Università degli Studi di Verona
Dipartimento di Informatica
Corso di laurea in Informatica

Anno accademico 2017/2018

DOCUMENTAZIONE PROTOTIPO INGEGNERIA DEL SOFTWARE

Sistema informatico per gestire il magazzino di una catena di negozi di articoli sportivi

De Marchi Mirco - VR408481
Guadagnini Filippo - VR408802

Requisiti

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.

Requisiti funzionali

- Requisito funzionale n° 1: Magazziniere

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.

- Requisito funzionale n° 2: Segreteria amministrativa catena negozi

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.

- Requisito funzionale n° 3: Utenti

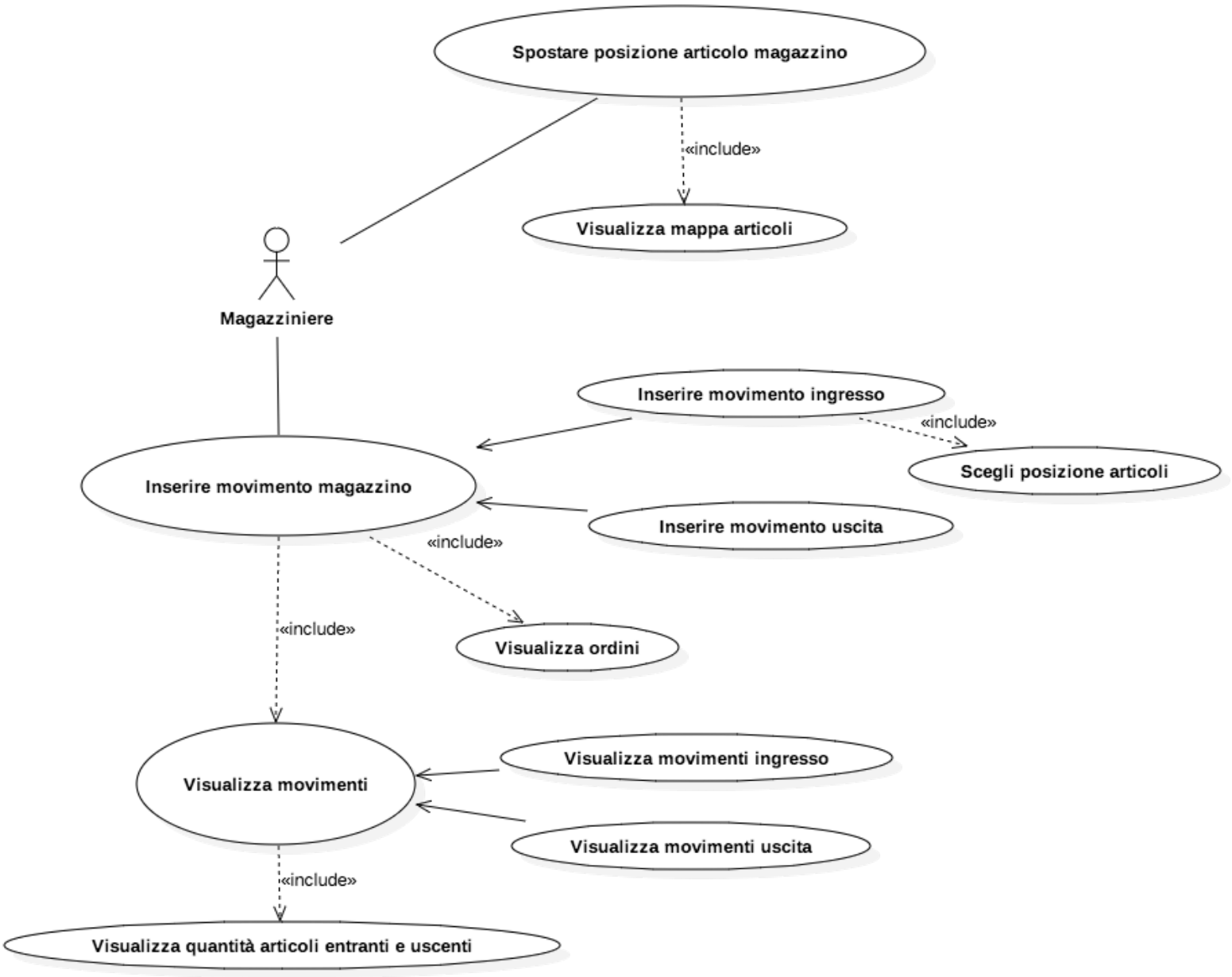
Tutti gli utenti sono opportunamente autenticati dal sistema, prima che possano accedere alle funzionalità specifiche.

- Requisito funzionale n° 4: Responsabile negozio

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

Casi d’uso

- Caso d’uso n° 1: Magazziniere



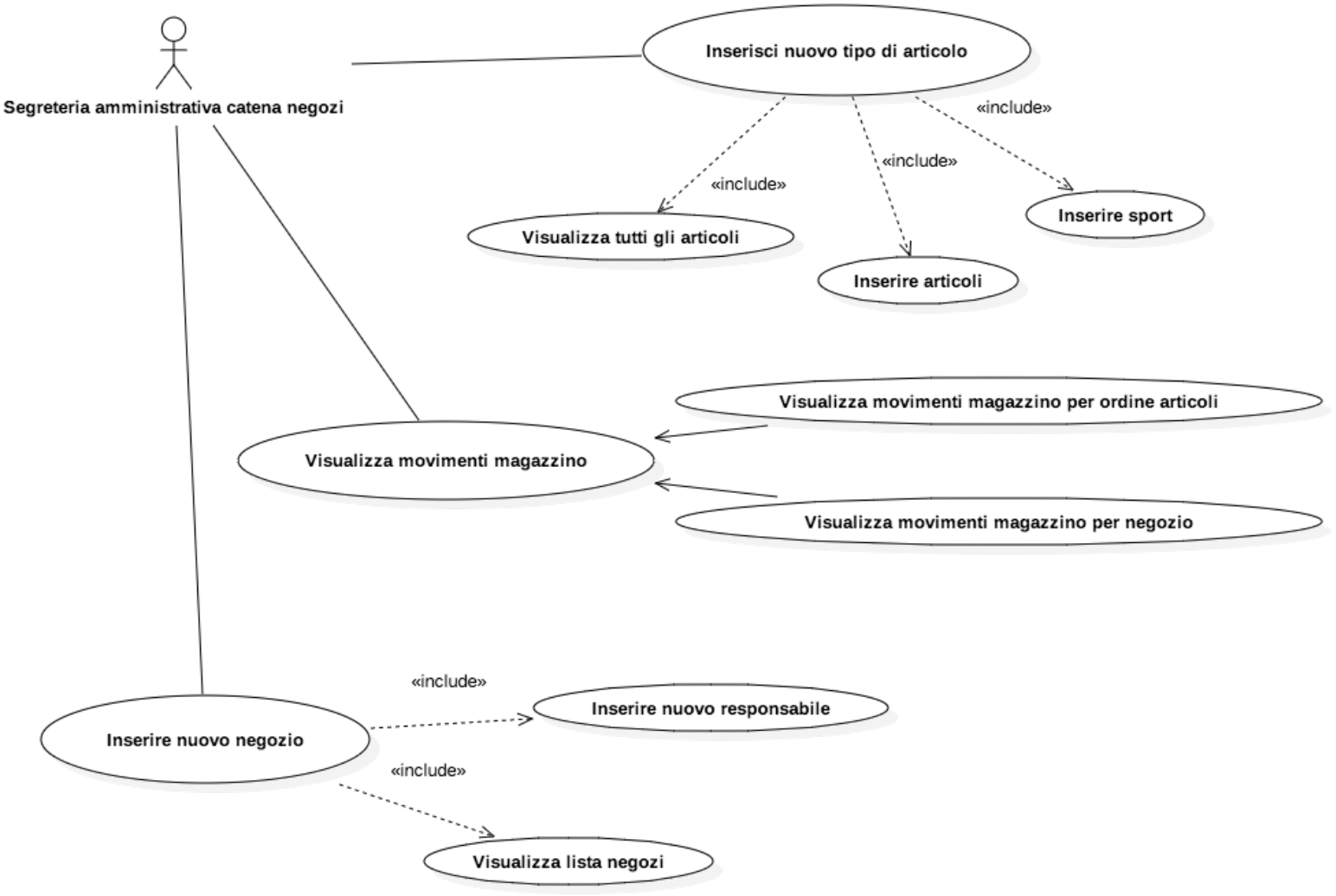
Scheda di specifica:

Attori: Magazziniere

Precondizioni: Aver effettuato il login del sistema

Operazioni permesse: Visualizzare ed inserire i movimenti di entrata e di uscita, visualizzare gli ordini e visualizzare la mappa degli articoli nel magazzino

- Caso d’uso n° 2: Segreteria amministrativa catena negozi



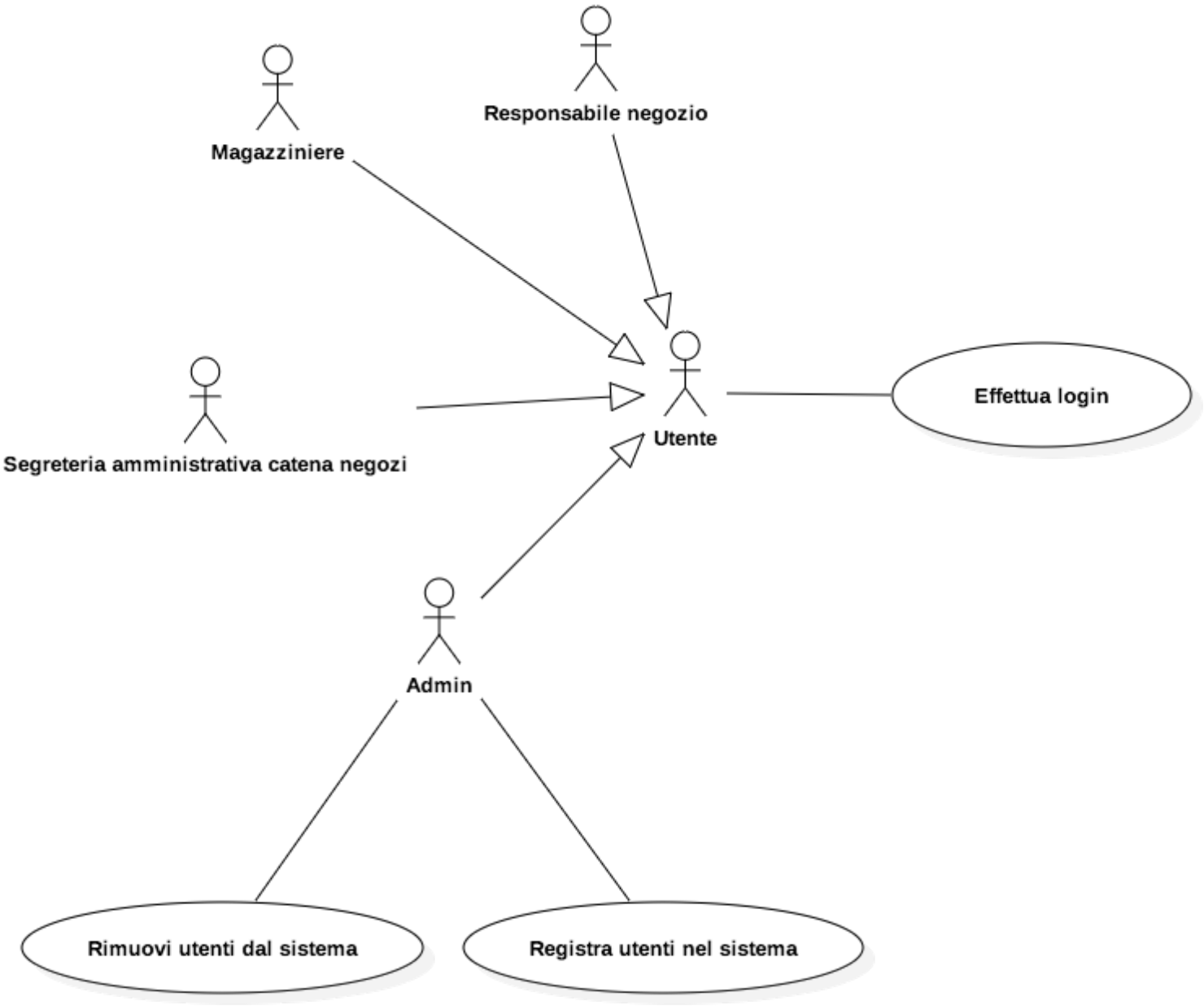
Scheda di specifica:

