Politecnico di Milano


# DD 1.1

## Progetto di Ingegneria del Software 2

SWIMv2: Small World Hypothesis Machine v2

Autori:
Bulla Jacopo
Caio Davide
Cappa Stefano

Professore:
Mottola Luca

# 2 Design Document

*Questo documento rappresenta la terza deliverable. Insieme ad esso sono stati forniti anche il RASD e il Project Planning aggiornati.*

*Lo scopo del Design Document (DD) è di mostrare in modo dettagliato la struttura del prodotto. Ovviamente, tutte le scelte effettuale nella stesura del precedente documento saranno rispettate e descritte in modo più preciso e a "basso livello", in particolare il capitolo 2.3 del RASD, cioè quello sulle assunzioni e le scelte di progettazione. Perciò, saranno introdotti i concetti in modo incrementale dalla descrizione dell'architettura fino a mostrare il comportamento e funzionamento di SWIMv2, tramite apposti schemi.*
*Saranno utilizzati: UX Diagram, BCE, ER Model, EER Model, Logical Model e Diagrammi di Sequenza.*

## Indice

# 1. Introduzione

## 1.1. Errata Corrige

Modifiche effettuate nel Project Planning 1.4:

- In seguito ad alcune modifiche effettuate in fase di Design è stato aumentato l'effort nella fase di revisione e sono state modificare alcune voci nella tabella del capitolo 4, aggiornando di conseguenza il Diagramma di Gantt.

Modifiche effettuate nel RASD 1.3:

- Aggiunta la funzione che permette di vedere la lista degli amici nel profilo dell'utente
- **Aggiunta spiegazione dettagliata del sistema dei suggerimenti di amicizie, vedi capitolo 2.3.**
- **Aggiunto nuovo scenario 7 (per l'utente) e i relativi BMPN e Sequence Diagram. Infine aggiunti due nuovi casi d'uso riferiti al nuovo scenario**

Modifiche effettuate nel Design Document 1.1:

- ER:
  Aggiunti i seguenti attributi:
  - Nella relazione Amicizia: *NotificaAlRichiedente*
  - Nell'entità Collaborazione: *NotificaAlRichiedente*
- EER:
  Aggiunti i seguenti attributi:
  - In Amicizia: *NotificaAlRichiedente*
  - In Collaborazione: *NotificaAlRichiedente*
- Logico: aggiornato per riflettere i cambiamenti dello schema EER
- UXModel:
  - Aggiornati tutti gli schemi per adattarli alle scelte implementative.
- BCE:
  - Gestione accesso rinominato in Gestione login
  - Rimosso creaUtente() dal control Gestione visitatore
  - Aggiunto il control Gestione registrazione con creaUtente()
  - In Gestione amministrazione, richiediAggiuntaAbilità() cambiato in aggiungiAbilita()
- EntityBeans:
  - Utente:
    - corretto errore di battitura, fotoProfilo:BLOB
  - Amicizia:
    - aggiunto attributo notificaAlRichiedente:boolean
    - rimosse le email e usati gli oggetti Utente: utente1, utente2
  - Collaborazione:
    - aggiunto attributo notificaAlRichiedente:boolean
    - rimosse le email e usati gli oggetti Utente: utenteRichiedente, utenteRicevente
  - Possiede
    - rimossa l'email e usato l'oggetto Utente: utente
  - PropostaAbilita
    - rimossa l'email e usato l'oggetto Utente: utente
- Diagrammi di sequenza:
  - fig 4.2: cambiato il nome del metodo confermaRegistrazione() in mostraProfilo().
- Aggiunte del capitolo 5 le spiegazioni di alcune scelte di progettazione e alcune possibili estensioni future del sistema

## 1.2. Definizioni, acronimi e abbreviazioni

- Design pattern: è "una soluzione progettuale generale a un problema ricorrente". É una descrizione o un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la progettazione e lo sviluppo del software.

- Browser: programma installato in un sistema operativo che permette l'accesso alla rete internet attraverso protocolli standard. Questo software mostra pagine web interpretando del codice ricevuto da un server. Esempi di codice con cui possono essere scritte sono: HTML, CSS, Javascript ecc…

- JEE: la Java Platform, Enterprise Edition o Java EE è una piattaforma software di calcolo Java. La piattaforma fornisce API e un ambiente di runtime per lo sviluppo e l'esecuzione di software per le imprese, compresi i servizi di rete e web, e di altre grandi applicazioni di rete a più livelli, scalabile, affidabile e sicuro. La Java EE estende la Java 2 Platform, Standard Edition (Java SE). La piattaforma incorpora un design basato in gran parte su componenti modulari in esecuzione su un server di applicazioni. Il software per Java EE è principalmente sviluppato in linguaggio di programmazione Java.

- EJB: gli Enterprise JavaBean (EJB) sono i componenti che implementano, lato server, la logica di business all'interno dell'architettura Java EE. Le specifiche per gli EJB definiscono diverse proprietà che questi devono rispettare, tra cui la persistenza, il supporto alle transazioni, la gestione della concorrenza e della sicurezza e l'integrazione con altre tecnologie, come JMS, JNDI, e CORBA. Lo standard attuale, EJB 3, differisce notevolmente dalle versioni precedenti; è stato completato nella primavera del 2006

- Session Bean: nella Java Platform è un tipo di Enterprise Bean. Una Session Bean implementa una business task ed è ospitata in un EJB container.

- Entity Bean: è un tipo di Enterprise JavaBean, un componente J2EE server-side che rappresenta dati persistenti gestiti in un database.

## 2. Architettura

Nel RASD non è stato specificato come sarà realizzato il lato client, ma **il team di sviluppo ha proposto queste due possibili soluzioni:**

- **Un'applicazione web, alla quale l'Utente potrà accedervi con un qualunque browser.**
- **Un'applicazione Java da scaricare ed installare sul proprio computer.**

Dopo un'accurata analisi, sono stati evidenziati i seguenti punti fondamentali per la scelta:

1. Un social network, per definizione, deve operare solo attraverso il web senza avere accesso al computer locale, a meno di possibili funzioni di upload/download.
2. Gli utenti sono distribuiti su molti sistemi operativi, comprese diverse distribuzioni Linux (anche se in modo meno rilevante)
3. Gli utenti di "smartphone" e "tablet" sono cresciuti così tanto da diventare un punto di riferimento nella realizzazione di un social network. Purtroppo, questi dispositivi usano sistemi operativi molto differenti, non supportano tutti gli stessi software/linguaggi di programmazione e spesso richiedono strumenti di sviluppo e requisiti differenti.

Possibili soluzioni ai punti elencati sopra:

1. **Realizzare un'applicazione web** accessibile tramite browser.
2. **Rendere indipendente il lato client dal sistema operativo (es. tramite un'applicazione Java)**
3. Sfruttare il browser (seguendo certi standard) per **rendere il lato client compatibile sui principali dispositivi mobili.**

Di conseguenza, **la migliore soluzione è la prima, cioè realizzare l'interfaccia come un'applicazione web, alla quale l'utente potrà accedervi con un qualunque browser, indipendentemente dal sistema operativo e dal tipo di dispositivo (mobile o fisso).**

**L'architettura del sistema è quella chiama Three-Tier** ("a tre strati/livelli"). Grazie ad essa vi è una separazione tra l'interfaccia utente (Presentation Tier), la logica funzionale (Business Tier) e la persistenza dei dati (Data Tier) in più strati. Essi possono essere sullo stesso server oppure su più macchine.

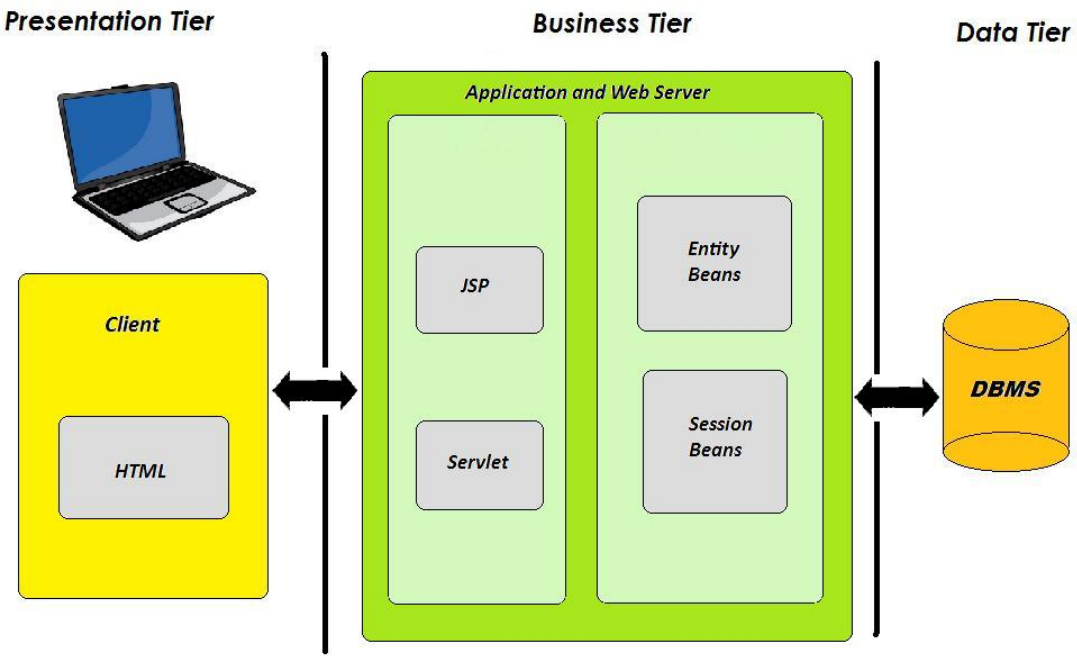


Fig. 2.1 – Architettura Three-Tier

## 2.1. Presentation Tier

Strato che si occupa della presentazione, cioè della visualizzazione dei dati. Essi sono mostrati tramite un'interfaccia grafica per far sì che l'utente possa interagire col sistema, tramite funzioni predefinite e semplici da usare. Alcune delle tecnologie utilizzate per raggiungere tale obiettivo sono JSP, HTML, JavaScript, e CSS.

