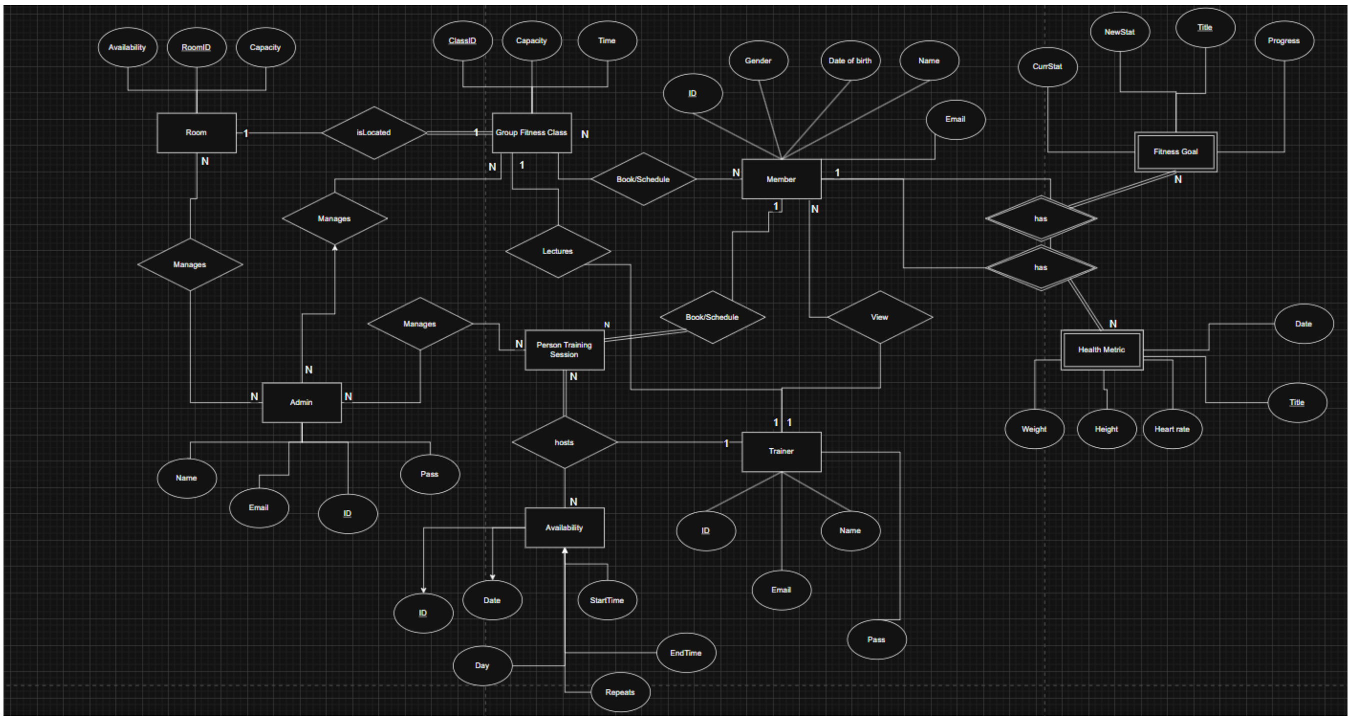
# COMP3005 PROJECT

GROUP 161

RAYAN CHOUDHARY: 101306022   NGOYI KABASO: 101320375

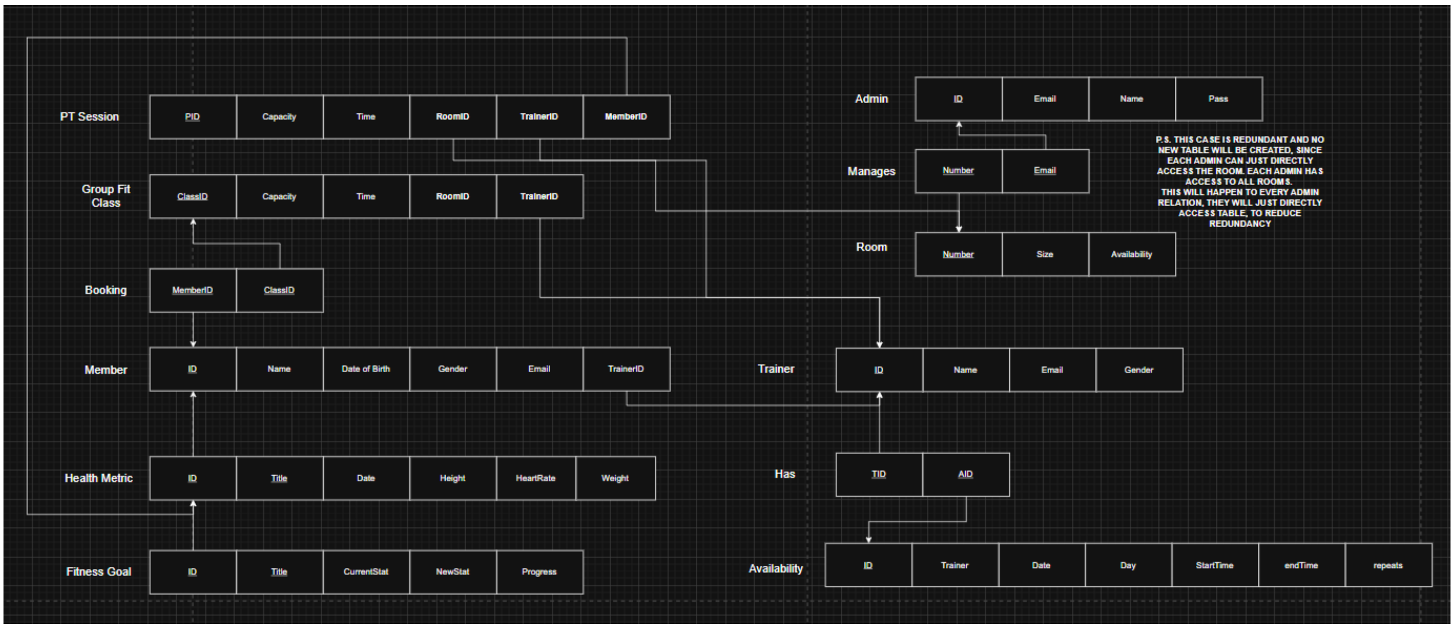
## Contents

## ER-Model



Link to draw.io:
https://app.diagrams.net/?src=about#G1CeKyXHVi6iylYEpZKr_YUiQ3Pe0opYP_#%7B"pageId"%3A"RVxCMzVvr24md0h16g40"%7D