Attori: Segreteria amministrativa negozio

Precondizioni: Aver effettuato il login del sistema

Operazioni permesse: Creare un nuovo tipo di articolo, indicando anche sport e articoli, visualizzare gli articoli creati, visualizzare i movimenti del magazzino ordinando gli elementi, inserire un nuovo negozio con relativo responsabile

- Caso d’uso n° 3: Utente



Scheda di specifica:

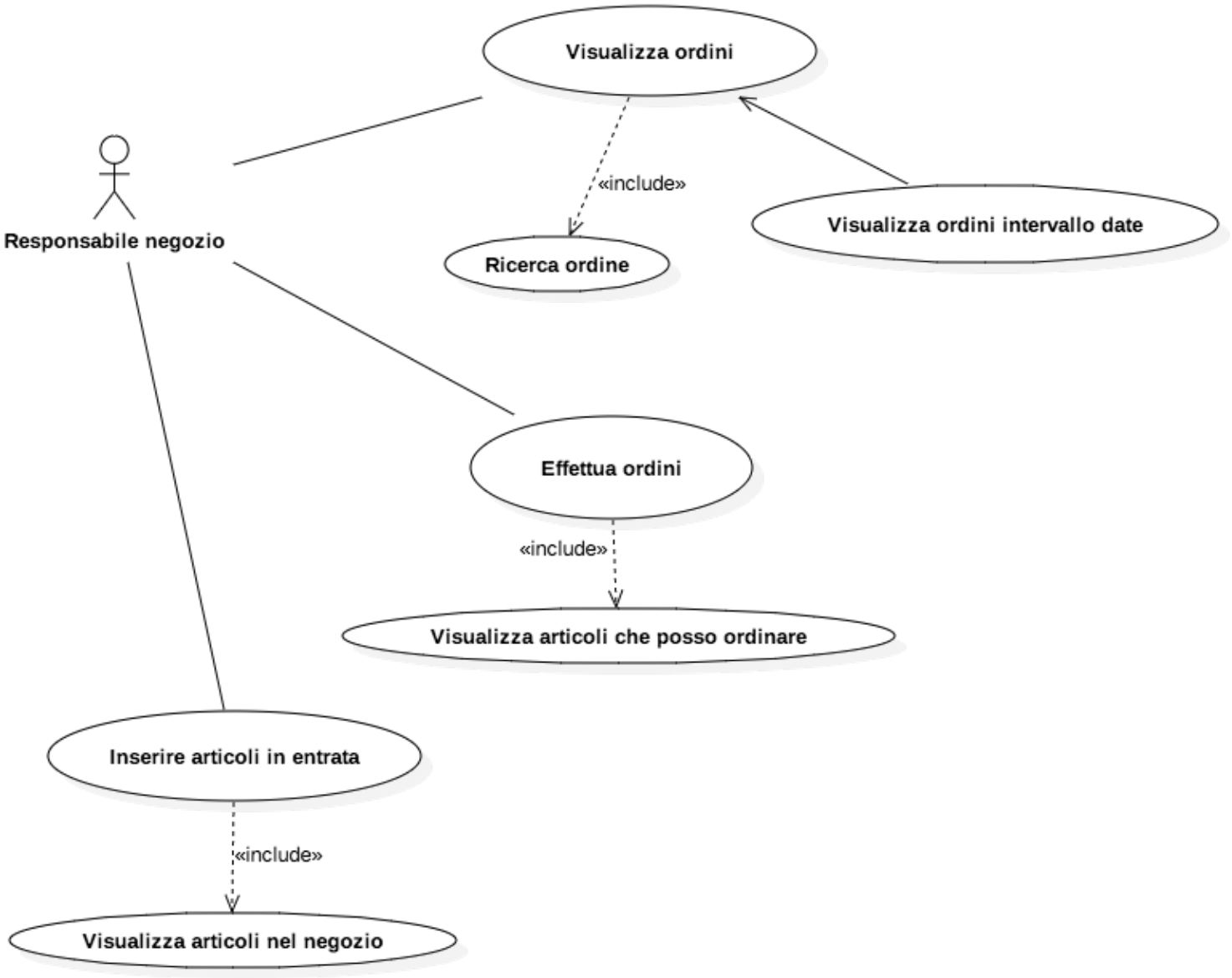
Attori: Utente

Precondizioni: Nessuna

Operazioni permesse: Inserire le proprie credenziali username e password e accedere al sistema.

Se si accede con Admin si possono aggiungere e rimuovere nuovi utenti

- Caso d’uso n° 4: Responsabile negozio



Scheda di specifica:

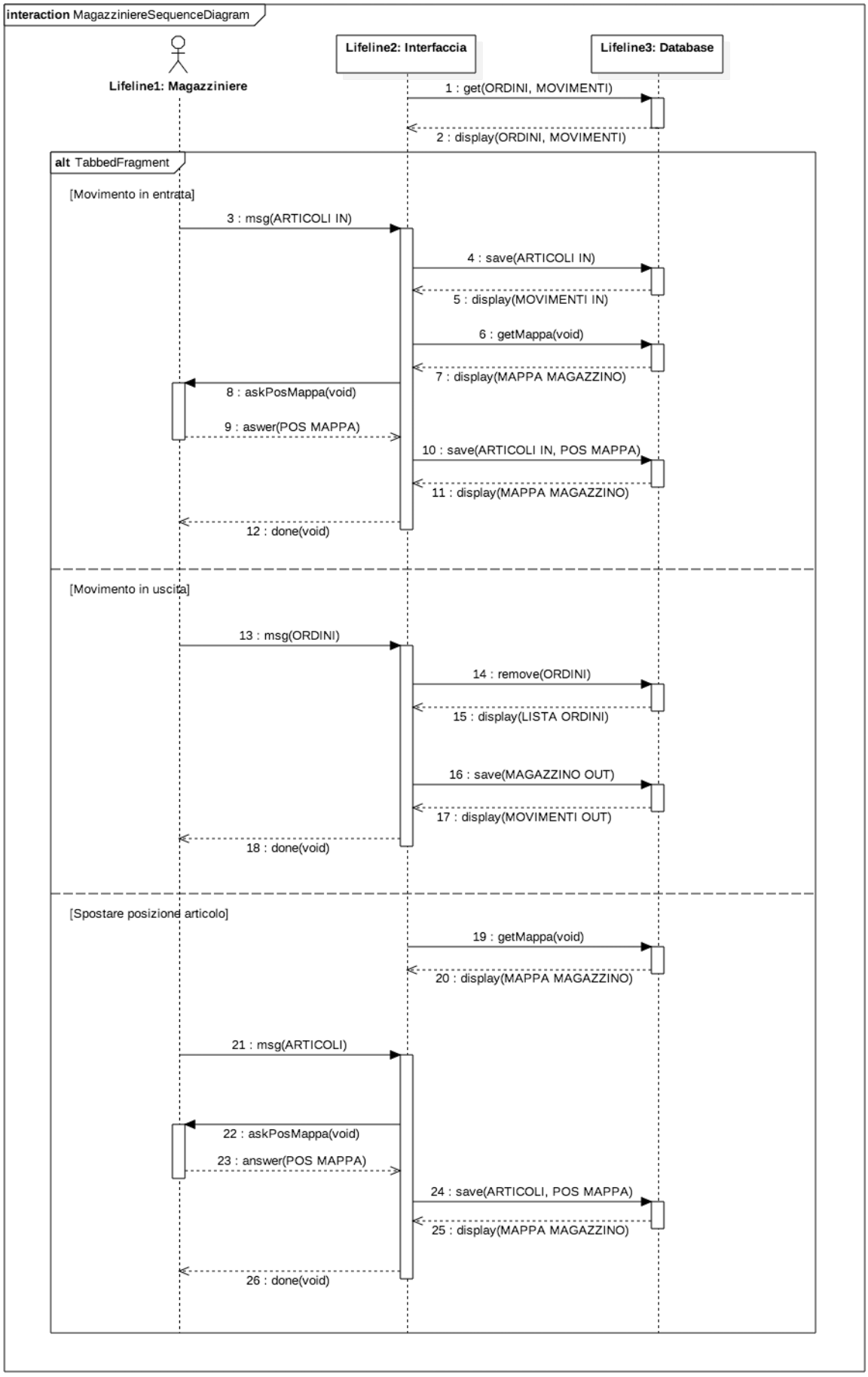
Attori: Responsabile Negozio

Precondizioni: Aver effettuato il login del sistema

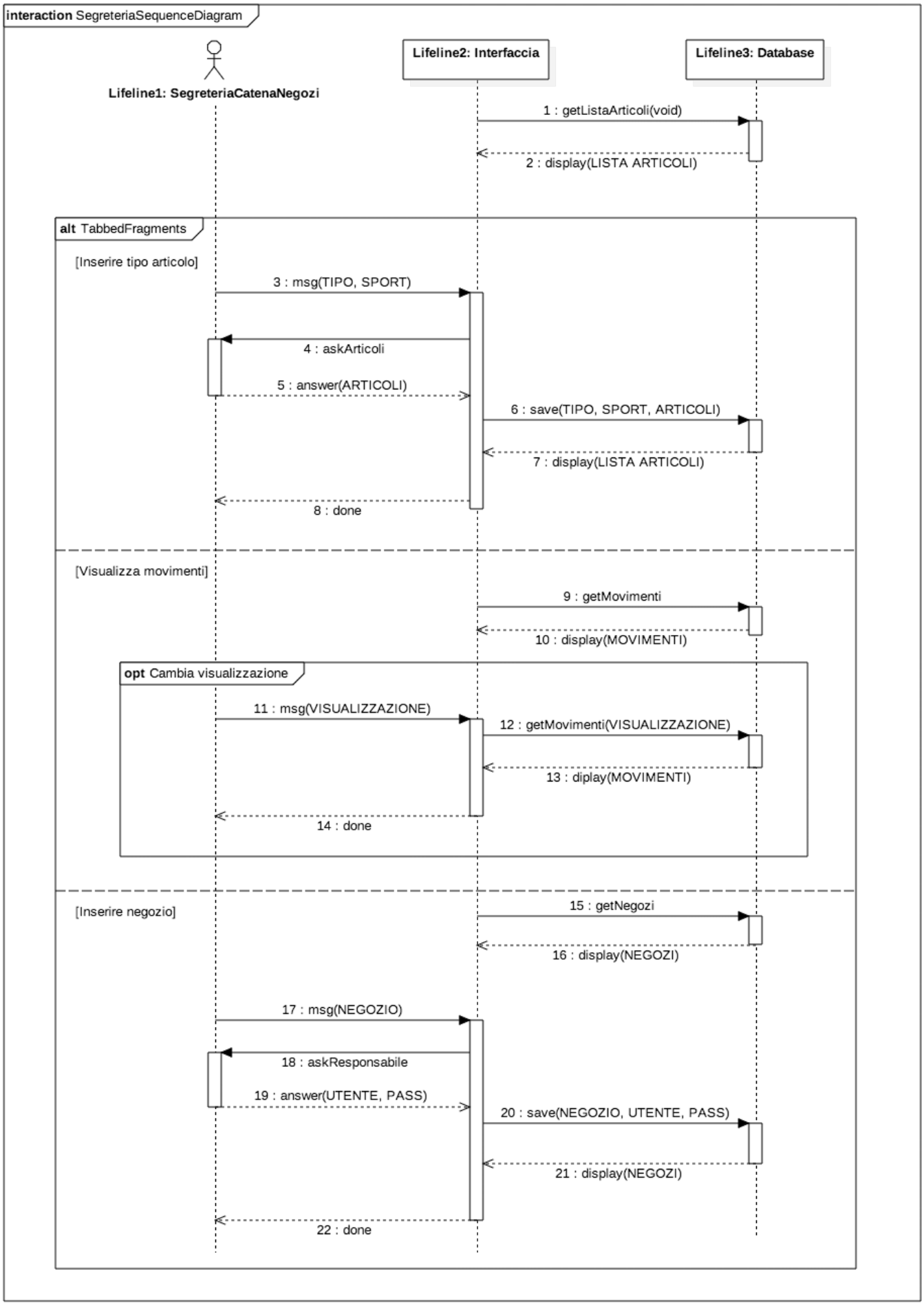
Operazioni permesse: Effettuare nuovi ordini visualizzando la lista di articoli, inserire nuovi articoli in entrata nel negozio visualizzando gli articoli nel negozio e vedere gli ordini effettuati per codice ordine o per intervallo di date

