# *Modern Concurrent Data Oriented Applcations in C#*
# *Exam project*
# *Student: Alessandro Messina (matr: O55000354)*

**Descrizione progetto**

Il progetto consiste nella realizzazione di un'applicazione Web per la gestione degli utenti di una palestra (per pesistica). In particolare si è voluta realizzare un'applicazione che consenta agli amministratori di una palestra di gestire gli utenti, gli istruttori, i macchinari e i corsi offerti dalla palestra.
Per avviare il progetto, occorre aprire sia ServiceAPI che SuperCoolApp in Visual Studio tramite i file .sln. Dunque occorre eseguirli entrambi e attendere che si apra sul proprio browser la SuperCoolApp.

**Progettazione database**

*Descrizione*

Come database si è scelto di utilizzare MySQL. La progettazione del database è stata effettuata considerando le seguenti specifiche relative alla palestra:

- la palestra offre vari *corsi* mirati alla pesistica: bodybuilding, powerlifting, crossfit e pesistica olimpica (weightlifting). Per ogni corso si vogliono memorizzare il nome, gli istruttori che vi insegnano, gli utenti iscritti, la descrizione e gli orari.
- i corsi sono forniti in un grande spazio chiuso (una sala di allenamento) in cui sono presenti diversi macchinari e attrezzi. Per ogni macchinario e attrezzo si vogliono memorizzare il nome e lo scopo per cui può essere utilizzato.
- per ogni *utente* si vogliono memorizzare il nome, il cognome, l'indirizzo di casa, il codice fiscale e i corsi seguiti;
- la palestra offre ogni giorno degli *istruttori* il cui compito è quello di seguire gli utenti. Per ogni *istruttore* si vogliono memorizzare il nome, il cognome, l'indirizzo di casa, il codice fiscale i corsi in cui insegna e il turno lavorativo;
- a ogni utente è assegnata una *scheda di allenamento* in cui sono indicati gli esercizi da svolgere in palestra. Per ogni scheda si vuole memorizzare la sua descrizione. Una scheda può essere assegnata a più persone, ma ogni persona può avere una sola scheda.

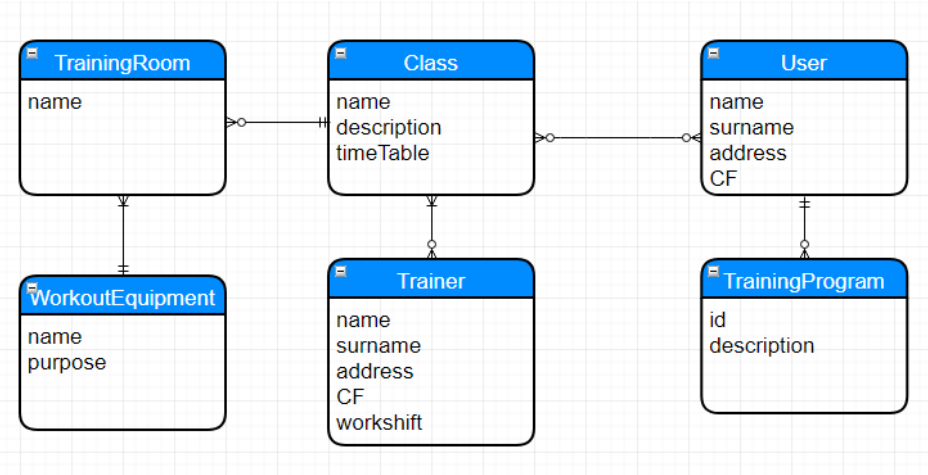


*Figura 1. E-R model*

*Relazioni del database*

TRAININGROOM(name)
CLASS(name, description, timeTable, trainingRoomName), FK: trainingRoomName → TRAININGROOM(name)
TRAININGPROGRAM(id, description)
USER(name, surname, address, FC, trainingProgramId), FK: trainingProgramId → TRAININGPROGRAM(id)
MEMBERSHIP(className, userFC), FK: className → CLASS(name), userFC → USER(FC)
WORKOUTEQUIPMENT(name, purpose, trainingRoomName), FK: trainingRoomName → TRAININGROOM(name)
TRAINER(name, surname, address, FC, workshift)
TRAINING(className, trainerFC), FK: className → CLASS(name), trainerFC → TRAINER(FC)

**serviceAPI**
La serviceAPI fornisce il servizio REST all'applicazione WEB. Per poter utilizzare l'applicazione web occorre solamente eseguire la serviceAPI.

*Entity framework, code first per la creazione del database*
Il database viene creato nell'applicazione *serviceAPI* tramite ORM con approccio *code first*. In particolare viene usato *Entity Framework*. L'utente può creare il database (popolato o meno) dalla home page della web application. Sempre da qui può anche cancellare il database. Questa scelta è stata fatta per permettere un più rapido utilizzo dello stesso. Il nome del database è "O55000354_gymDatabase".

*serviceAPIcontroller*
Nella classe *serviceAPIcontroller* sono stati aggiunti i metodi per eseguire le azioni CRUD sulle classi del database. In particolare in tutte le classi sono stati implementati i metodi per eseguire insert, update, read e delete.

**WebApp**
L'applicazione web consiste di una serie di schermate in cui l'utente può navigare per effettuare le operazioni CRUD sulle varie tabelle del database (che vengono mostrate come tabelle HTML nell'app). Per poter usare la Web App è sufficiente avviarla (nota che anche la serviceAPI deve essere avviata).
- Nella home l'utente può creare/cancellare il database, mentre nelle altre schermate può operare sulle tabelle.
- Nelle tabelle è stata inserita la possibilità di filtrare i risultati e di ordinarli tramite le Pipe Angular.
- Per poter inserire, cancellare o aggiornare un record di una tabella è necessario cliccare sugli appositi pulsanti. Per salvare le modifiche occorre infine cliccare sul pulsante "SAVE".

**Note**
Nota importante. Per questioni di tempo non mi è stato possibile implementare: accesso al database con password (si accede infatti tramite utente root), i meccanismi di verifica degli input dell'utente, la registrazione e il login dell'utente, le informazioni relative alle relazioni N a N (ad esempio i corsi cui partecipa un dato utente) e la gestione delle eliminazioni di eventuali record di tabelle da cui dipendono altri record di altre tabelle (chiavi esterne). In quest'ultimo caso, se si effettuano eliminazioni di tal genere nessuna modifica sarà apportata al database del sistema.

Nota sui Warnings. Nell'applicazione web sono presenti dei warning dovuti al css. Questi potevano anch'essi essere sistemati, ma per questioni di tempo non mi è stato possibile farlo.

Nota sui commenti. Per non riempire l'applicazione di troppi commenti, dei vari componenti aggiunti nell'applicazione web ho commentato solo "classes".