## 2.2. Business Tier

Gestisce la logica, cioè fornisce l'accesso al database ed è lo strato più importante del sistema, perché costituisce il cuore della logica di business. La tecnologia utilizzata è EJB, poiché costituisce un requisito di progetto fornito dal committente.

## 2.3. Data Tier

Strato che si occupa di gestire la persistenza dei dati, cioè il database. Il team di sviluppo ne ha scelto uno di tipo relazionale, cioè MySQL.

## 2.4. Design Pattern

La suddivisione tra visualizzazione (interfaccia utente), controllo (fornisce informazioni alla GUI, gestisce la logica e utilizza i dati) e modello (rappresenta i dati) permette di ricondursi al Design Patter MVC (Model – View – Controller).

Lo schema seguente (Fig. 2.2) mostra le interazioni dirette tra i vari moduli software e quelle indirette (frecce sottili). Le prime sono effettuate tramite associazioni dirette (richieste), le seconde tramite quelle indirette, spesso con Design pattern.

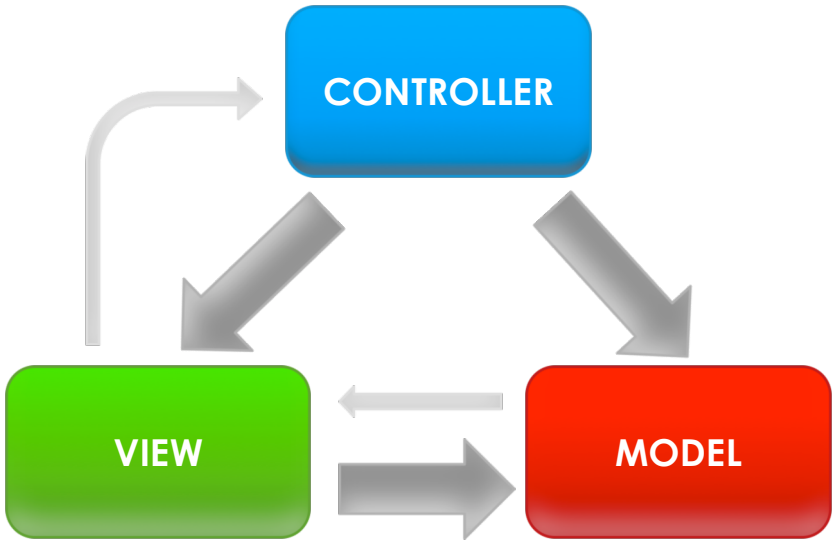


Fig. 2.2 – Architettura Three-Tear

## 3. Design

Questo capitolo descrive la fase di design, cioè la progettazione dell'intero prodotto.
Inizialmente, sarà mostrata quella concettuale, cioè la definizione ad alto livello della struttura del database. In seguito, verrà mostrata la progettazione logica, sempre riferita al database.
Terminata la realizzazione della persistenza dei dati ad alto livello, il team di sviluppo si è concentrato sul modello di navigazione. L'obiettivo è di scendere ad un livello più basso e dettagliato della descrizione del sistema in termini di visualizzazione/navigazione delle pagine web.
Per fornire una descrizione più dettagliata e per legarsi alla divisione in tre livelli descritta nel capitolo precedente, il team di sviluppo ha deciso di adottare anche i diagrammi di analisi (BCE).

Solo dopo a tutte queste fasi e dopo aver definito nel dettaglio ogni comportamento del sistema, sarà mostrata la progettazione di SWIMv2 basata su JEE (rispettando il vincolo imposto dal committente), cioè definendo gli Entity Bean e i Session Bean.

Infine, per una maggiore chiarezza ed evitare interpretazioni errate di alcune scelte di progettazione del team di sviluppo, sono stati allegati alcuni Diagrammi di Sequenza.

**Attenzione: è ovvio che ogni passo della fase di design debba rispettare tutte le scelte progettuali effettuate nei precedenti documenti, in particolare il RASD e il capitolo sulle assunzioni (vedi 2.3).**

## 3.1. Progettazione concettuale: ER Model

La progettazione concettuale di un database consiste nell'individuazione delle entità e delle relazioni. Essendo il livello più alto con cui è rappresentata una base di dati, è facilmente leggibile ed interpretabile.

Per comprenderne il significato sono necessarie alcune informazioni sulla notazione usata:

- Entità: identifica uno o più oggetti o classe con caratteristiche comuni. Nello schema ER è rappresentata come un rettangolo con all'intero un nome.
- Relazione: identifica un legame tra due o più entità. E' rappresentata come un rombo con all'interno un verbo o un sostantivo, collegato alle entità che deve "legare".
- Attributo: elemento contenuto in un'entità. Nello schema ER è rappresentato come un'ellisse/cerchio collegato ad una entità ed un nome.
- Attributo chiave: attributo che identifica univocamente gli elementi appartenenti alla stessa entità. Spesso è indicato come un cerchietto nero o col nome sottolineato.

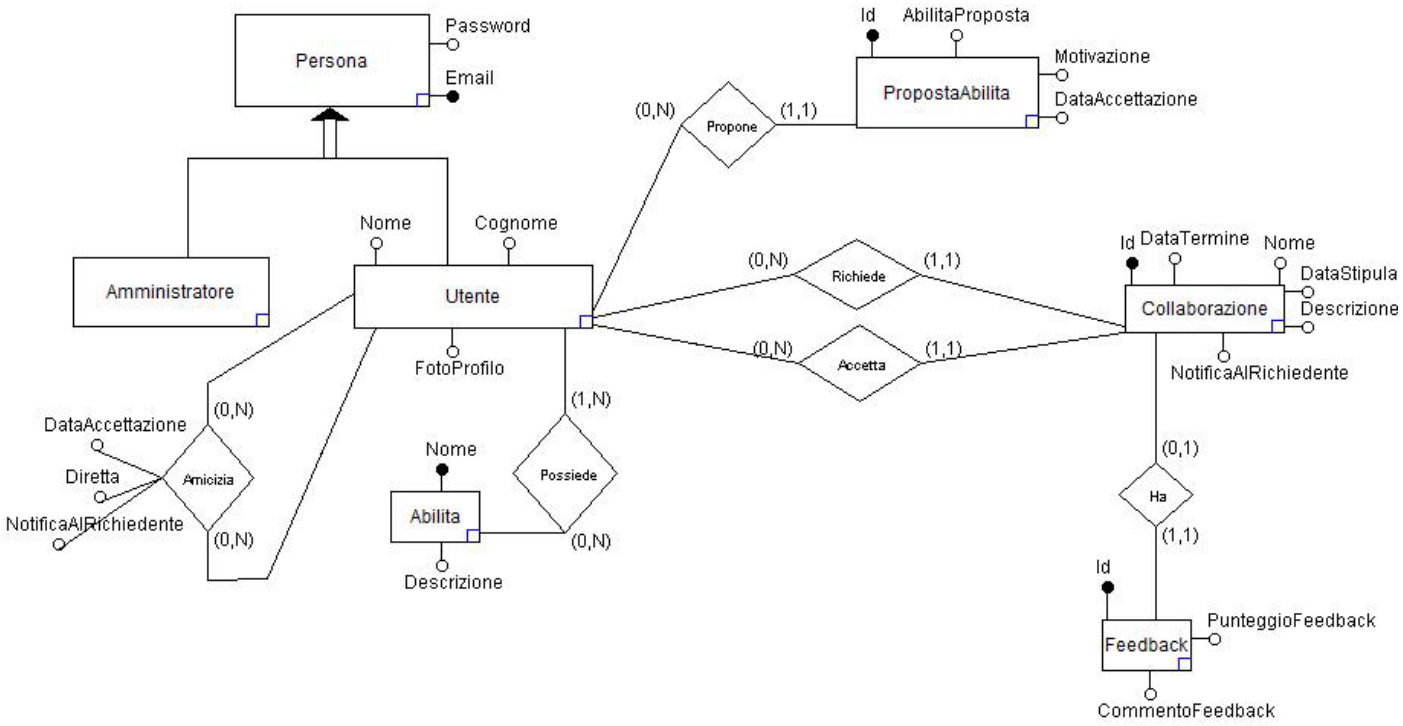


Fig. 3.1 - Schema ER

*3.1.1. Entità*

**Persona**

Entità generica che non sarà rappresentata negli schemi successivi. L'unico scopo è di esprimere che "Utente" e "Amministratore" condividono alcuni attributi.
Essa è identificabile grazie all'Email, che per definizione è univoca. Per tale motivo è stata scelta come unica chiave primaria (indicata sullo schema ER con il cerchio nero).

Gli attributi sono:

- *Email*: chiave primaria
- *Password*: parola chiave che permette il Login, insieme all'Email.

**Utente**

Entità che rappresenta l'utilizzatore registrato del sistema ed estende l'entità Persona. **E' importante notare che la classe Visitatore non è stata rappresentata nel database, perché non è necessario memorizzare i suoi dati.**
Gli attributi sono:

- *Cognome*: cognome reale dell'Utente. Esso sarà utilizzato per eseguire le ricerche di utenti.
- *Nome*: nome reale dell'Utente. Esso sarà utilizzato per eseguire le ricerche di utenti.
- *FotoProfilo*: Immagine che l'Utente può caricare in fase di registrazione e/o modificare in seguito tramite la funzione di modifica del profilo.