Sequence Diagram

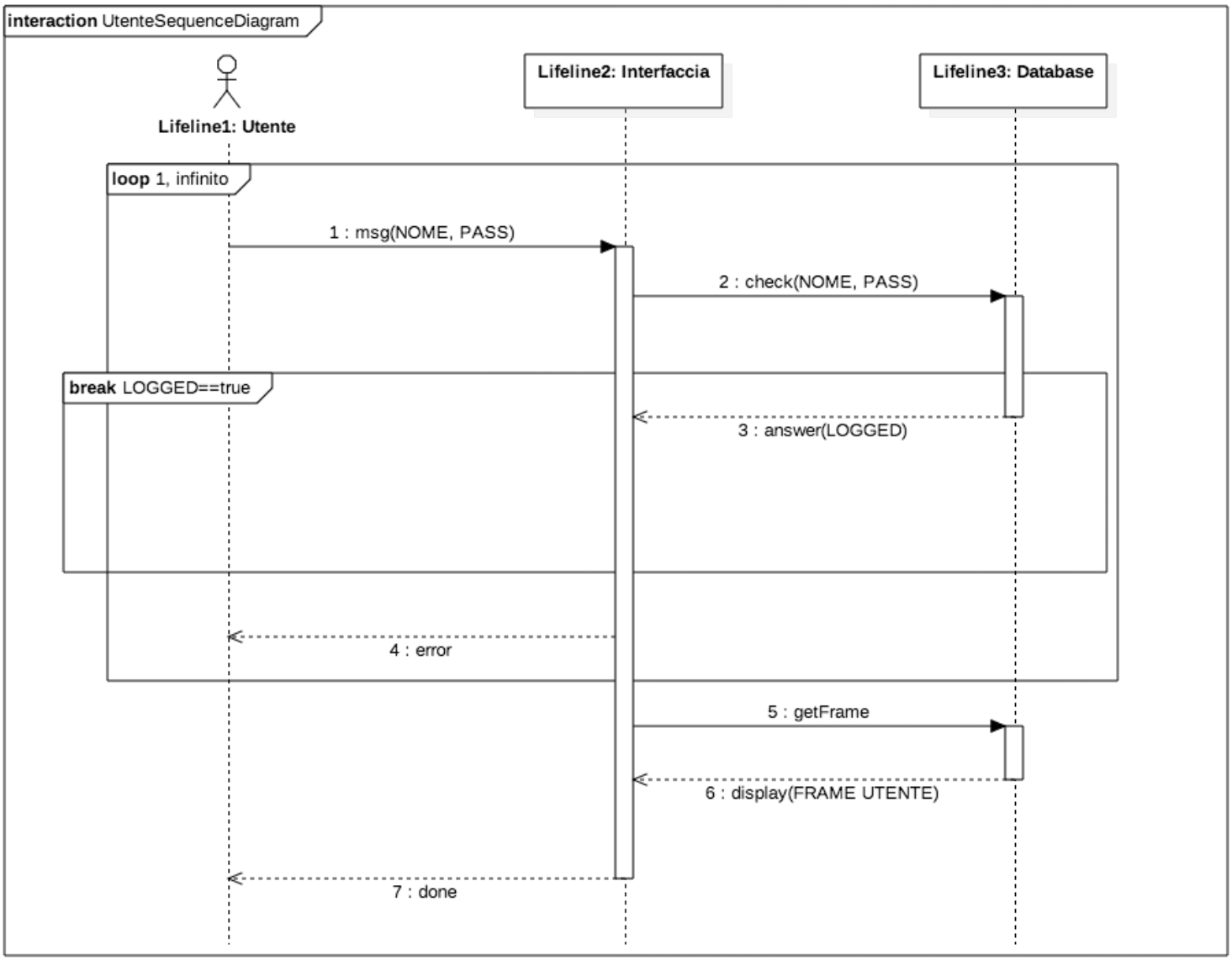
- Sequence Diagram n° 1: Magazziniere

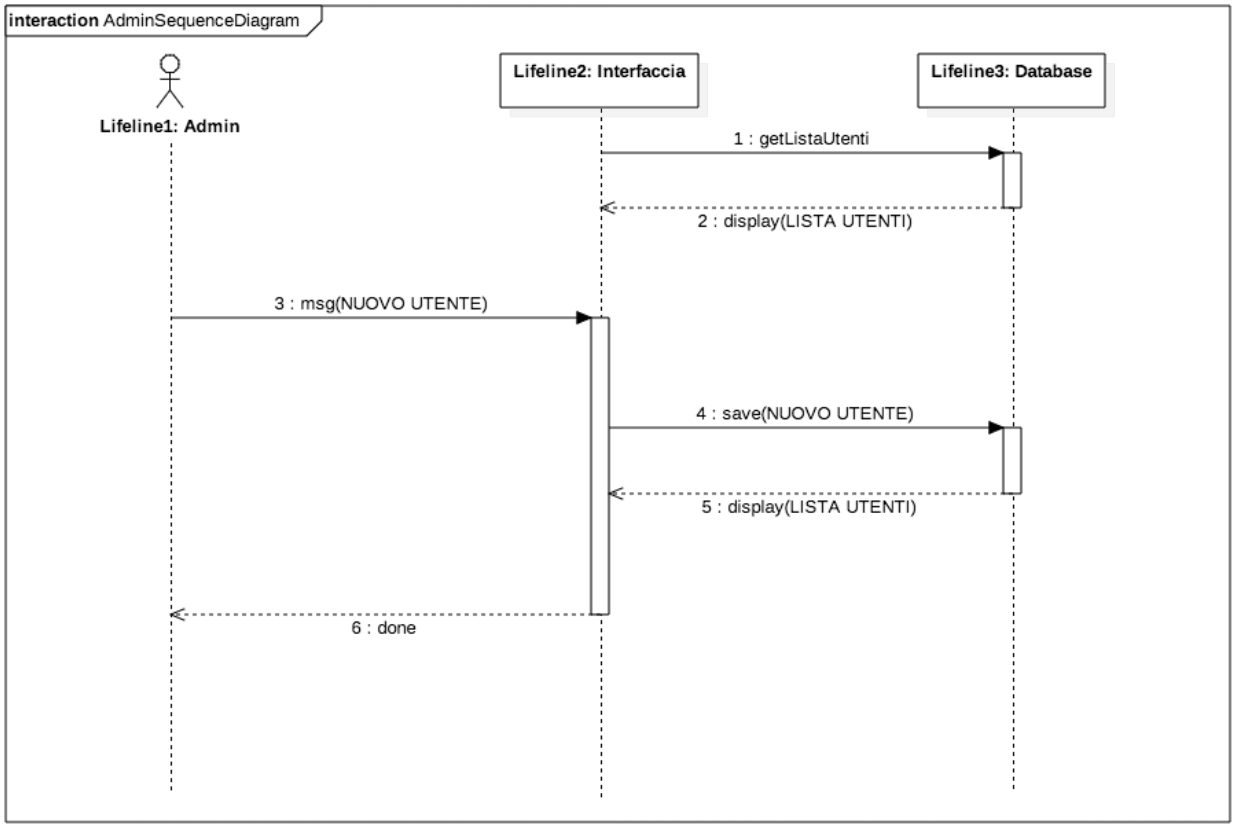


- Sequence Diagram n° 2: Segreteria amministrativa catena negozi

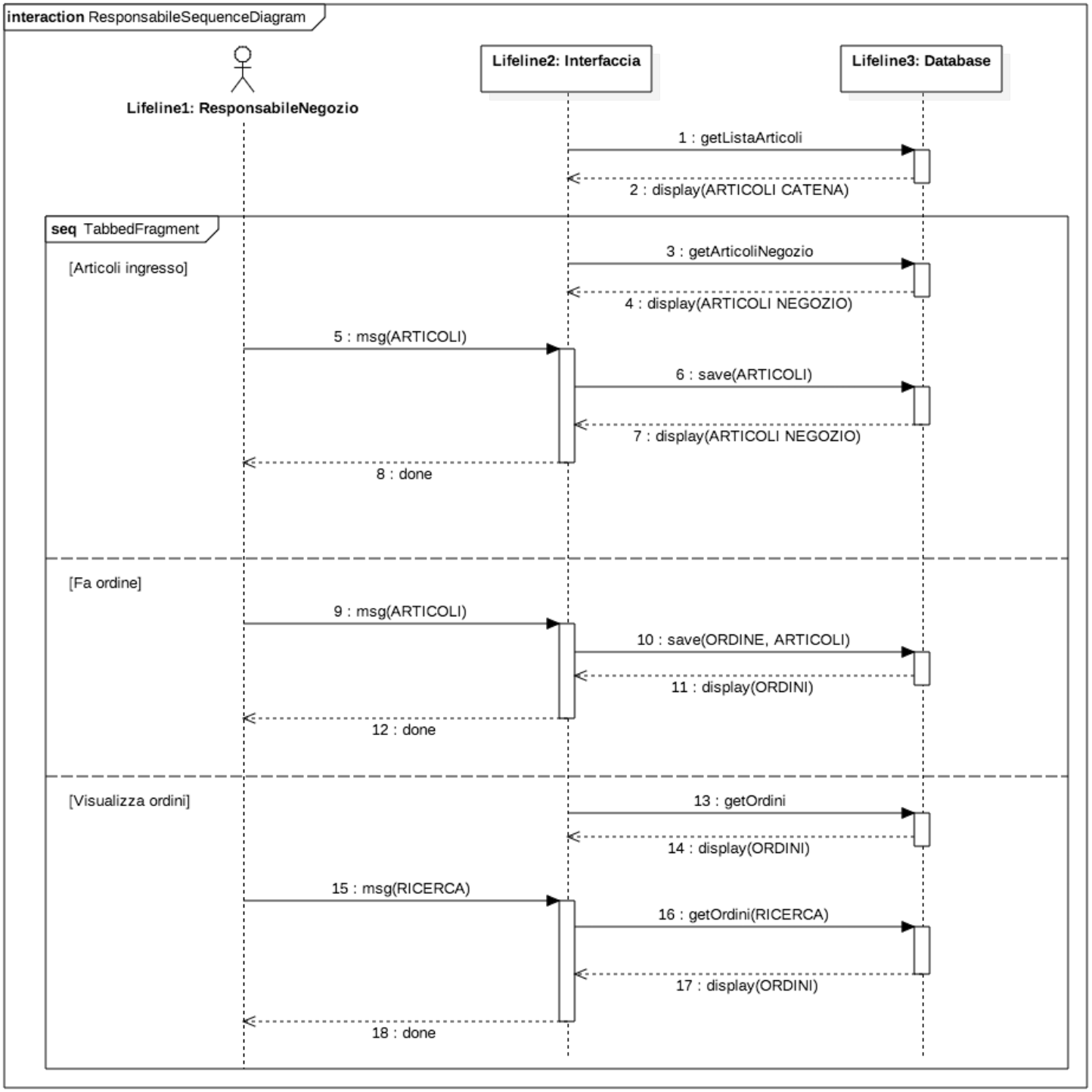


- Sequence Diagram n° 3: Utente e Admin



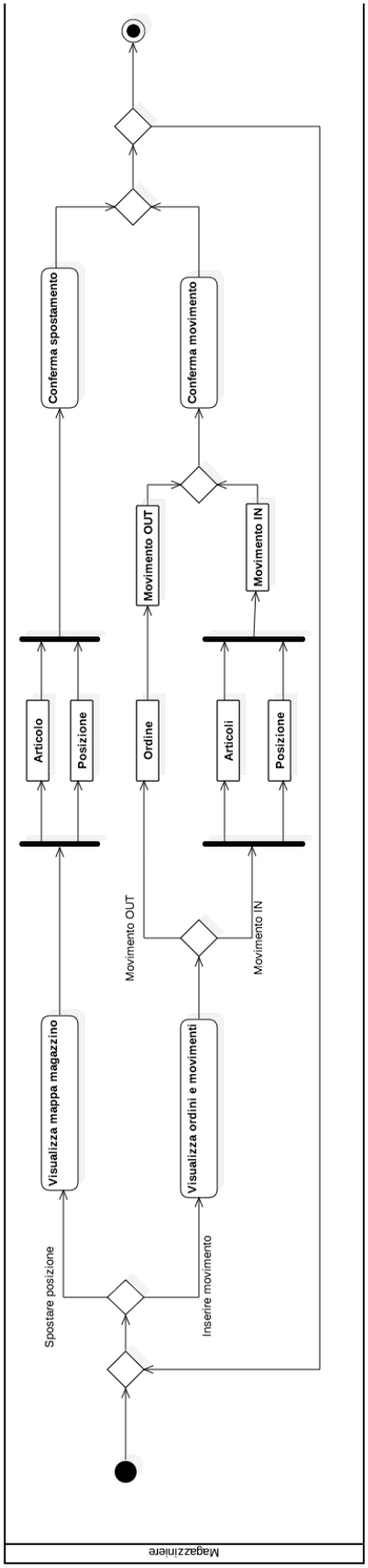


- Sequence Diagram n° 4: Responsabile Negozio

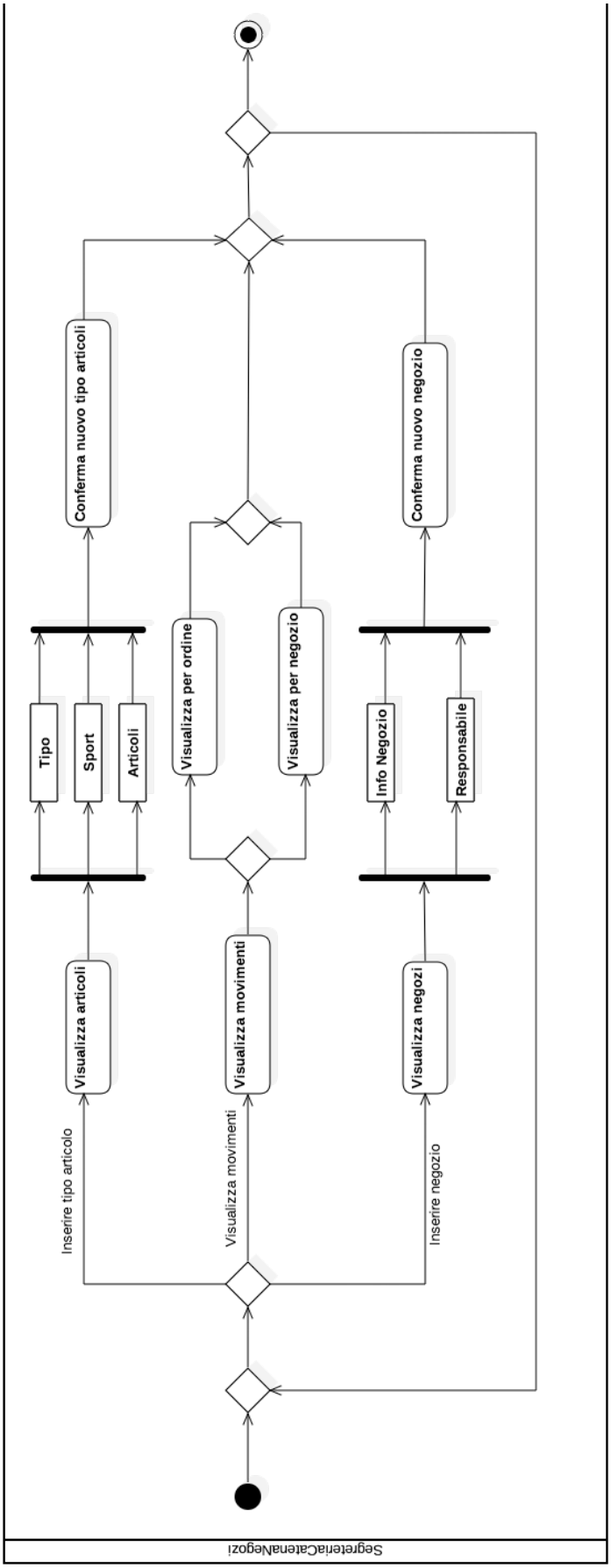


Activity Diagram

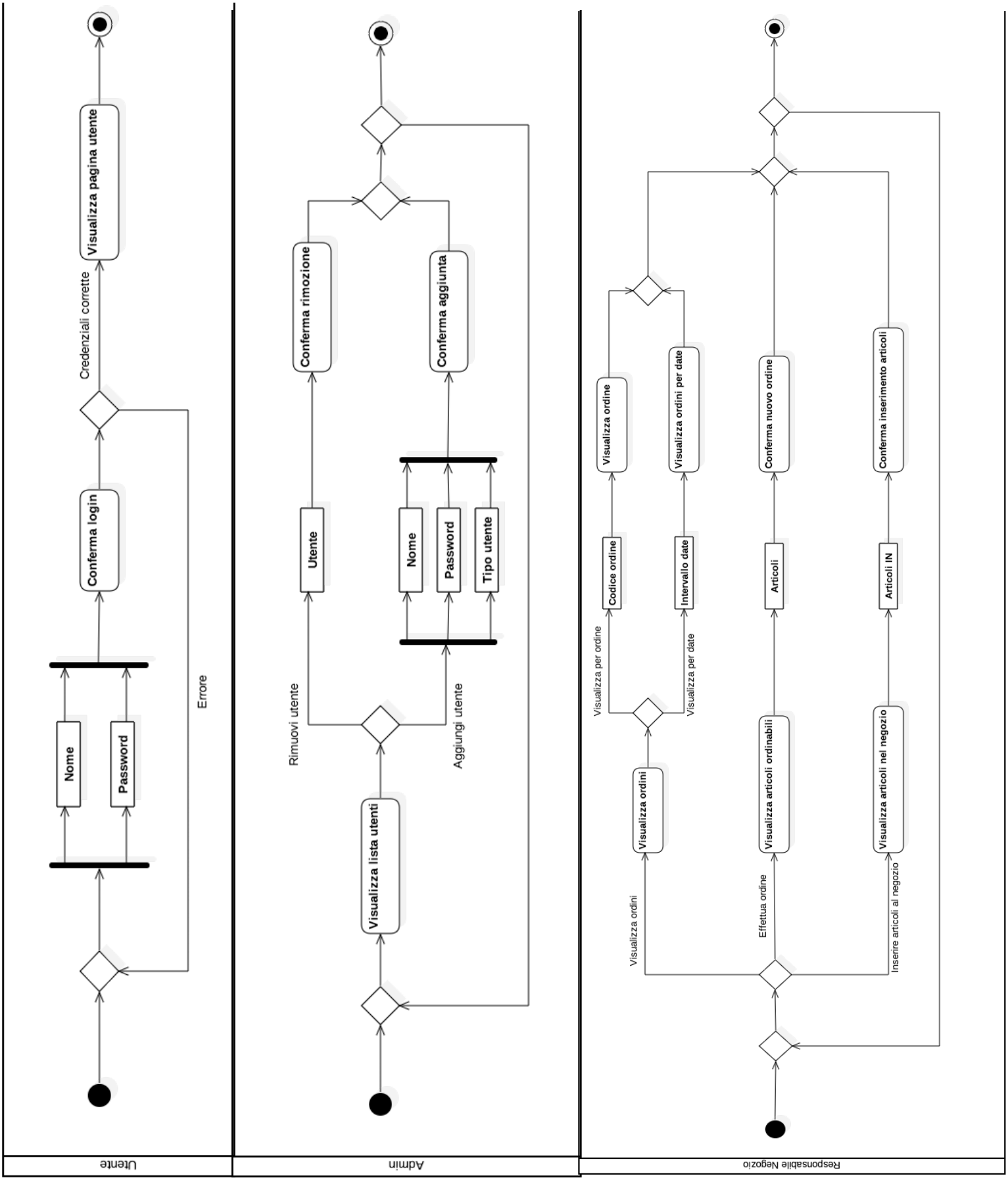
- Activity Diagram n° 1: Magazziniere



- Activity Diagram n° 2: Segreteria Amministrativa catena negozi



- Activity Diagram n° 3: Utenti e Admin
- Activity Diagram n° 4: Responsabile negozio



Introduzione al prototipo

Il prototipo è stato realizzato utilizzando interfaccia Swing e implementa 3 parti dei requisiti funzionali presentati:

- Login al sistema degli Utenti e utilizzo di un supporto Admin
- Gestione dei negozi della Segreteria Catena Negozi
- Gestione degli Ordini del Responsabile Negozio

Ogni utente potrà autenticarsi al sistema utilizzando un username e una password e il sistema capirà se si tratta di un Magazziniere, un addetto alla Segreteria Catena Negozi o di un Responsabile Negozio e caricherà dunque in automatico l'interfaccia relativa. L'interfaccia per l'Admin verrà caricata se vengono digitate le seguenti credenziali:

username: admin
password: admin

L'Admin potrà creare dei nuovi Magazzinieri o dei nuovi addetti alla Segreteria Catena Negozi. La creazione di nuovi responsabili è invece riservata alla Segreteria Catena Negozi.

