Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Informatica

Assignment 3
Processo e Sviluppo del Software

Andrea Carubelli 803192
Gianmaria Balducci 807141

Link repository:
https://gitlab.com/jiimmy.exe/2019_assignment3_gym_sharing/

Anno Accademico 2019-2020

Overview dell'applicazione

Lo scopo dell'applicazione è quello di permettere agli utenti di cercare delle palestre disponibili, consultando gli abbonamenti erogati, con i relativi prezzi e consultando i personal trainer che lavorano all'interno della palestra.
Inoltre è possibile che un utente scelga un personal trainer da cui può essere seguito durante i suoi allenamenti.
Viceversa, le palestre possono gestire i propri abbonamenti, i personal trainer iscritti, e scegliere se una palestra può essere una sua affiliata.
Inoltre il personal trainer può scegliere la palestra in cui vuole lavorare.

Diagramma E-R

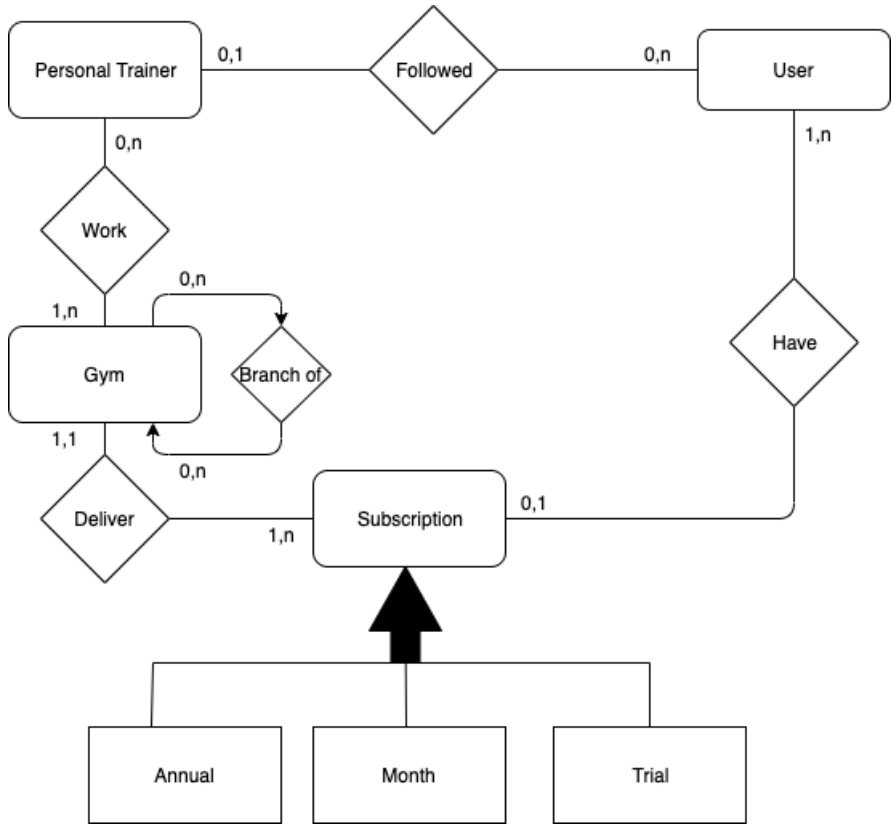

Figura 1: Diagramma E-R

Dal diagramma E-R possiamo osservarne le relazioni:

- Un utente può possedere un solo abbonamento, ma un abbonamento può essere acquistato da più utenti
- Una palestra può erogare più abbonamenti ma un abbonamento può essere erogato da una sola palestra

- Una palestra può avere delle filiali attraverso una relazione "self-loop"
- In una palestra ci possono essere più personal trainer e un personal trainer può lavorare in più palestre
- Un utente può essere seguito da uno o nessun personal trainer e un personal trainer può seguire più utenti