**Amministratore**

Entità senza attributi che estende Persona. **Si è scelto di non rappresentare la relazione di "dichiarazione" delle Abilità, perché "Amministratore" è un'entità che ha accesso a tutto il database ed inoltre non vi è alcune necessità di associare l'Email all'Abilità dichiarata inizialmente o approvata in seguito.**

**Collaborazione**

Entità che rappresenta la collaborazione tra due Utenti, cioè il rapporto di cooperazione che costituisce la richiesta di aiuto.
Poiché vi possono essere moltissime Collaborazioni, soprattutto in vista di un'estensione del sistema, si è scelto di assegnare un identificativo univoco progressivo.
Gli attributi sono:

- *Id*: identificativo progressivo univoco che costituisce la chiave primaria.
- *Nome*: attributo che rappresenta il nome della Collaborazione. Esso è utilizzato solo per mostrare la lista delle collaborazioni e renderle facilmente identificabili.
- *Descrizione*: descrizione della Collaborazione, cioè una sorta d'invito ad accettare che l'Utente può fare verso quello che possiede le abilità ricercate.
- *DataStipula*: data che rappresenta l'istante temporale di apertura o stipula della Collaborazione, cioè nel momento in cui l'Utente ricevente risponde in modo positivo alla richiesta.
- *DataTermine*: data che rappresenta l'istante temporale di fine Collaborazione, cioè quando l'Utente ritiene sia il momento di terminare il rapporto con l'altro Utente per lasciare, eventualmente, un Feedback.
- *NotificaAlRichiedente*: attributo necessario per gestire le accettazioni delle richieste di collaborazione.

**Feedback**

Entità che rappresenta il commento di Feedback associato alla Collaborazione.
Gli attributi sono:

- *Id:* **Identificativo univoco del Feedback che corrisponde all'Id della Collaborazione.**
- *CommentoFeedback*: eventuale commento dell'Utente richiedente per spiegare meglio l'esperienza avuta durante la Collaborazione con l'Utente ricevente.
- *PunteggioFeedback*: valore intero da 1 a 5 che rappresenta il voto. Esso sarà utilizzato per calcolare il punteggio di Feedback dell'Utente ricevente.

**Abilita**

Entità che rappresenta l'Abilità, cioè la capacità di svolgere una determinata mansione o di conoscere un certo argomento. E' una parte fondamentale del sistema, perché tramite essa avviene la ricerca di aiuto, e rende identificabili le conoscenze che ha un particolare Utente.

**Attenzione: l'attributo Descrizione di Abilità potrà essere modificato solo dall'Amministratore al momento dell'accettazione e non dall'Utente.**

Gli attributi sono:

- *Nome*: nome dell'Abilità che costituisce la chiave primaria.
- *Descrizione*: eventuale descrizione dell'Abilità. Attributo opzionale che può essere utile per spiegare la differenza di alcune Abilità, che potrebbero avere nomi ambigui o facili da confondere.

**PropostaAbilita**

Entità che rappresenta la proposta di un Utente di aggiungere un'Abilita nell'insieme generale delle Abilità. Quando il sistema crea la proposta di Abilita su richiesta dell'Utente, non viene generata una nuova Abilità, ma il sistema ne richiede la creazione all'Amministratore. Egli può scegliere se confermarla, inserendola nell'insieme generale delle abilità e di conseguenza il sistema creerà l'Abilità associata.

Questo significa che **non ci sarà una relazione diretta** tra PropostaAbilita e Abilita.

Gli attributi sono:

- *Id*: identificativo progressivo univoco che costituisce la chiave primaria insieme alla relazione con l'entità Utente.
- *AbiltaProposta*: nome dell'abilità proposta. E' una semplice Stringa e non costituisce nessuna chiave esterna.
- *Motivazione*: eventuale commento dell'Utente per motivare la proposta di aggiunta.
- *DataAccettazione*: indica la data di accettazione della proposta d'abilità da parte dell'Amministratore. Se assume valore NULL significa che la proposta è ancora in attesa di una risposta. Se l'Amministratore la rifiuta, il sistema rimuove l'informazione da PropostaAbilità

*3.1.2. Relazioni binarie*

- **Amicizia** l'Utente "è amico di" da 0 a N Utenti (amici), mentre l'Utente (amico) è a sua volta "amico di" da 0 a N Utenti. Cioè ogni Utente ha da 0 a N amici e questi amici hanno a loro volta da 0 a N amici.
  Questa relazione ha i seguenti attributi:
  - *DataAccettazione*: indica la data di accettazione dell'amicizia. Se assume valore NULL significa che la richiesta è ancora in attesa di una risposta. Quando l'utente rifiuta una richiesta di amicizia, il sistema rimuove l'informazione sull'Amicizia.
  - *Diretta*: Indica se è un'amicizia stretta in modo diretto, su richiesta di un utente, o indiretta, su proposta del sistema.
  - *NotificaAlRichiedente*: attributo necessario per gestire le accettazioni delle richieste di amicizia.

- **Possiede**: l'Utente "possiede" da 1 a N Abilità, mentre l'Abilità "è posseduta" da 0 o N Utenti.
- **Richiede**: l'Utente "inizia" da 0 a N Collaborazioni, mentre la Collaborazione "è iniziata" da un solo Utente. **Questa relazione identifica l'Utente *richiedente*.**
- **Accetta:** l'Utente "partecipa" da 0 a N Collaborazioni, mentre la Collaborazione "è composta" da un solo Utente. **Questa relazione identifica l'Utente *ricevente*.**
- **Ha:** la Collaborazione può "avere" un Feedback, mentre il Feedback, se esiste, "è riferito" ad una ed una sola Collaborazione.
- **Propone:** l'Utente "propone" da 0 a N ProposteAbilità, mentre la PorpostaAbilità "è proposta" da uno ed un solo Utente.

## 3.2. Progettazione concettuale: EER Model

Durante la progettazione concettuale, il team si sviluppo ha scelto di realizzare anche l'EER (Enhanced ER o Extended ER) che "estende" l'ER Model.
In questo schema sono messe in evidenza le cardinalità delle relazioni tramite una simbologia differente rispetto l'ER. Inoltre, **questo modello mostra più nel dettaglio le entità, perché indica il tipo degli attributi e le chiavi esterne.**

Simboli delle entità:

- Chiave gialla: indica la chiave primaria
- Rombo rosso: indica una chiave esterna (non utilizzato)
- Rombo bianco: attributo che può essere NULL
- Rombo azzurro: attributo che non può essere NULL

Le relazioni si distinguono in:

- "Nonidentifying Relationship" quando la chiave esterna non fa parte della chiave primaria della tabella figlia (non utilizzata in questo schema).
- "Identifying Relationship" quando la chiave esterna fa parte della chiave primaria nella tabella figlia.

Cardinalità delle relazioni:

- Indica una cardinalità "1"
- Indica una cardinalità "N"