Il Magazziniere in questo prototipo non è implementato, quindi se si accede con un utente Magazziniere, verrà caricata una pagina vuota.

La Segreteria Catena Negozi è implementata con 3 funzionalità:

1) Creazione nuovo articolo
2) Creazione nuovo negozio con responsabile
3) Visualizzazione tutti gli ordini dei negozi con un certo ordine

Il Responsabile Negozio è implementato con altrettante funzionalità:

1) Creazione nuovo ordine
2) Visualizzazione history ordini con filtri per codice ordine e per date
3) Aggiunta al negozio di articoli di ordini in sospeso

Modello di processo software

Il modello di processo software che abbiamo utilizzato si è basato molto sugli esami che dovevamo dare e su come volevamo organizzarci la sessione.

Visto che ognuno di noi aveva diversi esami da dare e per la maggior parte in periodi diversi, non potevamo approcciarci al progetto con un modello di processo di tipo Agile, questo perché non avevamo la possibilità di incontrarci spesso per discutere sulle specifiche e lavorare poi in pair-programming. Dunque abbiamo deciso di lavorare con un modello di processo software a **Sviluppo Incrementale Plan Driven**.

Abbiamo iniziato a lavorare a questo prototipo ad inizio Giugno. Inizialmente dovevamo fare un quadro generale del progetto, quindi ci trovavamo spesso e abbiamo cercato di ragionare insieme sui requisiti, ricavando i requisiti funzionali e procedendo subito a comporre i Use Case Diagram, i Sequence Diagram e gli Activity Diagram. Quindi già da subito avevamo ben chiaro tutti i requisiti e a grandi linee cosa dovevamo implementare.