Diagramma UML

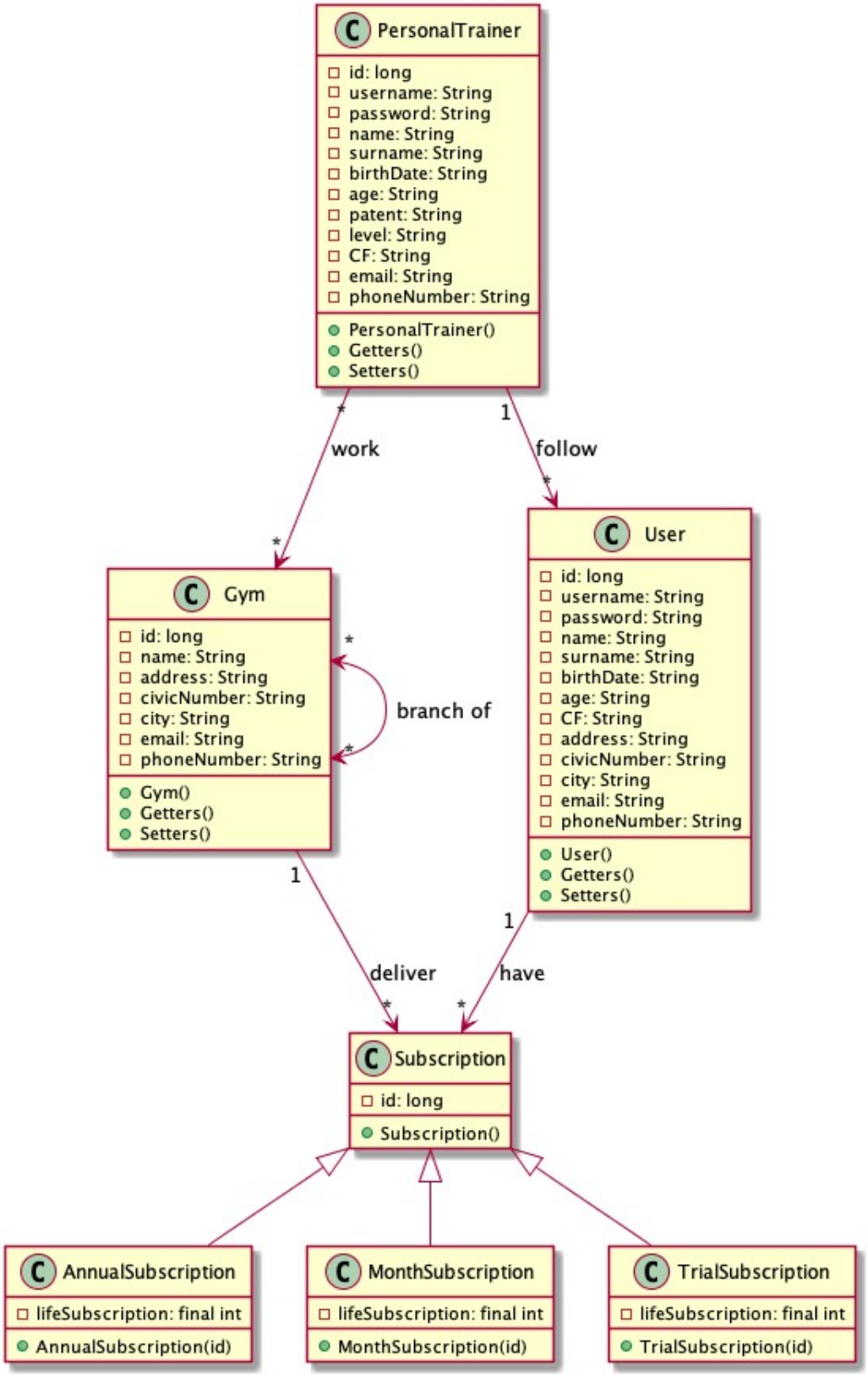

Figura 2: Diagramma UML

Descrizione classi

Abbiamo scelto l'implementazione "front-end only" per un' applicazione Spring-MVC.
Il nostro package base è com.example.assignment3 in cui sono inclusi i seguenti package:

- Entities
- Controller
- Repository

Il package **Entities** contiene le seguenti entità:

- **Gym** che rappresenta le palestre che sono collegate attraverso una relazione **@ManyToMany** con l'entità affiliateGyms (di tipo Gym) e con l'entità PersonalTrainer. Invece, collegata con l'entità Suscription attraverso una relazione **@OneToMany**.
- **User** che rappresenta gli utenti è collegata attraverso una relazione **@OneToOne** con l'entità Subscription e Personal Trainer.
- **Personal Trainer** che rappresenta i personal trainer è collegata all'entità User attraverso una relazione **@OneToMany** e con l'entità Gym attraverso una relazione **@ManyToMany**.
- **Subscription** che rappresenta gli abbonamenti erogati dalle palestre. **AnnualSubscription, MonthSubscription, TrialSubscription** ereditano questa classe.
  L'entità Subscription è collegata all'entità User attraverso una relazione **@OneToMany** e con l'entità Gym attraverso una relazione **@OneToOne**.