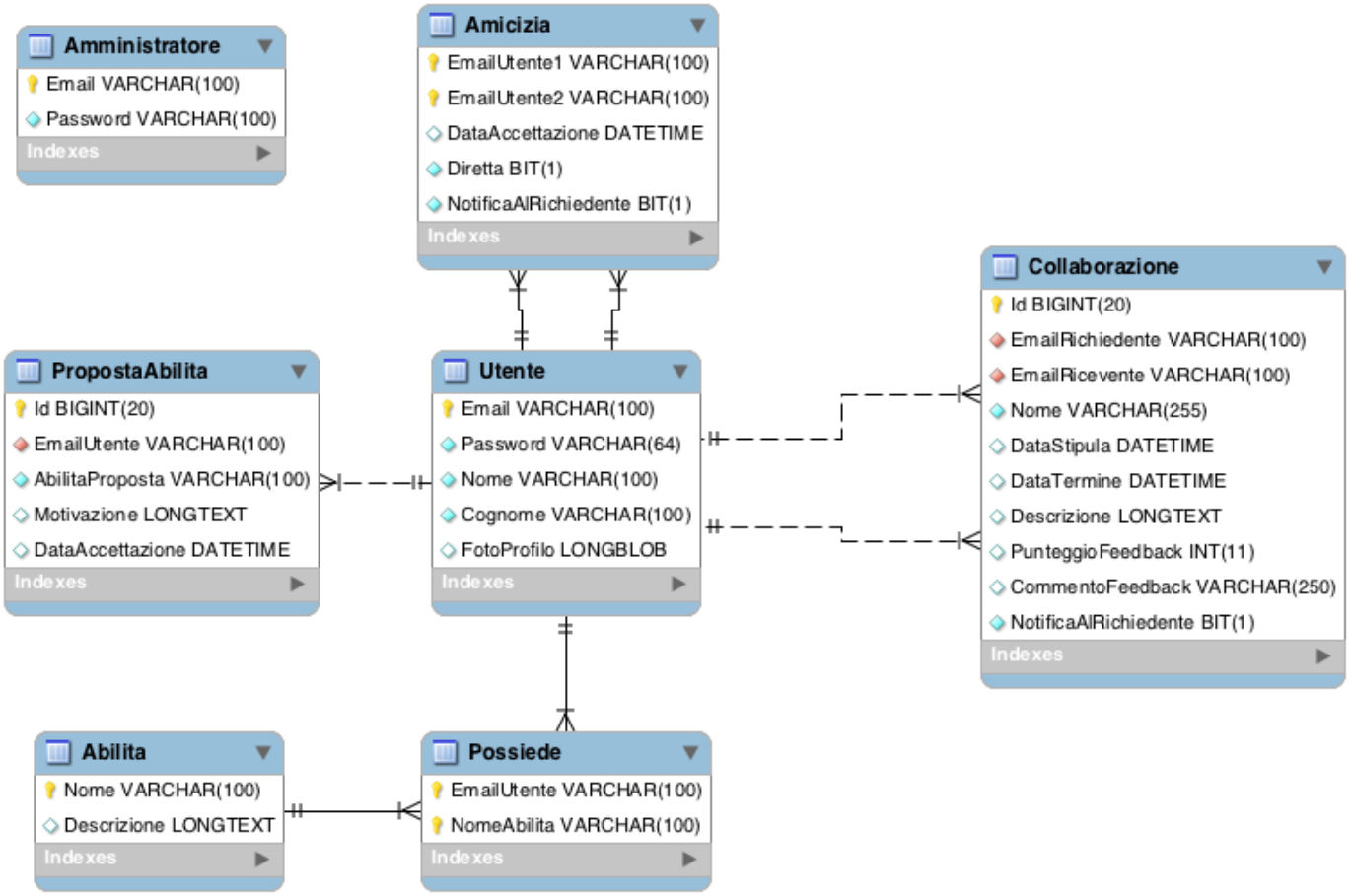


Fig. 3.2 - EER Model

Una differenza fondamentale dallo schema ER precedente è la "fusione" di Feedback in Collaborazione, poiché ad una Collaborazione è associato al massimo un solo Feedback. Quindi, si può aggiungere nella Collaborazione i due attributi "PunteggioFeedback" e "CommentoFeedback" dell'entità Feedback. Nel caso esso non sia stato rilasciato, assumeranno valori NULL, altrimenti vi sarà associato il punteggio e l'eventuale commento. In seguito saranno spiegati meglio tutti gli attributi di questa entità.

Ora segue la spiegazione dettagliata di ogni entità dello schema EER e dei loro attributi.

**Amministratore**

Entità che rappresenta l'Amministratore, cioè una persona unica che si occupa di gestire il social network.
Poiché "eredita" tutti gli attributi di Persona (nello schema ER), avrà i seguenti attributi:

- *Email*: chiave primaria che costituisce, insieme alla Password, le informazioni di Login dell'Amministratore.
- *Password*: parola chiave che permette il Login, insieme all'Email. Ovviamente non può valere NULL.

Come già specificato nello schema ER, **si è scelto di non rappresentare nessuna relazione con l'Amministratore perché è un'entità che "vede" e "opera" sull'intero sistema, senza che nessuna sua informazione sia tracciabile. In futuro sarà comunque possibile modificare il database per introdurre alcuni sistemi di sicurezza.**

**Utente**

Come già anticipato, rappresenta l'Utente, cioè chi si è registrato a SWIMv2.
Non vi è alcuna differenza con lo schema ER a parte il fatto che "eredita" gli attributi da Persona, proprio come l'Amministratore. Per maggiore completezza sono riportati tutti gli attributi dell'entità.
Attributi:

- *Email*: chiave primaria che costituisce, insieme alla Password, le informazioni di Login dell'Utente.
- *Password*: parola chiave che permette il Login, insieme all'Email. Ovviamente non può valere NULL.
- *Cognome*: cognome reale dell'Utente. Esso sarà utilizzato per eseguire le ricerche di utenti. Ovviamente non può valere NULL.
- *Nome*: nome reale dell'Utente. Esso sarà utilizzato per eseguire le ricerche di utenti. Ovviamente non può valere NULL.
- *FotoProfilo*: Immagine che l'Utente può caricare in fase di registrazione e/o modificare in seguito tramite la funzione di modifica del profilo. Tale attributo è stato rappresentato tramite il tipo BLOB, perché indica un file.

**Possiede**

Entità che indica quando un'Abilità è presente nell'insieme personale delle Abilità di un Utente.
La chiave primaria è costituita da entrambi gli attributi.
Attributi:

- *EmailUtente*: chiave esterna, parte della chiave primaria, che indica quale Utente possiede una certa Abilità. Ovviamente, tale attributo si riferisce a "Email" della tabella "Utente".
- *NomeAbilita*: chiave esterna, parte della chiave primaria, che indica il nome dell'Abilità posseduta da un Utente. Ovviamente, tale attributo si riferisce a "Nome" della tabella "Abilità".

**Abilità**

Entità che rappresenta la capacità di svolgere una determinata mansione o di conoscere un certo argomento. Tramite essa avviene la ricerca di aiuto e rende identificabili le conoscenze che ha un particolare Utente. **Attenzione: l'attributo Descrizione di Abilità potrà essere modificato solo dall'Amministratore al momento dell'accettazione e non dall'Utente.**
Attributi:

- *Nome*: chiave primaria che rappresenta il nome dell'Abilità, cioè quella stringa da selezionare durante l'aggiunta di un'abilità nell'insieme personale/generale delle Abilità o durante la ricerca di aiuto.
- *Descrizione*: eventuale descrizione dell'Abilità. Attributo opzionale che può essere utile per spiegare la differenza di alcune Abilità, che potrebbero avere nomi ambigui o facili da confondere.

**PropostaAbilita**

E' la stessa entità dello schema ER con l'aggiunta dell'Email dell'Utente che propone un'Abilità all'Amministratore.

Attributi:

- *Id*: identificativo progressivo univoco che costituisce la chiave primaria.
- *EmailUtente*: identificativo dell'Utente che ha proposto l'Abilità. Tale attributo è la chiave esterna verso l'Email Utente.
- *AbiltaProposta*: nome dell'abilità proposta. E' una semplice Stringa e non costituisce nessuna chiave esterna. Ovviamente, non potrà mai essere NULL.
- *Motivazione*: eventuale commento dell'Utente per motivare la proposta di aggiunta. Questo attributo può assumere valore NULL.
- *DataAccettazione*: rappresenta la data di accettazione della proposta. Quest'attributo è NULL quando non è ancora stata accettata dall'Amministratore. **Se la proposta fosse rifiutata, l'intera tupla sarebbe cancellata e di conseguenza non è necessario utilizzare un attributo "rifiutata".**

**Amicizia**

E' la stessa entità dello schema ER con l'aggiunta delle Email dei due Utenti che stringono amicizia, perché sono chiavi esterne.

Attributi:

- *EmailUtente1*: identificativo dell'Utente che **richiede** amicizia. Tale attributo è la chiave esterna verso l'Email Utente ed insieme all'altro EmailUtente2 costituisce la chiave primaria.
- *EmailUtente2*: identificativo dell'Utente **a cui è proposta** l'amicizia. Tale attributo è la chiave esterna verso l'Email Utente ed insieme all'altro EmailUtente1 costituisce la chiave primaria.
- *DataAccettazione*: rappresenta la data di accettazione della richiesta d'amicizia. Questo attributo è NULL quando non è ancora stata accettata. **Se l'amicizia venisse rifiutata, l'intera tupla sarà cancellata e di conseguenza non è necessario utilizzare un attributo "rifiutata".**
- *Diretta*: attributo che assume valore '1' se l'amicizia è stata richiesta in modo diretto,'0' in modo indiretto.
- *NotificaAlRichiedente*: attributo necessario per gestire le accettazioni delle richieste di amicizia. Assume valore '1' per indicare che l'utente richiedente (EmailUtente1) ha già visto il messaggio in cui è specificato che l'utente ricevente (EmailUtente2) ha accettato la richiesta di amicizia,'0' altrimenti.

**Collaborazione**

E' la stessa entità dello schema ER precedente con la fusione di "Feedback".

Gli attributi sono:

- *Id*: identificativo progressivo univoco che costituisce la chiave primaria.
- *EmailRichiedente*: rappresenta l'email dell'utente che ha avviato la collaborazione, essa è una chiave esterna collegata alla tabella Utente.
- *EmailRicevente*: rappresenta l'email dell'utente che ha accettato la collaborazione, cioè colui che fornisce l'aiuto. Essa è una chiave esterna collegata alla tabella Utente.
- *Nome*: attributo che rappresenta il nome della Collaborazione. Esso è utilizzato solo per mostrare la lista delle collaborazioni, all'interno della sezione Utente, e renderle facilmente identificabili.
- *Descrizione*: descrizione della Collaborazione, cioè una sorta d'invito ad accettare che l'Utente può fare verso quello che possiede le abilità ricercate. Quest'attributo può essere NULL.
- *DataStipula*: vale NULL se la collaborazione non è ancora stata stipulata (non confermata), invece assume come valore una data, nel caso in cui sia confermata dall'Utente (data del momento dell'accettazione).

- *DataTermine*:
  - *Se DataStipula* e *DataTermine valgono entrambe NULL, allora la Collaborazione è in corso/ancora da confermare o rifiutare.*
  - *Se DataStipula* diversa da NULL e *DataTermine uguale a NULL, allora* la Collaborazione è stata accettata dall'Utente ricevente, ma deve ancora terminare. Sarà solo l'Utente richiedente a poterla terminare ed eventualmente rilasciare il feedback.
  - *Se DataStipula* uguale a NULL e *DataTermine diversa da NULL, allora* la Collaborazione è stata rifiutata dall'Utente ricevente, ma deve ancora essere notificata come tale al richiedente. Cioè in questa situazione, la riga della tabella resterà in Collaborazione solo fino a quando l'Utente richiedente deciderà di visualizzare le notifiche di accettazione sul suo profilo.

- *PunteggioFeedback*: valore intero da 1 a 5 che rappresenta il voto. Esso sarà utilizzato per calcolare il punteggio di Feedback dell'Utente ricevente. Nel caso in cui il valore assuma valore NULL è perché il Feedback non è ancora stato rilasciato. Questo attributo potrà essere solo NULL finche la DataTermine sarà uguale a NULL.