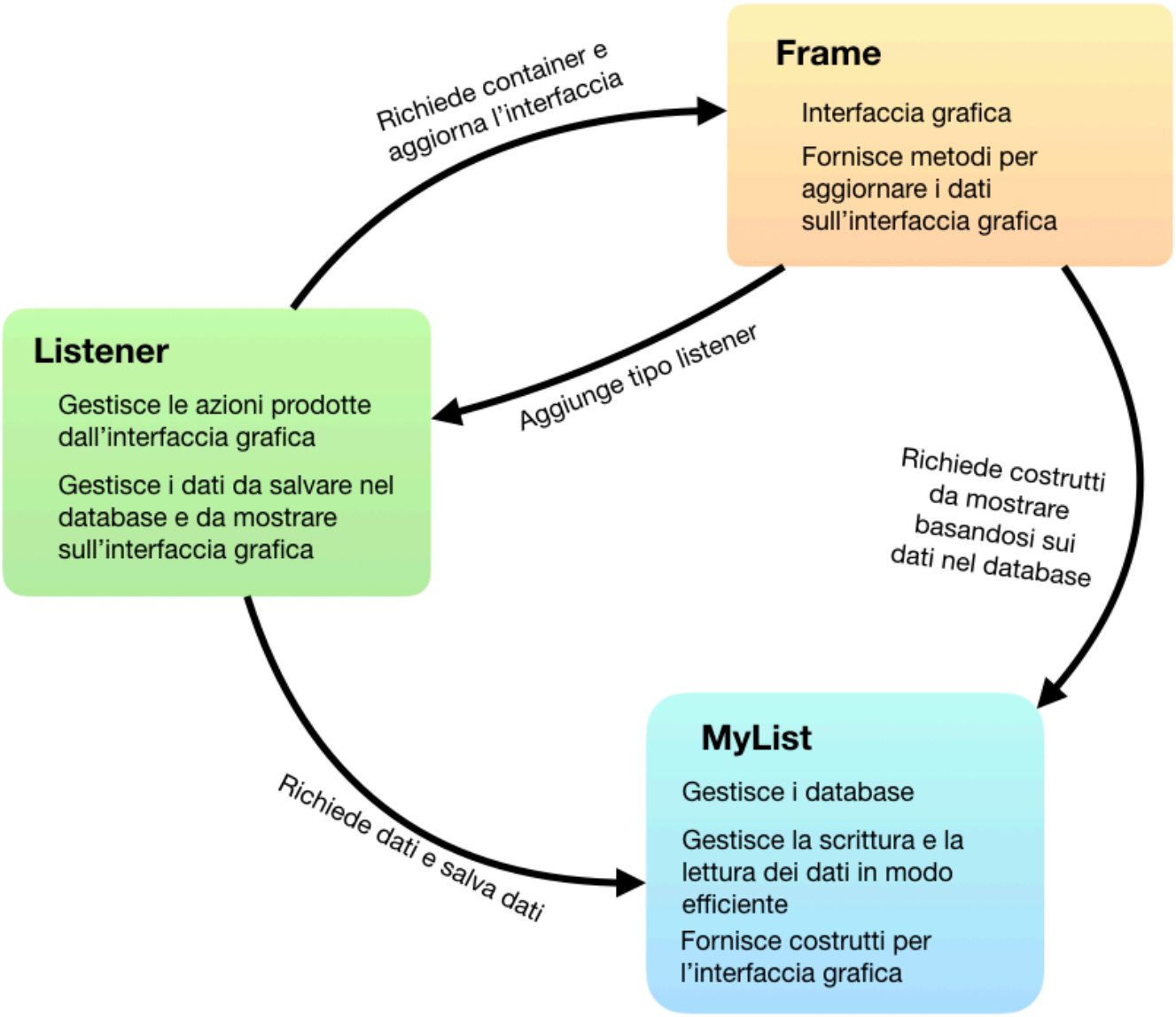
Il passo successivo è stato capire come salvare tutte le informazioni prodotte dagli utenti che avrebbero utilizzato il sistema. Abbiamo deciso di salvare i dati su file utilizzando Serializable. In questo modo potevamo scrivere su file gli oggetti che implementano Serializable.

A questo punto abbiamo fatto un breve periodo (circa una settimana) in cui abbiamo lavorato in pair-programming, applicando un modello di processo di tipo **Agile**, per poter sviluppare tutte le classi necessarie per salvare gli oggetti Serializable su file. In questo modo abbiamo ottenuto una base funzionante, che è stata chiamata MyList, per poter gestire questo database di dati su file e avremmo potuto utilizzare durante l'implementazione di tutto il prototipo.

Dopodiché ci siamo divisi le specifiche, ovvero Filippo si è occupato prevalentemente della Segreteria Catena Negozi, Mirco si è occupato prevalentemente del Responsabile Negozio e abbiamo iniziato a lavorare individualmente, fissandoci delle scadenze, in cui ci saremmo incontrati per verificare che le rispettive parti implementate comunicassero tra loro.

Progettazione Architetturale

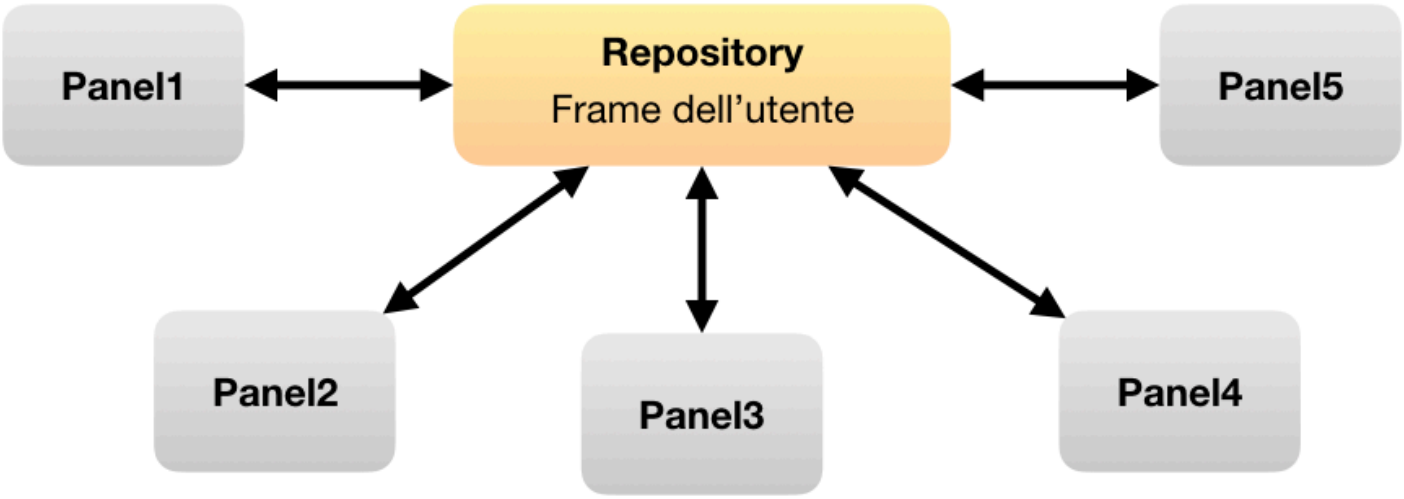
Pattern MVC



Il pattern architetturale Model View Controller è stato adottato per poter suddividere in modo semplice e indipendente più parti del software progettato.

Infatti, come spiegato nel paragrafo precedente, avevamo bisogno di realizzare una serie di classi che lavorassero in modo automatico sui database, in modo tale da avere una base su cui appoggiarsi per ottenere i dati salvati sul database in modo efficiente. Realizzare MyList è stato dunque necessario perché in questo modo abbiamo potuto suddividerci il lavoro e lavorare in modo autonomo mantenendo le stesse interazioni su MyList e quindi permettendo ai vari componenti del sistema di comunicare senza troppi accorgimenti.

Pattern Repository



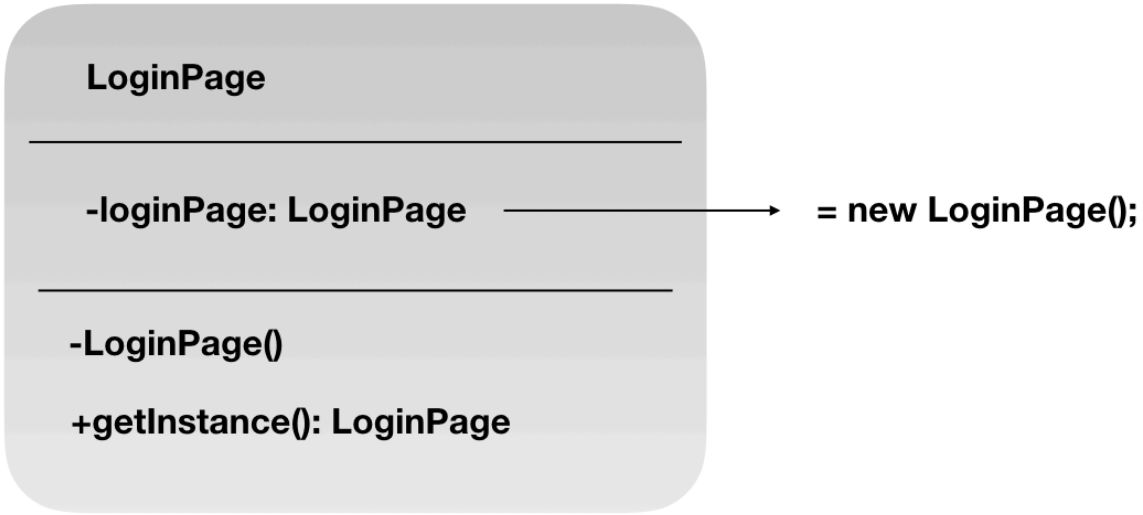
Il pattern architetturale Repository è stato adottato per poter mostrare sullo stesso Frame di interfaccia grafica più pannelli in parallelo e poterli far comunicare tra loro.

Questo pattern è stato adottato sia a livello visivo, ovvero sullo stesso Frame possono essere visualizzati più Panel, sia a livello funzionale, ovvero ogni modifica su un Panel viene propagata anche sugli altri Panel attraverso al Repository Frame.

A livello visivo è implementato tramite un JTabbedPane, il livello funzionale è implementato tramite un listener sul JTabbedPane.

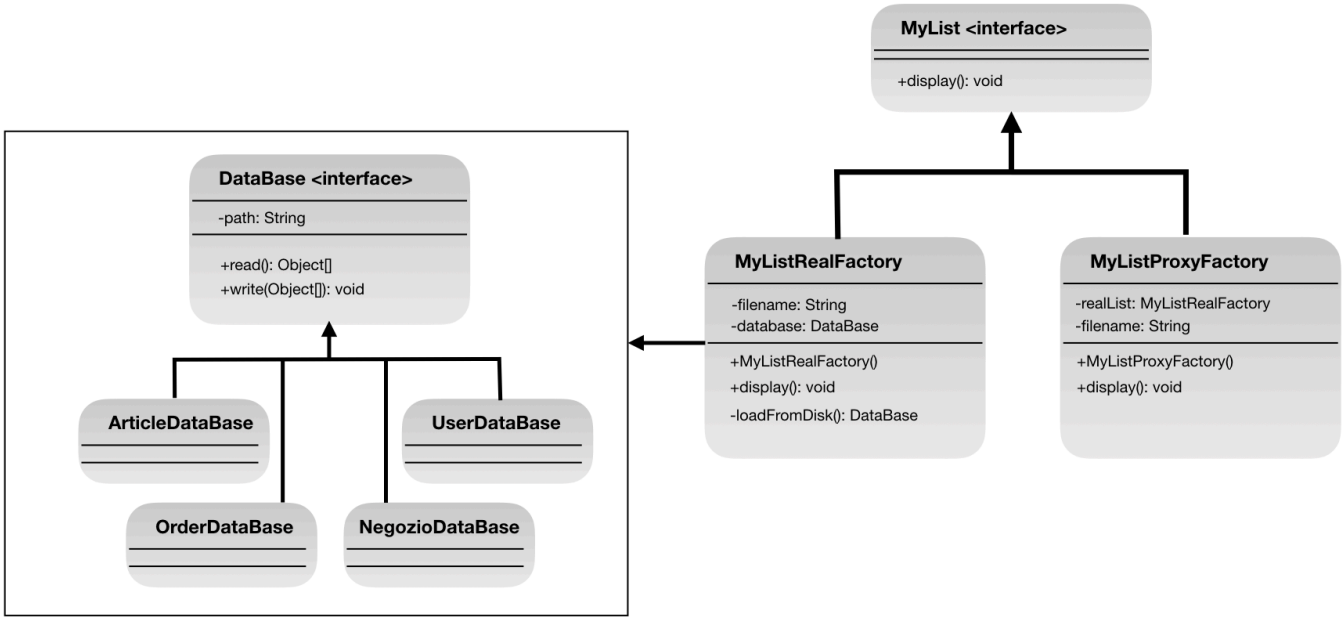
Progettazione Sistema

Singleton Design Pattern



Login Page è una finestra implementata con il design pattern Singleton, in questo modo quando il sistema viene inizializzato il frame di LoginPage viene istanziato solo una volta e non viene mai chiuso e riaperto, ma quando necessario viene reso visibile o no.

