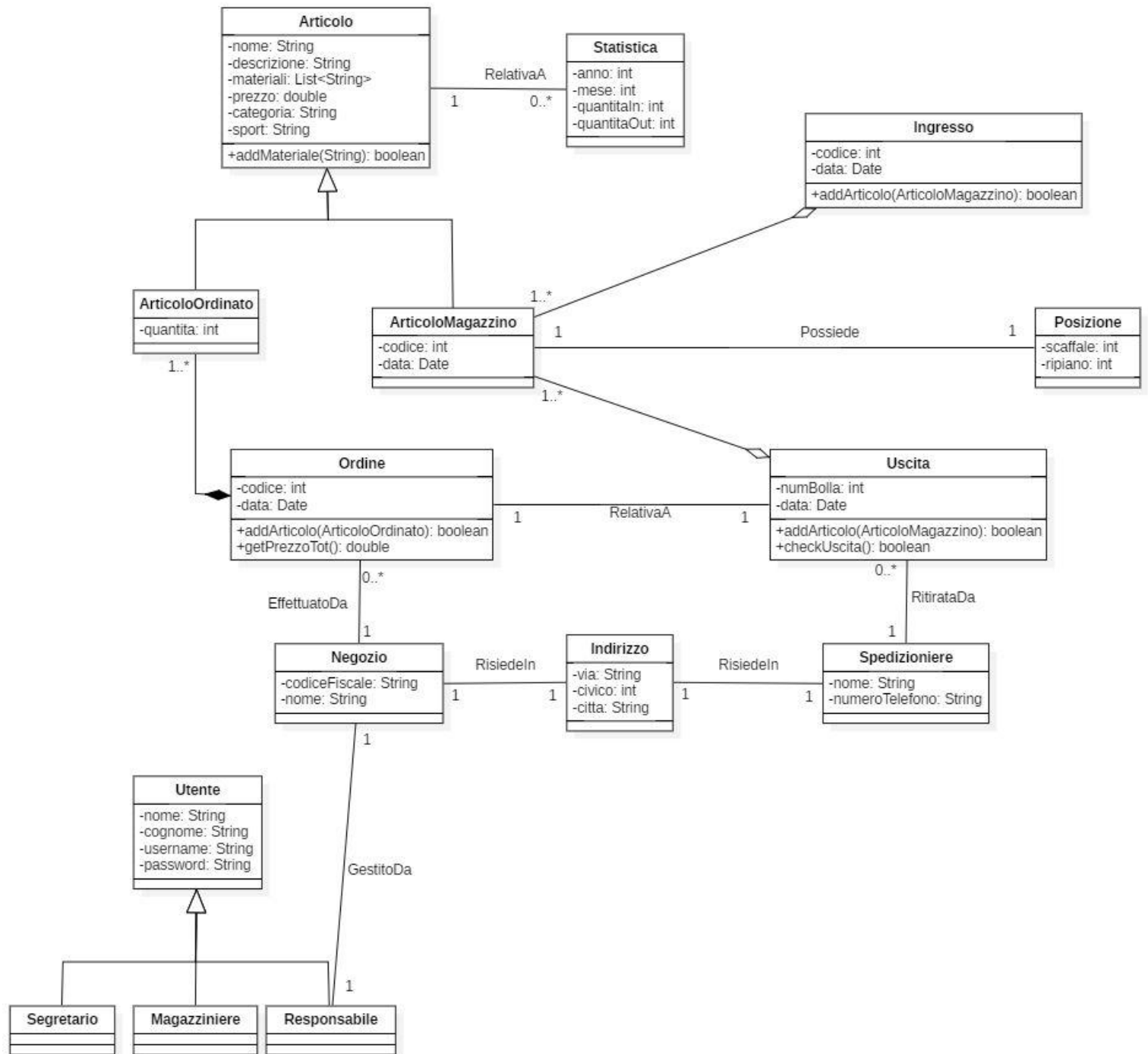


SEQUENCE DIAGRAM SPOSTAMENTO PRODOTTO IN MAGAZZINO



CLASS DIAGRAM

Il seguente Class Diagram rappresenta le classi e le relazioni fra esse che utilizzeremo nel nostro progetto Java per rappresentare le informazioni.



METODOLOGIA DI SVILUPPO

Abbiamo utilizzato una metodologia di sviluppo Agile per avere una consegna del progetto ed uno sviluppo rapido del software mescolando e alternando le fasi di progettazione, specifica e implementazione.

Abbiamo inoltre dato una maggiore importanza al codice che alla sua documentazione. La parte iniziale di analisi è stata svolta in coppia (pair designing).

Ci siamo divisi le attività in base ad aree di competenza, uno principalmente si è occupato della parte di gestione delle informazioni, l'altro principalmente della gestione grafica. Di conseguenza ognuno ha potuto crearsi una sua To Do List personale per organizzarsi le proprie attività.

Ci siamo comunque quotidianamente confrontati sulle attività svolte e sulle attività in elaborazione oltre che su vari problemi di sviluppo.

Abbiamo inoltre utilizzato Git come sistema di versioning e GitHub come piattaforma per la condivisione di codice

PATTERN

PATTERN ARCHITETTURALI

Per l'architettura del software abbiamo utilizzato il pattern MVC (Model View Control) suddividendo così la parte grafica da quella di gestione delle informazioni e da quella di controllo.

Abbiamo anche utilizzato il pattern architetturale DAO per accedere alle informazioni memorizzate in un Database. Infatti per ogni tabella del DB abbiamo creato una classe contenente i metodi per interagire col DB tramite delle query relative a quella tabella.

DESIGN PATTERN

Nel progetto abbiamo utilizzato il design pattern Observer per gestire l'iterazione dell'utente col software. In questo caso, gli oggetti ascoltati sono gli oggetti grafici (come JButton, JList ecc.), gli ascoltatori invece sono le classi Listener che gestiscono il verificarsi di un evento.

Un altro pattern usato è il Singleton per la creazione dell'oggetto DAOSettings che rappresenta le impostazioni di connessione al DB. L'oggetto è privato e viene creato solamente una volta nella classe Main; è reso accessibile alle altre classi tramite il metodo Main.getDAO().

TESTING

Per ogni attività completata abbiamo eseguito dei test sulla relativa parte, infine a progetto concluso abbiamo svolto un test finale verificando attentamente l'integrazione delle diverse parti sviluppate.