- *CommentoFeedback*: eventuale commento dell'Utente richiedente per spiegare meglio l'esperienza avuta durante la Collaborazione con l'Utente ricevente. Tale attributo assume valore NULL nel caso in cui il Feedback non sia stato rilasciato, oppure "Non ci sono commenti rilasciati" nel caso in cui il commento non sia stato fornito dall'Utente nel momento del rilascio. Questo attributo potrà essere solo NULL finche la DataTermine e PunteggioFeedback saranno uguali a NULL.
- *NotificaAlRichiedente*: attributo necessario per gestire le accettazioni delle richieste di collaborazione. Assume valore '1' per indicare che l'utente richiedente ha già visto il messaggio in cui è specificato che l'utente ricevente ha accettato la richiesta di collaborazione, '0' altrimenti.

## 3.3. Progettazione logica: Logic Model

La progettazione logica è una possibile traduzione dello schema ER. Essa mostra sempre le entità con gli attributi e le chiavi, ma si concentra soprattutto sul descrivere i collegamenti concettuali con altre tabelle.
Nello schema seguente sono indicate in giallo le "primary key" (chiavi primarie) e in rosso le "foreign key" (chiavi esterne).

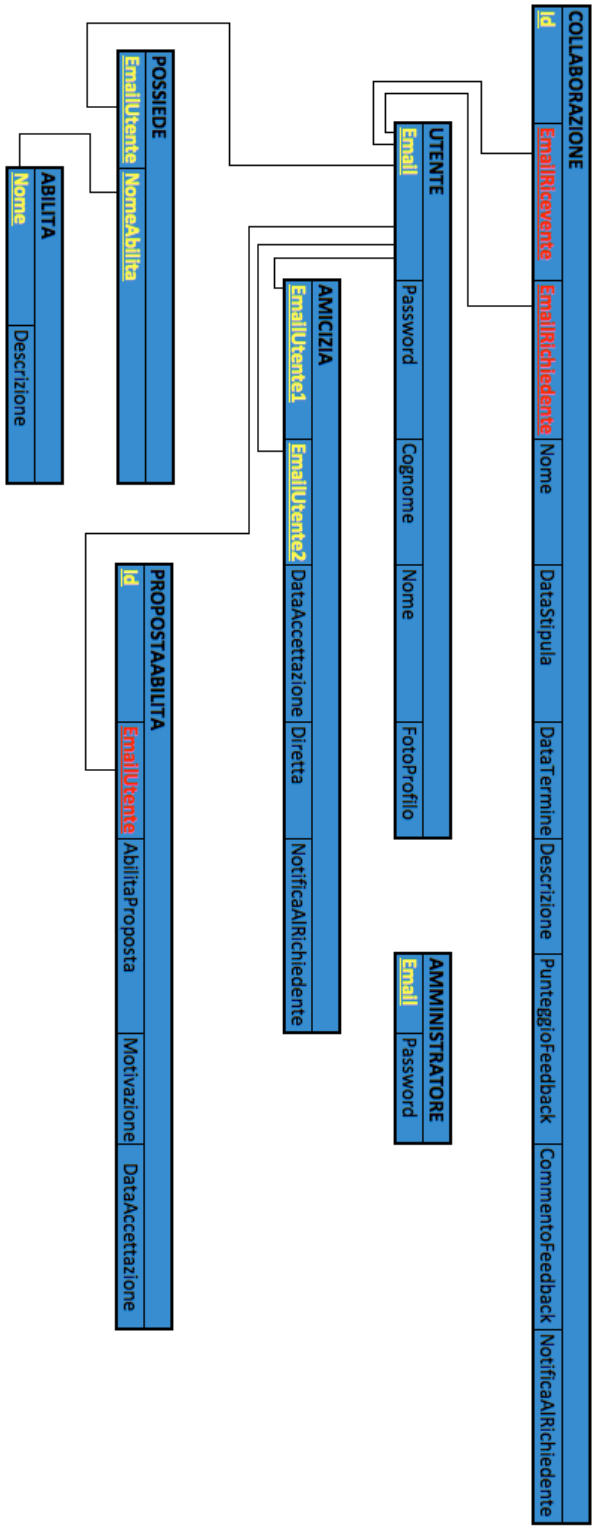


Fig. 3.3 – Schema Logico

## 3.4. Modello di navigazione: UX Model

Un modello di navigazione, detto User eXperience Model ("UX Model"), rappresenta la navigazione tra le pagine di SWIMv2. Ci sono <<screen>>, <<input form>>, aggregazioni, composizioni e metodi.
Gli elementi come <<screen>>, <<input form>> hanno nome, attributi e metodi.

Il simbolo "$" indica che la <<screen>> è raggiungibile in qualunque momento durante la navigazione, mentre il simbolo "+" rappresenta una lista di elementi con gli attributi specificati nella classe oppure all'esterno tramite la relazione di aggregazione (rombo vuoto).