Factory e Proxy Design Pattern



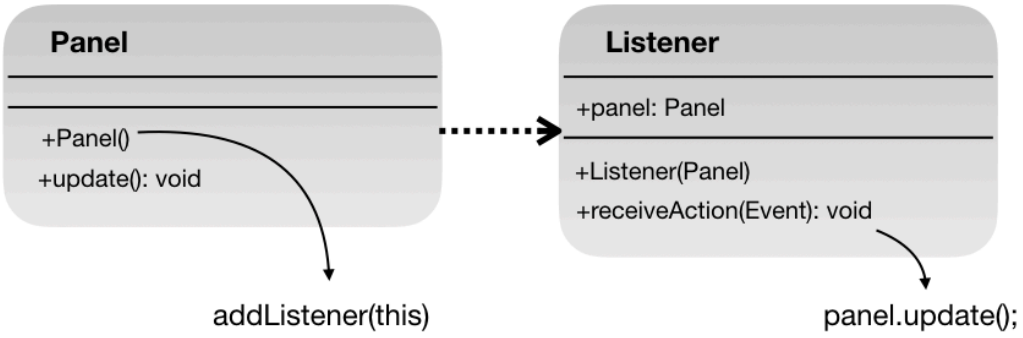
MyList gestisce tutta la parte di scrittura e lettura dai database ed è implementata con due design pattern intrecciati tra loro: FactoryPattern e DesignPatter.

La parte di FactoryPattern serve per definire il database corretto tramite una stringa, ovvero filename. Ad esempio se inserisco come filename la stringa “ArticleDataBase” allora il database che verrà caricato sarà un ArticleDataBase.

La parte di ProxyPattern per ridurre al minimo le operazioni di I/O e per caricare i dati il più “tardi” possibile. Infatti le altre classi si interfacciano solo a MyListProxyFactory, che quando viene istanziata non fa alcuna operazione di I/O. Quando viene richiamato un metodo display(), allora MyListProxyFactory si preoccuperà di istanziare MyListRealFactory che a quel punto caricherà i dati sul database giusto.

Inoltre le operazioni di I/O avvengono solo 2 volte: per caricare i dati in input, quando MyListRealFactory viene stanziata, e per scrivere i dati in output, quando la finestra viene chiusa. Per fare ciò viene usato localmente un ArrayList in cui vengono memorizzati tutti i dati del database, poi vengono fatte operazioni di add e remove su quello e in fine quando la finestra viene chiusa l’ArrayList viene scritto sul database.

Observer Design Pattern



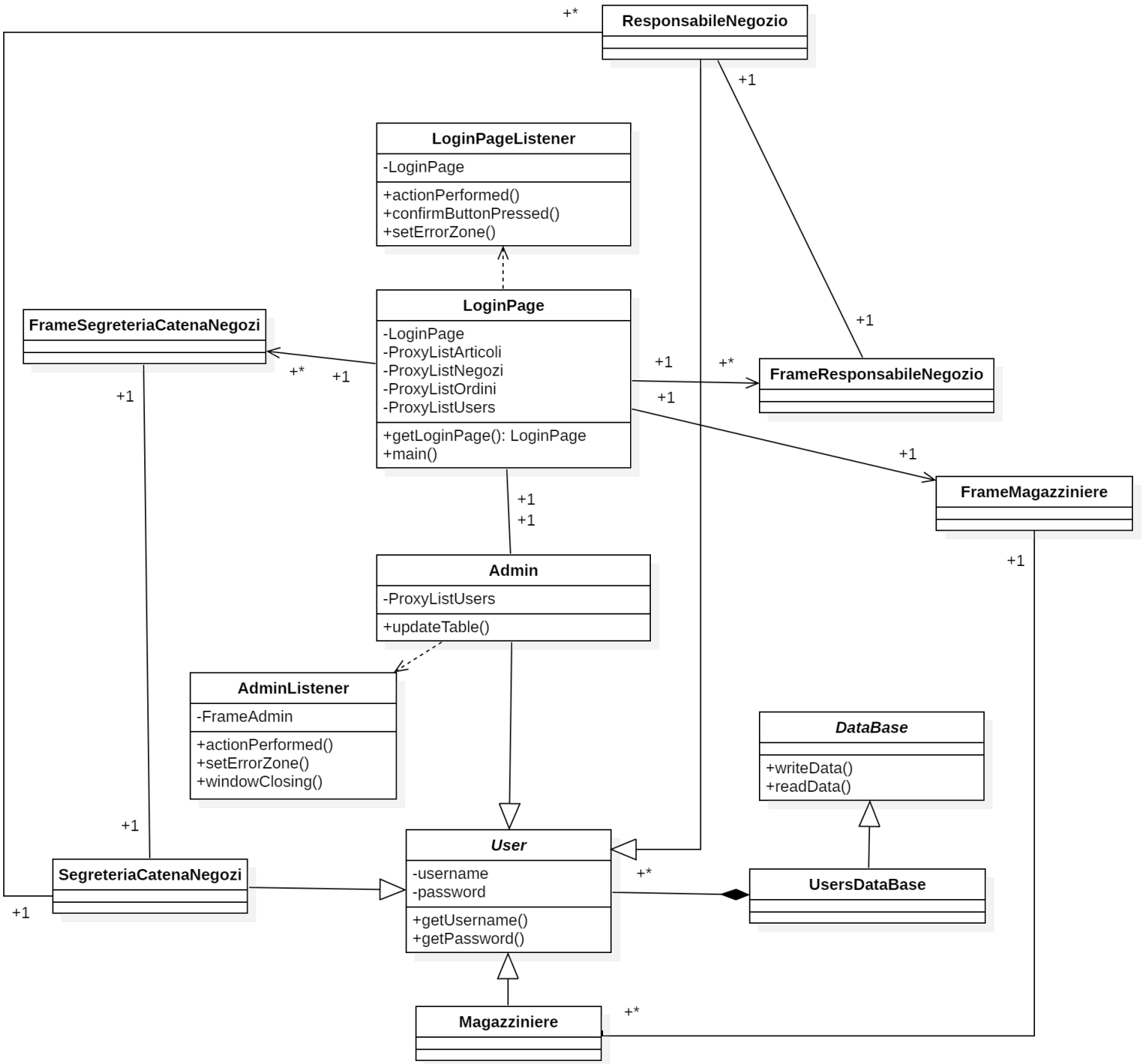
Implicitamente viene usato anche l’ObserverPattern. Infatti ogni finestra è formato da molti Panel, ai quali viene associato un Listener per poter configurare le azioni sui costrutti all’interno dell’interfaccia grafica.

In questo caso il Panel funge da Observer e il Listener funge da Subject, infatti quando Panel si registra al Listener, tramite il metodo addListener() richiamato nel costruttore, poi quando il Listener riceve un Event, allora fa un update() sul Panel aggiornando le informazioni.

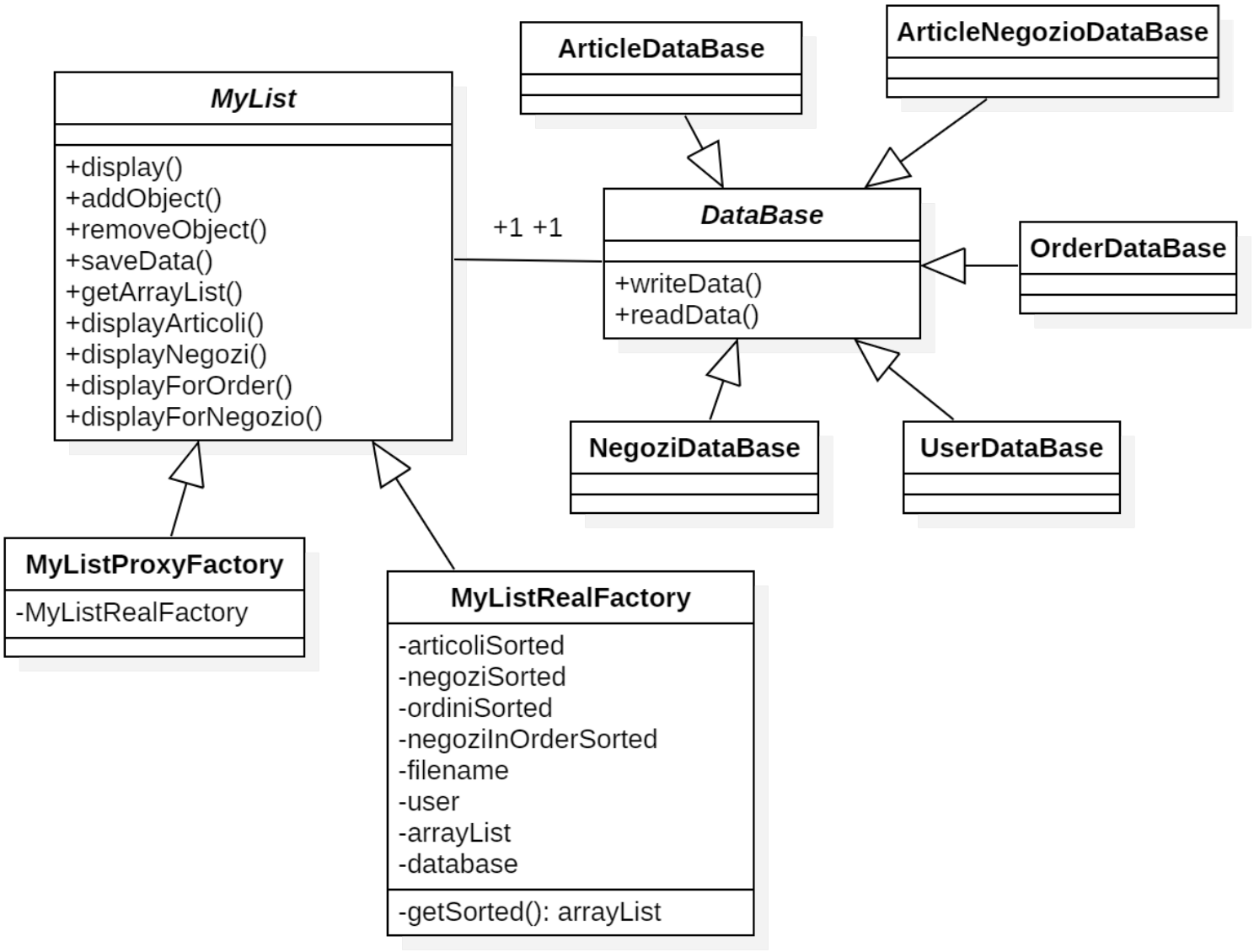
Tra il Panel e il Listener c’è una associazione 1 a 1.