Tutte le relazione @ManyToMany e @OneToMany sono state implementate attraverso l'utilizzo di liste, mentre le relazioni @OneToOne sono state implemenetate attraverso l'istanziazione del singolo oggetto.

Il package **Controller** contiene i controller, uno per ogni classe sopracitata che permetteranno alla webapp di chiamare le operazioni CRUD da effettuare, muoversi tra le views ed effettuare le search.
I controller presenti sono i seguenti:

- **BasicController** che gestisce le *route* delle views, e le funzioni di registrazione/-login per le tre entità: user, gym e personal trainer
- **GymController** che gestisce le operazioni CRUD e search per l' entità Gym
- **UserController** che gestisce le operazioni CRUD per l' entità User

- **PersonalController** che gestisce le operazioni CRUD e search per l' entità Personal Trainer
- **SubscriptionController** che gestisce le operazioni CRUD per l' entità Subscription

Il package **Repository** contiene le interfacce che ci permetteranno di salvare gli oggetti in modo volatile.
Le interfacce create sono:

- **GymRepository**
- **UserRepository**
- **PersonalTrainerRepository**
- **SubScriptionRepository**

Queste interfacce estendono la classe CrudRepository che permette di utilizzare dei metodi CRUD, i quali vengono utilizzati dai controller per gestire le relazioni tra entità e le stesse operazioni CRUD.
Inoltre, queste interfacce sono state estese aggiungendo dei metodi find, utilizzati durante le operazioni di search.

Le **resources** sono invece tutti i file html che ci permetteranno di interagire con la web-app.

Nel **pom.xml** abbiamo tutte le dipendenze che ci serviranno per avviare la web-app.

Nel file **Assignment3Application.java** invece abbiamo le inizializzazioni degli oggetti e delle relazioni al primo avvio dell'app, così che si possa già avere una base sui cui poter vedere tutte le funzionalità implementate.

Implementazione operazioni CRUD

In questa applicazione le operazioni **CRUD** (create, read, update, delete) e l'operazione search sono state implementate per ogni entità.

Create

L'operazione *Create* viene effetuata tramite la registrazione dei diversi tipi di utenza: user, gym e personal trainer. Vengono creati quindi dei nuovi ogetti corrispondenti alle classi entità. Per l'entità *SubScription* questa operazione viene gestita dall' account *gym* che può aggiungere diversi tipi di abbonamento.

Read

Questa operazione viene richiesta quando, da un' entità (ad esempio personal trainer) si vuole consultare l'elenco di oggetti di un'altra entità (ad esempio gym) per effetturare poi delle operazioni (iscriversi) o solamente a scopo consultativo. Oppure quando si vuole vedere le informazioni riguardo il propro account, cliccando su *My profile*.

Update

Per ogni entità è possibile modificare tutti i suoi attributi, tramite il link *Edit account* nelle informazioni del profilo. L'aggiornamento è quindi propagato in tutte le entità collegate da una relazione all'entità modificata.

Delete

La delete è stata implementata in modo tale da dare la possibilità ad ogni entità di eliminarsi dalla piattaforma tramite il link *Remove account*, conseguentemente un' entità rimossa sarà rimossa anche dalle entità ad essa collegata. Nel caso degli abbonamenti, questi possono essere eliminati solo da un'account gym che li possiede.

Search

L'operazione di search in questa piattaforma è implementata tramite delle search box in modo full-text. Offre la possibilità di cercare un' entità tramite tutti i suoi campi, ma la ricerca è gestita in maniera diversa da campo a campo: se cerco un personal trainer per nome mi basterà che il nome di tale oggetto contenga la stringa in input, invece se la ricerca è effettuata in base al livello di esperienza la stringa in input dovrà essere uguale al valore dell'attributo.