## Relational tables

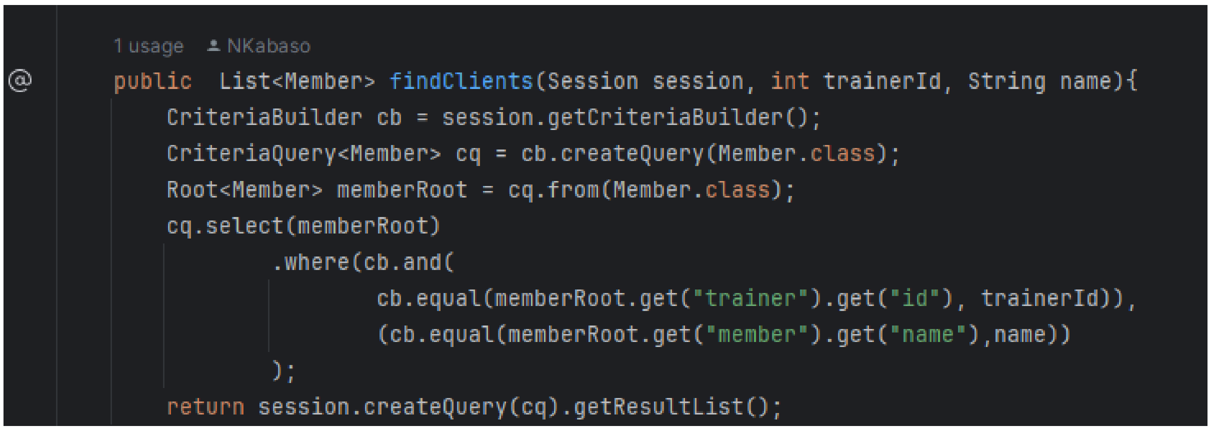


P.S. THIS CASE IS REDUNDANT AND NO NEW TABLE WILL BE CREATED, SINCE EACH ADMIN CAN JUST DIRECTLY ACCESS THE ROOM. EACH ADMIN HAS ACCESS TO ALL ROOMS.
THIS WILL HAPPEN TO EVERY ADMIN RELATION, THEY WILL JUST DIRECTLY ACCESS TABLE, TO REDUCE REDUNDANCY

## ORM Integration

Classes that related to database entities were turned into entity classes.  Thus, any queries that involved these classes will be handled by Hibernate.

A big use was in retrieving data from the database. For example, if we wanted to data from the *members* table we could use Hibernate's *CriteriaBuilder* class to build a query to get the data.



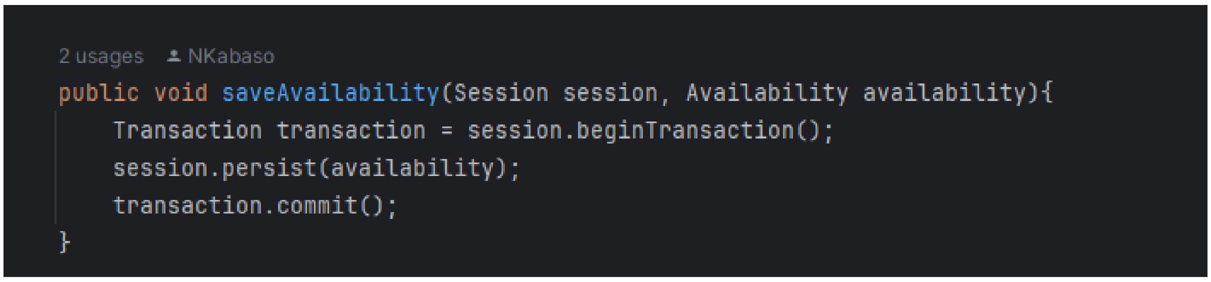
```java
1 usage   NKabaso
public  List<Member> findClients(Session session, int trainerId, String name){
    CriteriaBuilder cb = session.getCriteriaBuilder();
    CriteriaQuery<Member> cq = cb.createQuery(Member.class);
    Root<Member> memberRoot = cq.from(Member.class);
    cq.select(memberRoot)
            .where(cb.and(
                    cb.equal(memberRoot.get("trainer").get("id"), trainerId)),
                    (cb.equal(memberRoot.get("member").get("name"),name))
            );
    return session.createQuery(cq).getResultList();
```

*Figure 1: findClients return a list of Member objects associated with a specific trainer*

The SQL version of this query would be:

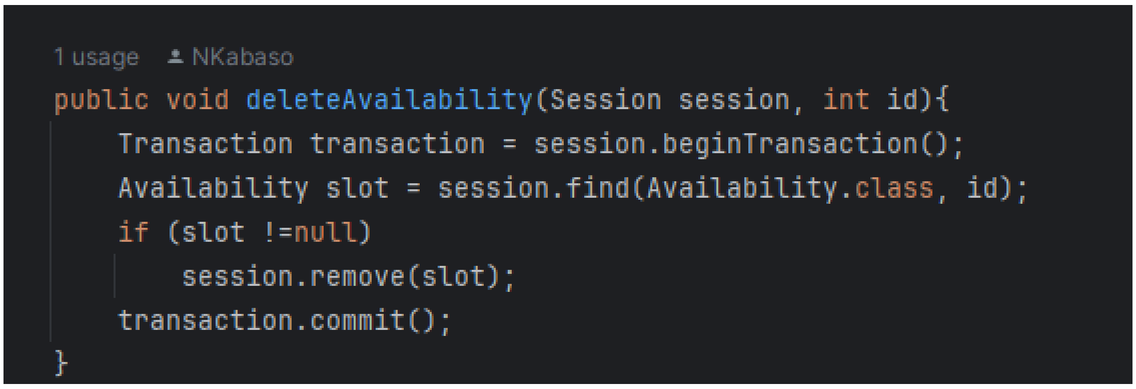
SELECT * FROM Member WHERE trainer.id = *trainer_id* AND Member.name = *name*