Diagrammi UML del software progettato

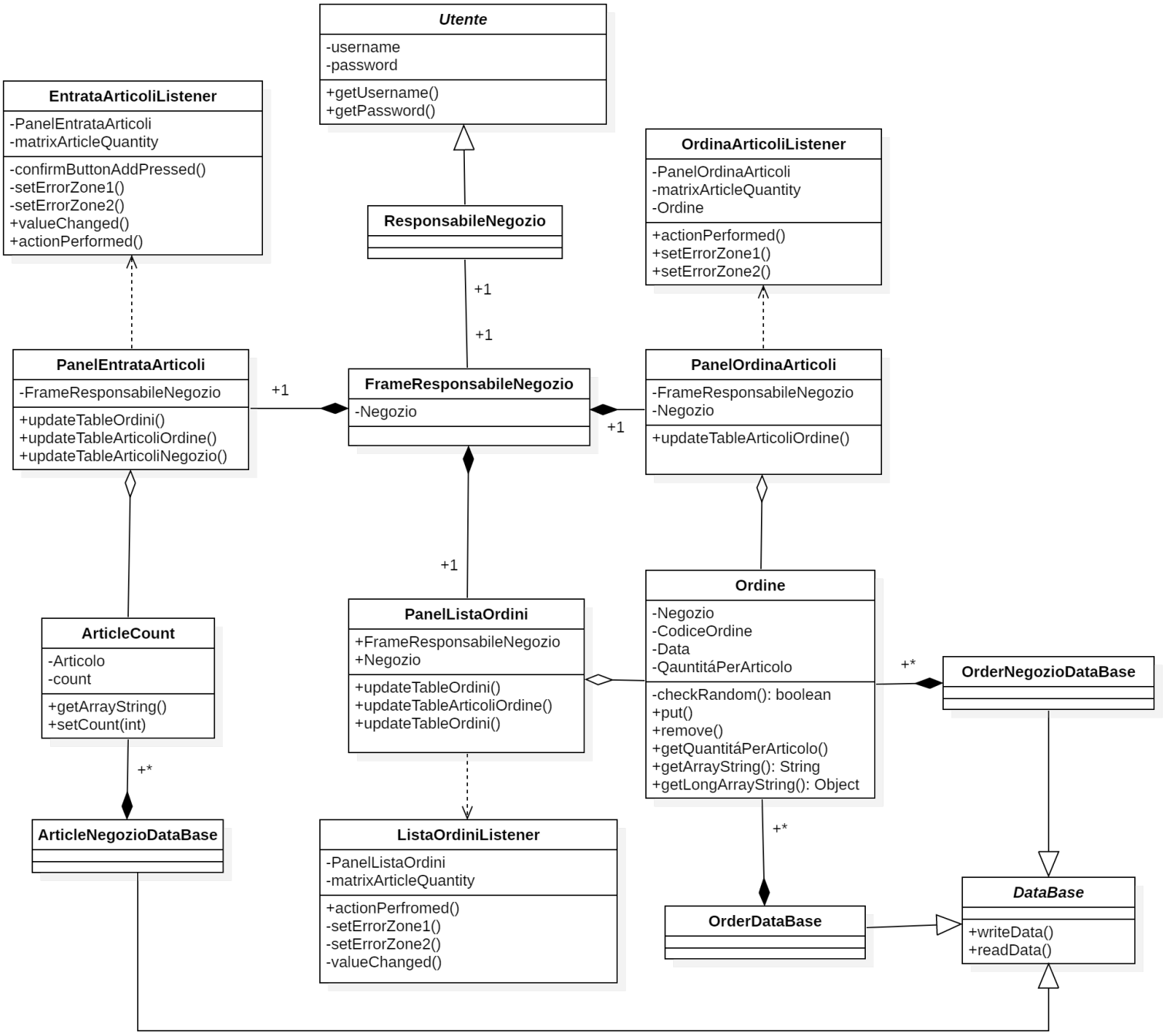
- LoginPage



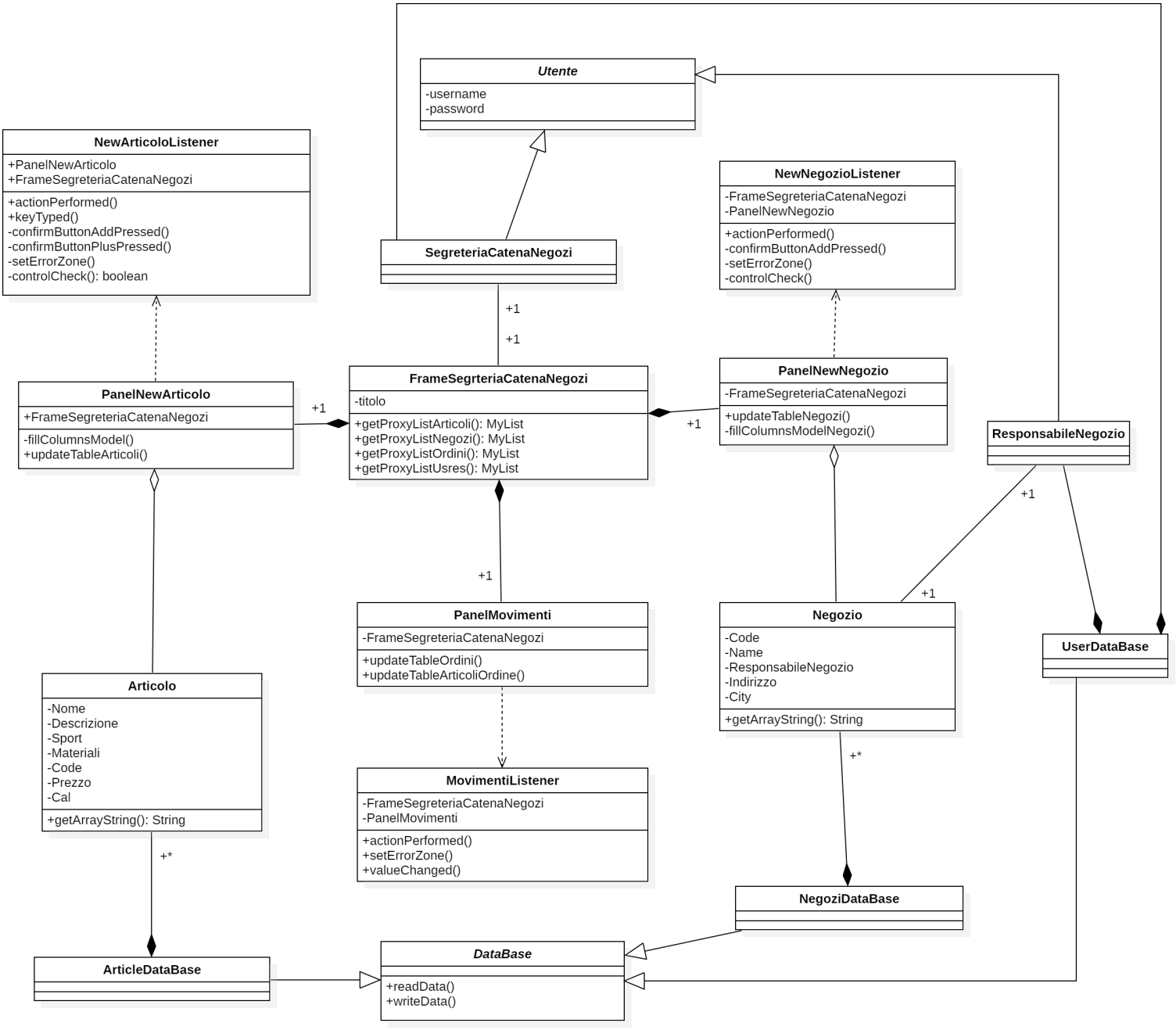
- MyList


ArticleDataBase
ArticleNegozioDataBase
MyList
+display()
+addObject()
+removeObject()
+saveData()
+getArrayList()
+displayArticoli()
+displayNegozi()
+displayForOrder()
+displayForNegozio()
+1 +1
DataBase
+writeData()
+readData()
OrderDataBase
NegoziDataBase
UserDataBase
MyListProxyFactory
-MyListRealFactory
MyListRealFactory
-articoliSorted
-negoziSorted
-ordiniSorted
-negoziInOrderSorted
-filename
-user
-arrayList
-database
-getSorted(): arrayList