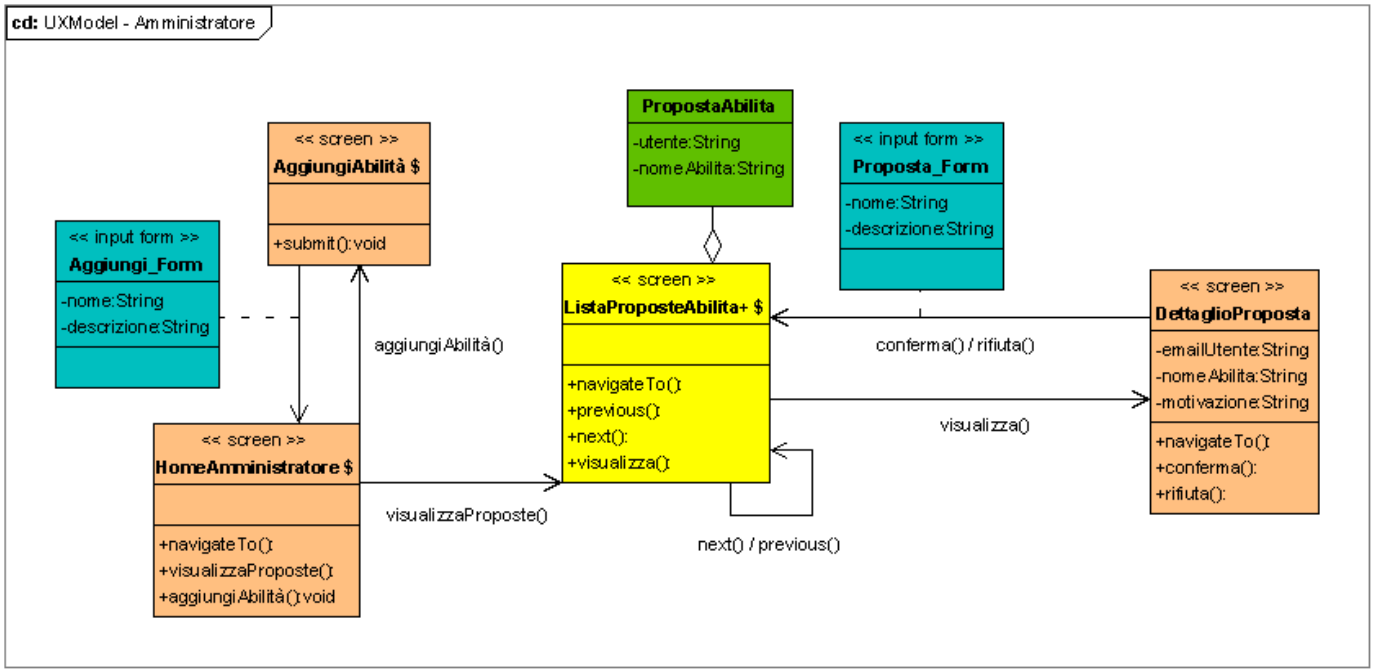


Fig. 3.4 - UXModel Amministratore

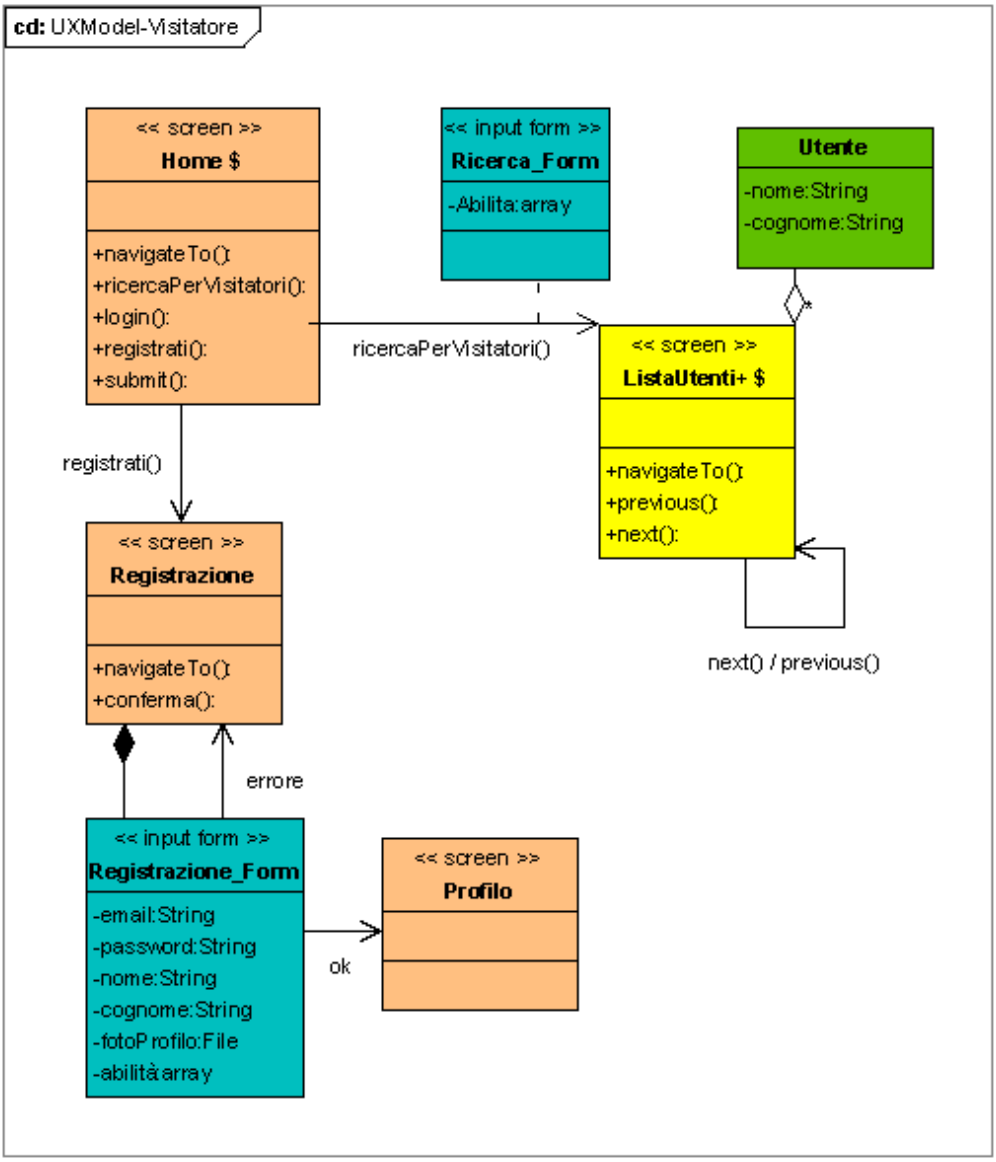


Fig. 3.5 - UXModel Visitatore

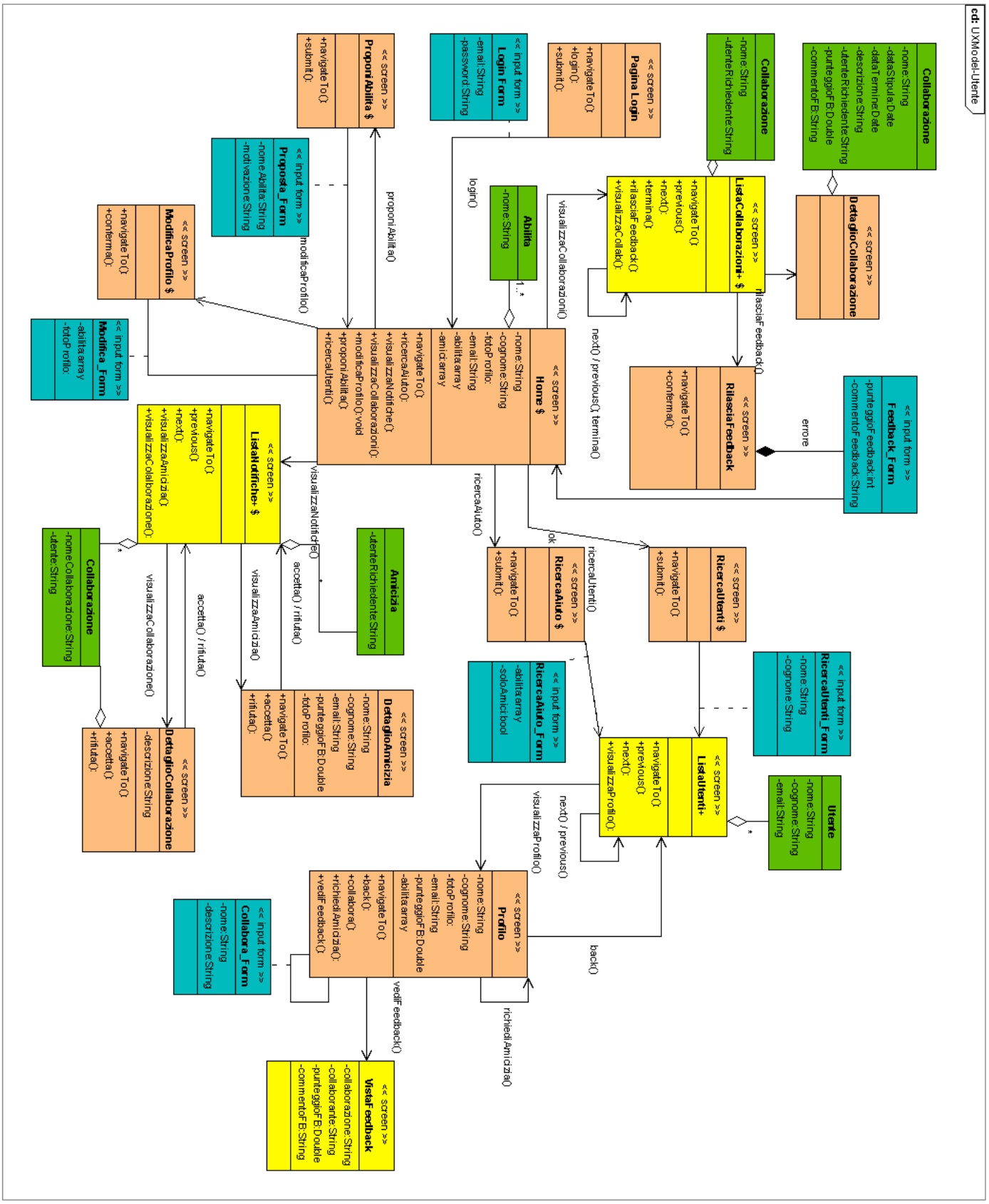


Fig. 3.6 - UXModel Utente

## 3.5. Altri diagrammi

### *3.5.1.* Diagramma di analisi (BCE)

Il diagramma Boundary-Control-Entity divide il sistema in presentazione (boundary), controllo (control) e dati (entity).
Questi diagrammi usano il seguente standard:

- Classi: rappresentate come rettangoli con il tipo (<<control>>, <<entity>> o <<boundary>>), il nome, gli attributi e i metodi.
    - <<boundary>>: classe che fa parte dell'interfaccia utente
    - <<entity>>: classe che rappresenta la persistenza dei dati
    - <<control>>: classe che rappresenta la logica del sistema
- Relazioni: associazioni tra classi specificando le cardinalità
- Frecce: indicano

A seguire lo schema completo.

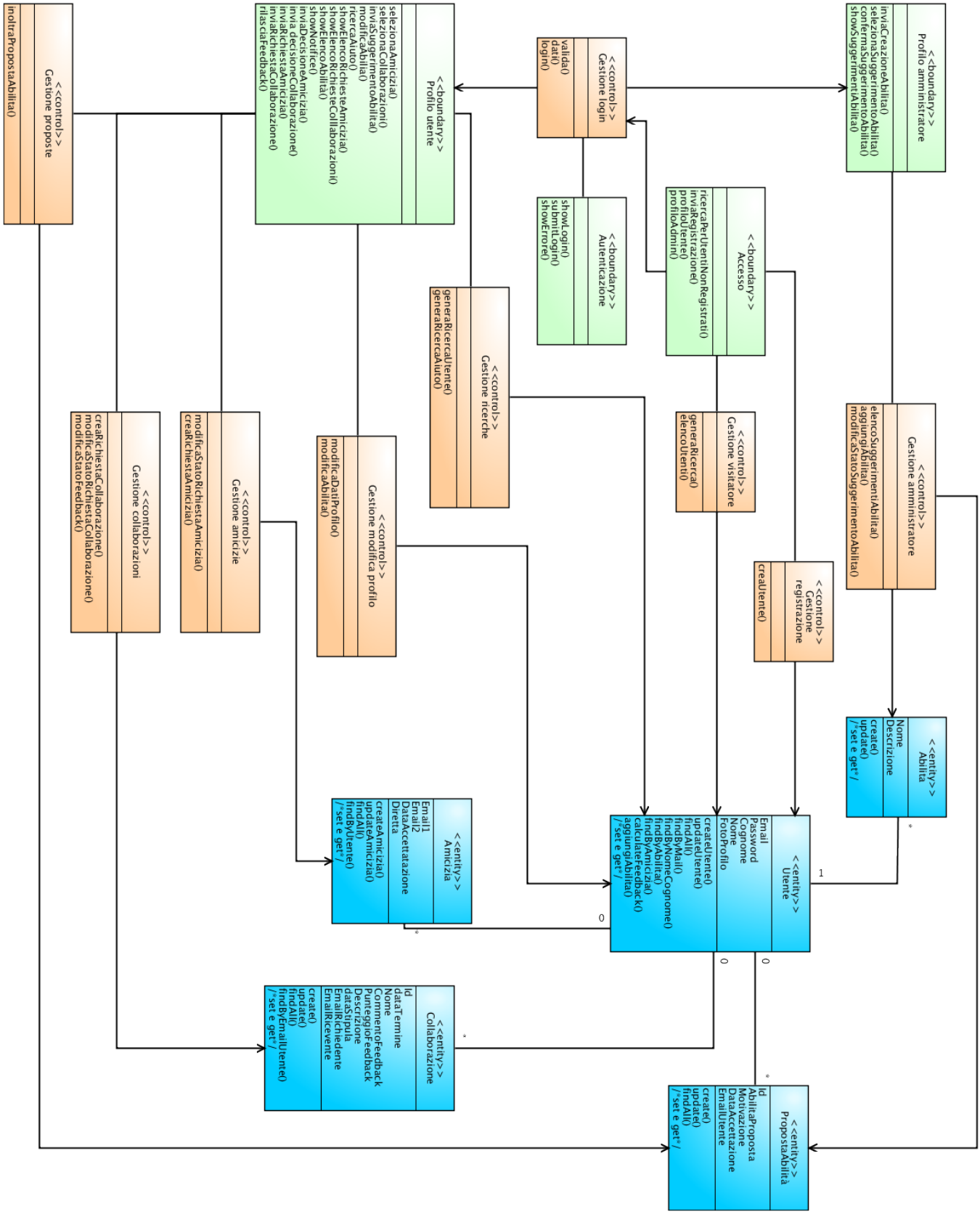


Fig. 3.7 - BCE Model

## 4. Progetto per JEE

Dopo aver progettato il sistema ad alto livello, è necessario scendere in quello più basso in cui sono specificati gli elementi che saranno implementati direttamente tramite JEE.
Prima sono definiti gli "Entity Beans" per rappresentare i dati e in seguito i "Session Bean" che rappresentano i controlli.

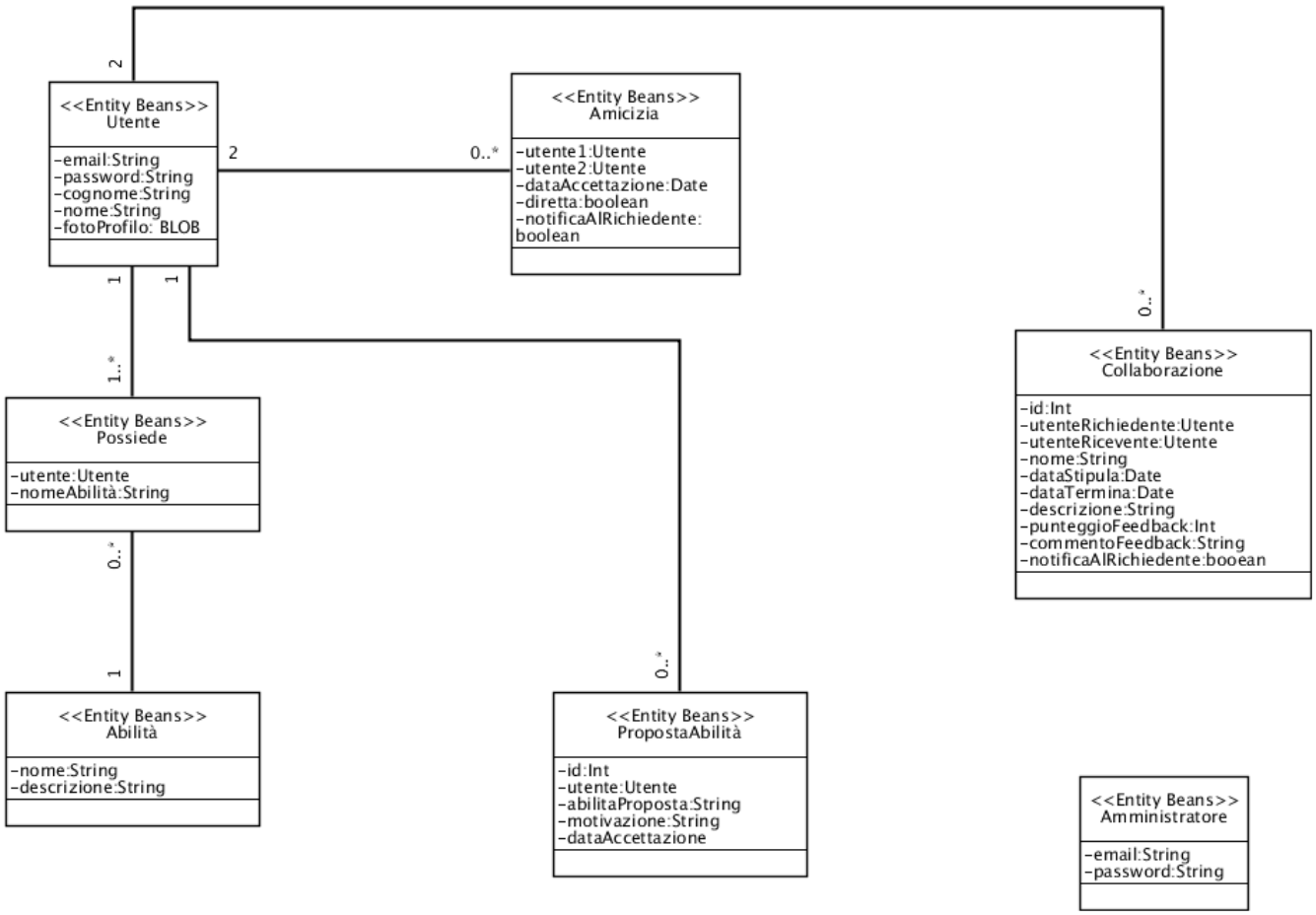


Fig. 4.1 - EntityBeans

Di solito, i "Session Bean" coincidono con i <<control>> del diagramma BCE, compresi i metodi.
Il control "Gestione visitatore" non sarà rappresentato come Session Bean, ma le sue funzionalità saranno inglobale nel Sessio Bean chiamato "GestioneRicerche".
Un Session Bean può essere di tipo Stateless o Stateful. Sotto sono specificati i nomi e i tipi.

- GestioneRegistrazione (stateless)
- GestioneLogin (stateless)
- GestioneAmministratore (stateless)
- GestioneModificaProfilo (stateless)
- GestioneRicerche (stateless)
- GestioneProposte (stateless)
- GestioneAmicizie (stateless)
- GestioneCollaborazioni (stateless)

*4.1.1.* Diagrammi di sequenza

Per descrivere meglio la sequenza di operazioni che avvengono nel sistema sono stati allegati alcuni diagrammi di sequenza.