- Responsabile negozio

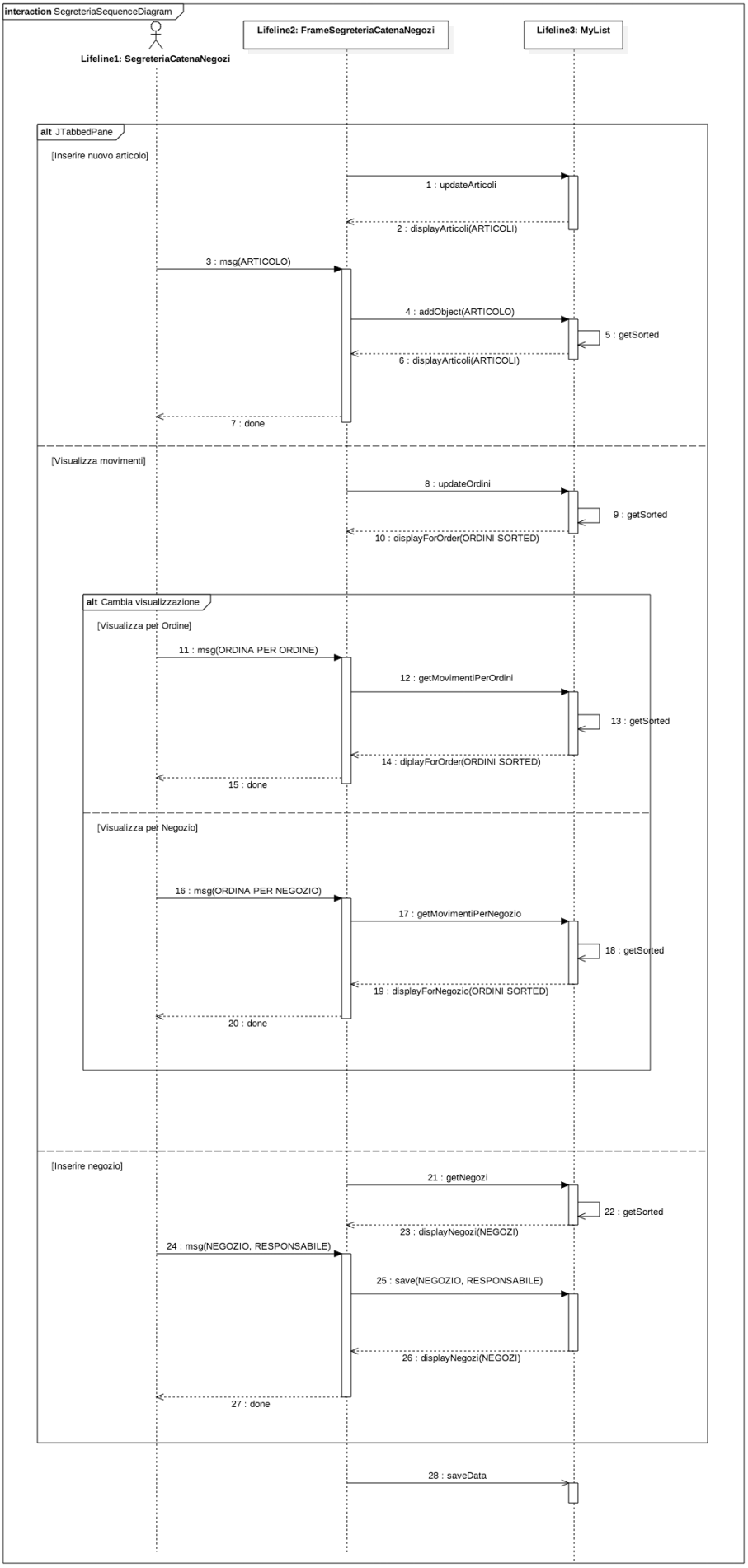


- Segreteria amministrativa catena negozi

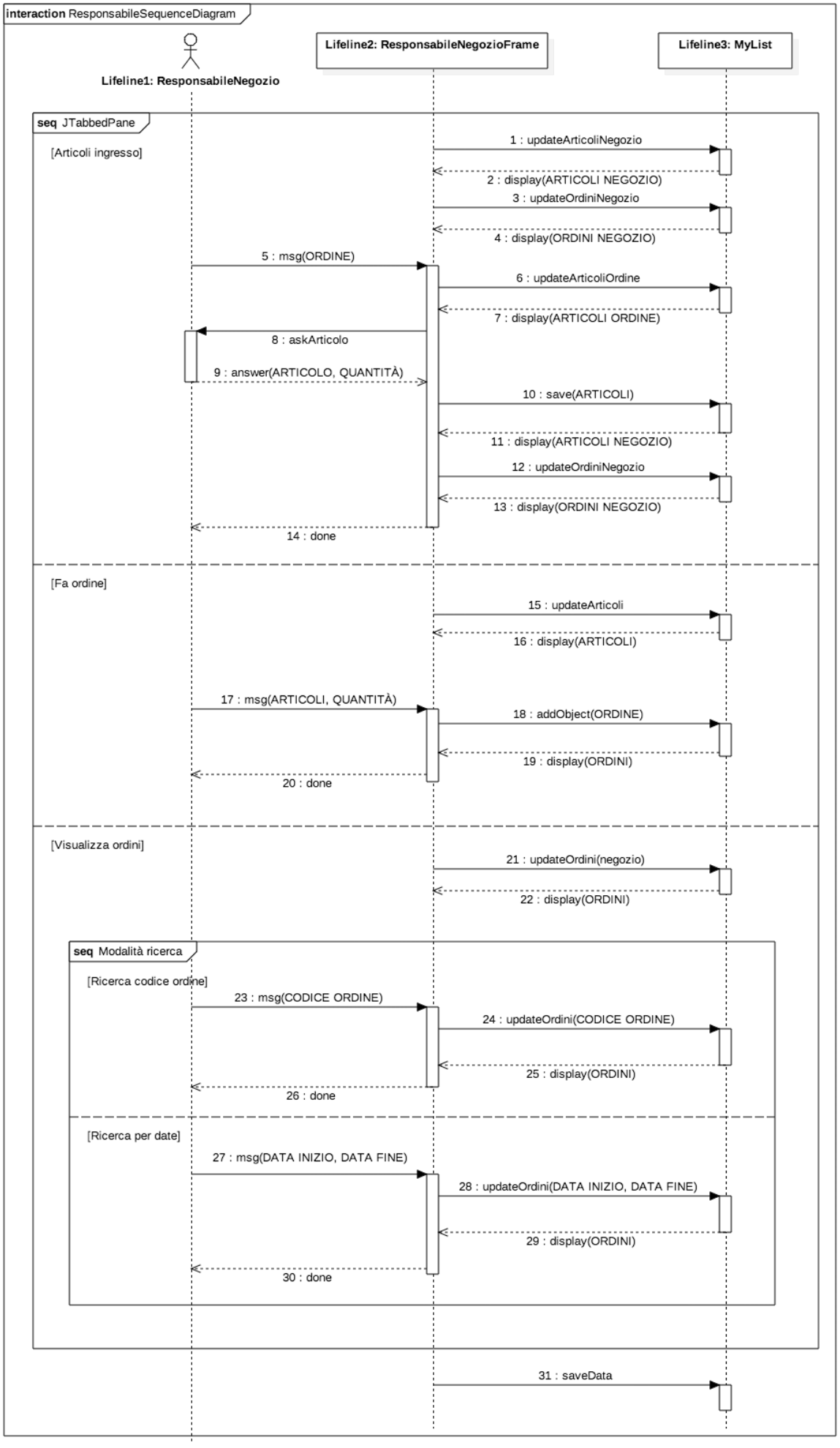


Sequence Diagram del software progettato

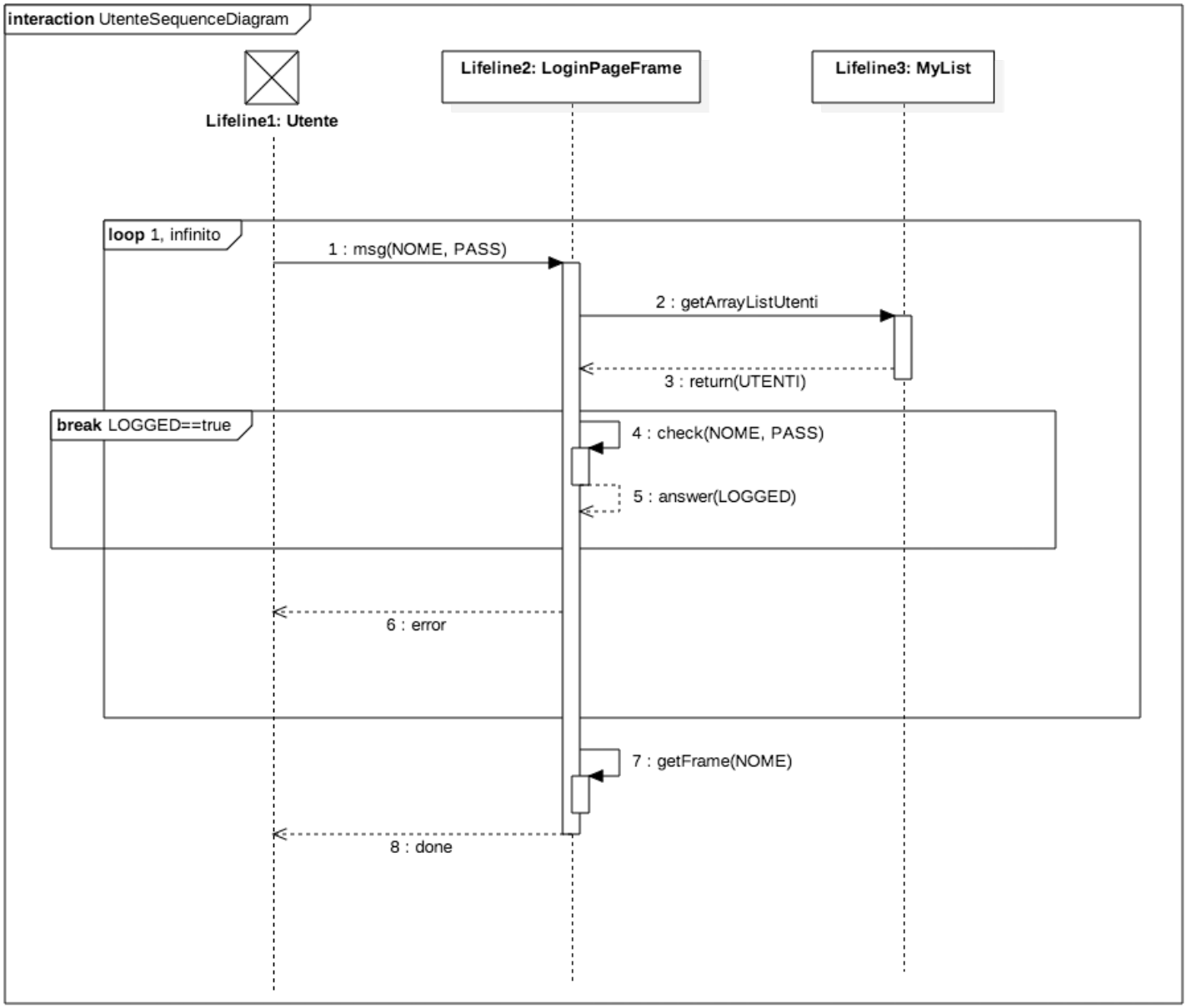
- Segreteria amministrativa catena negozi



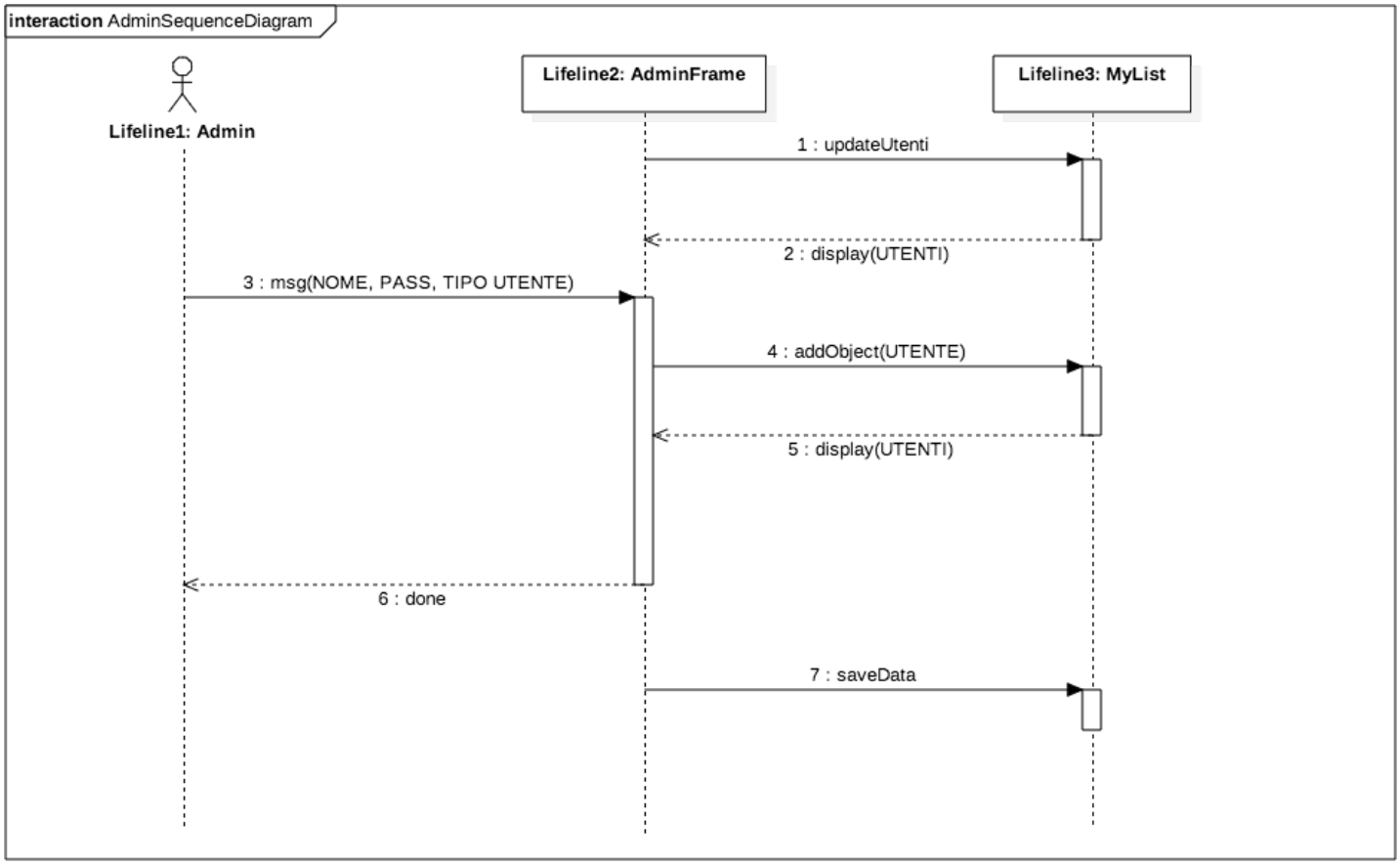
- Responsabile negozio



- Utente - Login



- Admin


interaction AdminSequenceDiagram
Lifeline1: Admin
Lifeline2: AdminFrame
Lifeline3: MyList
1 : updateUtenti
2 : display(UTENTI)
3 : msg(NOME, PASS, TIPO UTENTE)
4 : addObject(UTENTE)
5 : display(UTENTI)
6 : done
7 : saveData

Attività di test

La fase di test è stata svolta in maniera molto semplice eseguendo il programma e provando il software con diversi casi limite.

Inoltre avendo adottato un approccio Plan Driven, nei giorni di scadenza ci trovavamo, verificavamo che le due sottoparti prodotte comunicassero tra loro, dopodiché ci mostravamo a vicenda ciò che avevamo prodotto e in fine facevamo scambio di computer provando il software del proprio collega, segnalando le caratteristiche che non ci piacevano, eventuali problemi e suggerimenti.

Inoltre durante le ultime fasi del progetto sono stati molto utili suggerimenti di amici e parenti dopo avergli fatto provare il software.