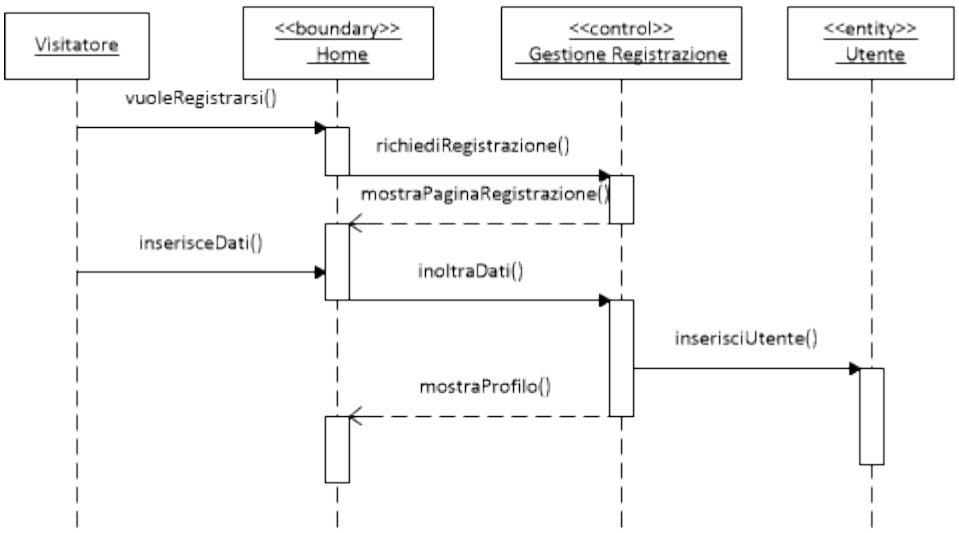


Fig. 4.2 - Sequence Diagram registrazione

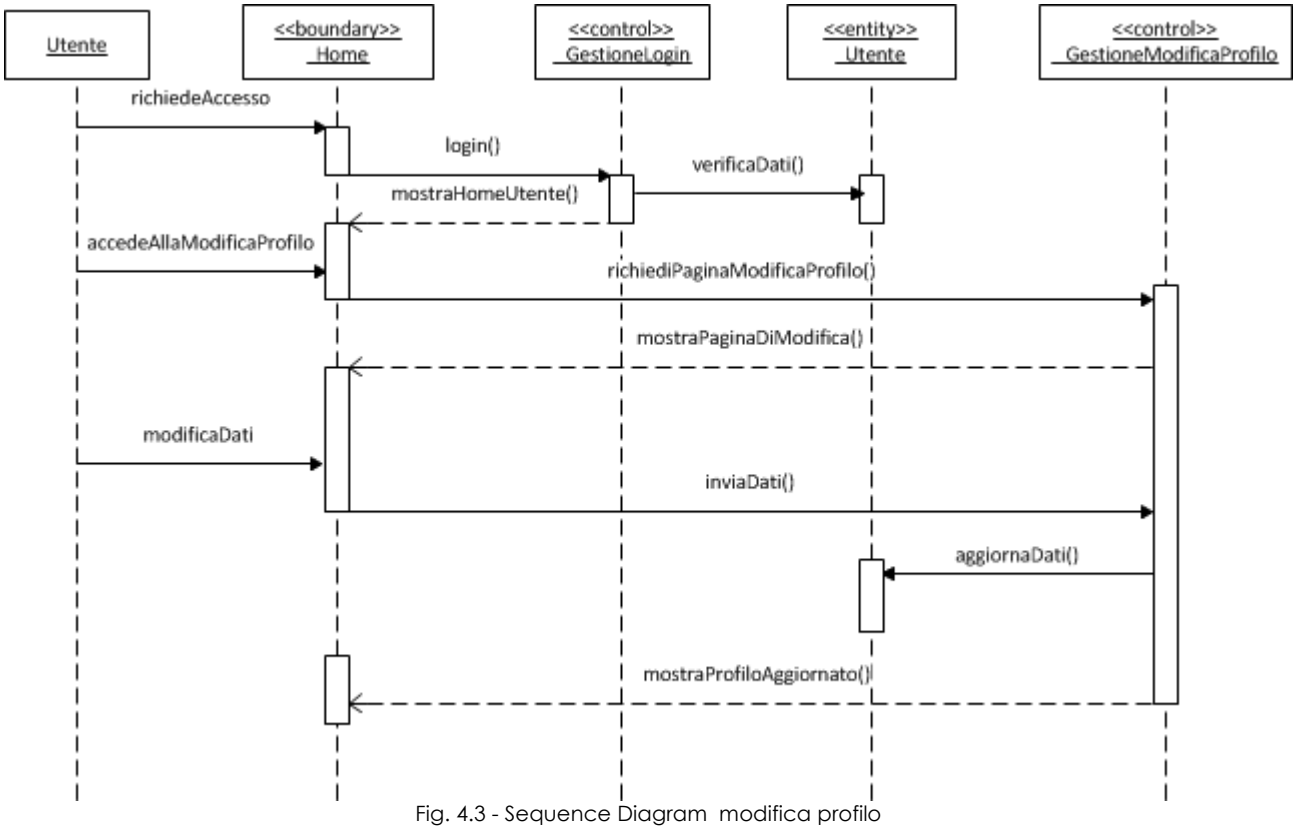


Fig. 4.3 - Sequence Diagram  modifica profilo

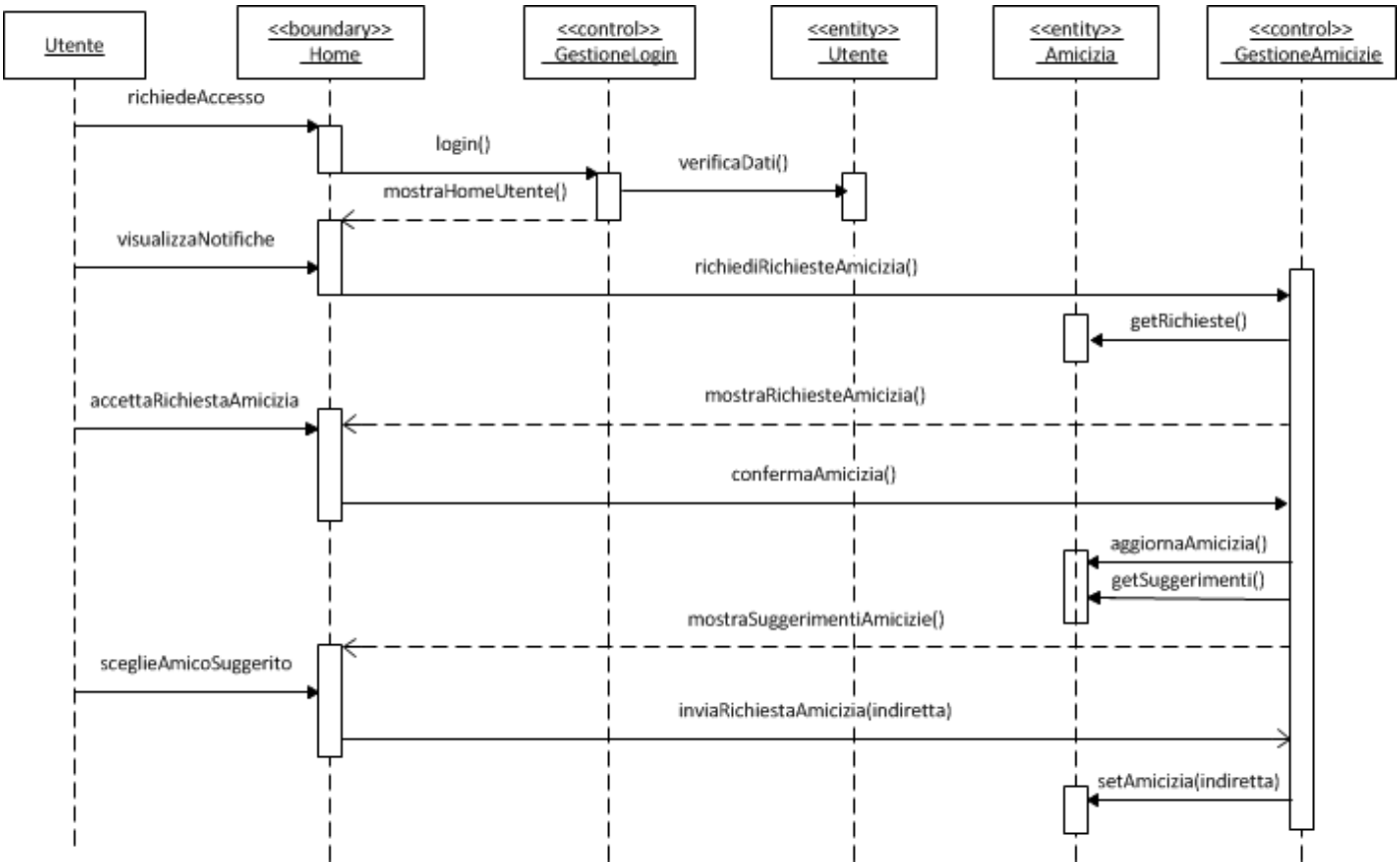


Fig. 4.4 - Sequence Diagram accetta richiesta amicizia

# 5. Informazioni aggiuntive

## 5.1. Possibili estensioni future

- Quando le abilità aumenteranno, si potrà pensare ad un sistema diverso per sceglierle, per esempio un modulo di ricerca che mostri i risultati in tempo reale mentre l'utente ne digita il nome.
- Le ricerche potrebbero essere realizzate in modo di mostrare risultati anche inserendo nomi e/o cognomi parziali.
- Ai risultati delle ricerche si potrebbero aggiungere le miniature delle foto dei profili degli utenti.
- Possibilità di mettere il titolo di studio nel profilo (opzionale in modo che un utente di tredici anni possa non inserire nulla, nel caso non volesse far conoscere la sua età)
- Possibilità di rimuovere gli amici (funzionalità prevista dal team di sviluppo che sarà inserita nella versione successiva del prodotto).
- Possibilità di vedere la lista delle notifiche al richiedente, anche dopo averle visualizzate una volta e quindi far si che debbano essere "lette" perché il sistema le indichi come viste.
- Integrazione con altri social network.
- Se il sistema crescesse, sarebbe necessario qualche controllo di sicurezza, per esempio la possibilità di applicare un "ban" agli utenti (espellerli dal sistema ed impedirne nuove registrazioni).
- Con l'eventuale introduzione di sistemi di sicurezza, potrebbe rivelarsi necessario prevedere un sistema di tutela della privacy per far sì che l'utente possa scegliere quali dati mostrare nel proprio profilo.
- Nonostante il sistema sia accessibile anche da Smartphone e Tablet, in futuro potrebbe essere necessario ridisegnare l'interfaccia grafica aggiungendo una versione destinata ai soli dispositivi mobili.
- SWIMv2 potrà essere esteso aggiungendo Città, Provincia e CAP nella tabella Utente del database. Gli utenti già registrati potrebbero essere obbligati ad aggiungerli nel profilo per continuare ad utilizzare il software, invece coloro che non sono ancora utenti di SWIMv2 sarebbero costretti ad inserire questi dati per poter ottenere l'accesso al sito web.
- Per migliorare la sicurezza sull'identità degli utenti, si potrà inserire il Codice Fiscale, come requisito obbligatorio (non visibile agli utenti) e utilizzare un servizio esterno per verificarne la validità, comprese le omonimie o persone nate in paesi esteri. Per gli utenti che sono già registrati si potrebbe imporre l'inserimento del Codice Fiscale per continuare ad usare il servizio ed accedere nuovamente al profilo.
- Introdurre un sistema che notifichi le registrazioni tramite e-mail, inviando la password scelta in fase di registrazione o una password temporanea da reimpostare al primo login.
- Possibilità di rilasciare/ricevere un feedback sia come ricevente sia come richiedente ed eventualmente la possibilità di rispondere ad un commento di feedback fatto da un altro utente.

## 5.2. Spiegazioni di alcune scelte di progettazione

**Perché l'utente non può specificare il sesso?**
Il team di sviluppo ha ritenuto che la conoscenza del sesso di un utente non fosse importante, anzi, in alcuni casi potrebbe rilevarsi addirittura sfavorevole.
Per esempio, una donna che ha l'abilità "Meccanico" potrebbe non essere considerata seriamente dagli utenti, poiché per convenzioni sociali, in genere, un meccanico è di sesso maschile.

**Perché l'utente non può specificare la sua età/data nascita?**
Per un motivo simile a quello sopra, cioè potrebbe essere uno svantaggio per un ipotetico utente di sedici anni che programma app per iPhone. Infatti, è possibile che per la sua giovane età venga ritenuto poco credibile o affidabile.

**Perché l'utente non può specificare la sua città?**
In questa versione il sistema è limitato ad una sola città e di conseguenza è stato dato per scontato che gli utenti cercassero/offrissero aiuto nella sola città in cui è in funziona il software. Ovviamente, anche altre presone possono registrarsi, poiché non vi sono sistemi di sicurezza, ma per offrire aiuto sarà lui a dover cambiare città e per richiederlo potrebbe non trovare nessuno disposto a spostarti dalla proprio città. E' anche possibile che l'aiuto in questione non preveda cambiamenti di sede o l'incontro fisico tra gli utenti e quindi in tal caso non vi sarebbero particolari problemi.

## Indice delle figure