Hibernate also takes cares of DDL queries like *INSERT* and *DELETE*. An example would be adding or removing the time slot a trainer is available.



```java
2 usages   NKabaso
public void saveAvailability(Session session, Availability availability){
    Transaction transaction = session.beginTransaction();
    session.persist(availability);
    transaction.commit();
}
```

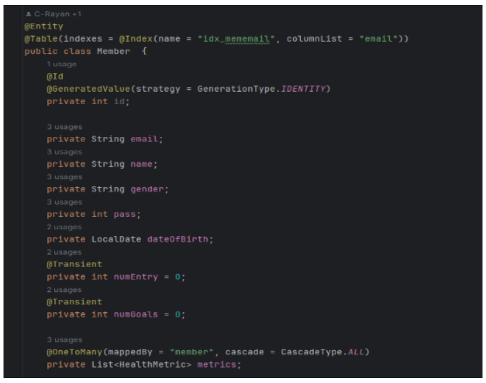
*Figure 2: saveAvailability inserts a new available time slot to the database*



```java
1 usage   NKabaso
public void deleteAvailability(Session session, int id){
    Transaction transaction = session.beginTransaction();
    Availability slot = session.find(Availability.class, id);
    if (slot !=null)
        session.remove(slot);
    transaction.commit();
}
```

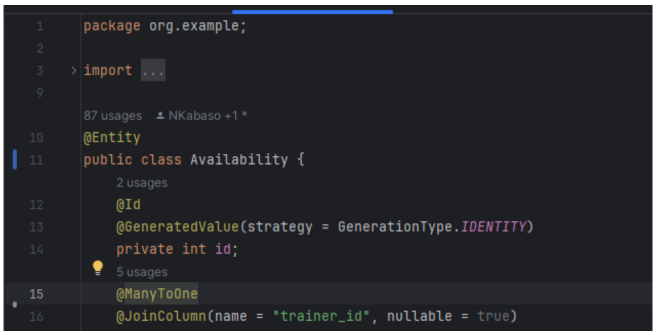
*Figure 3: deletAvailability finds the specified time slot and deletes it from the database*

## Major entities



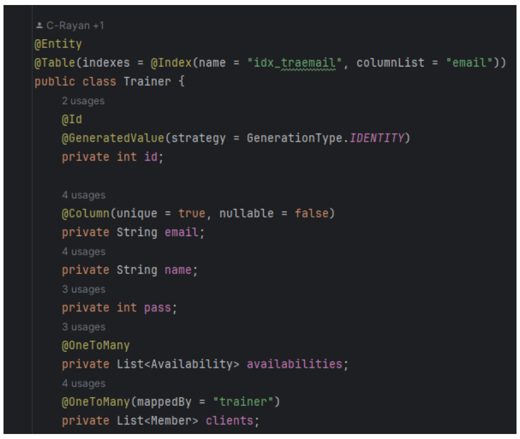
```java
C-Rayan +1
@Entity
@Table(indexes = @Index(name = "idx_mememail", columnList = "email"))
public class Member  {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    3 usages
    private String email;
    3 usages
    private String name;
    3 usages
    private String gender;
    3 usages
    private int pass;
    2 usages
    private LocalDate dateOfBirth;
    2 usages
    @Transient
    private int numEntry = 0;
    2 usages
    @Transient
    private int numGoals = 0;

    3 usages
    @OneToMany(mappedBy = "member", cascade = CascadeType.ALL)
    private List<HealthMetric> metrics;
```

*Figure 4: Member class*



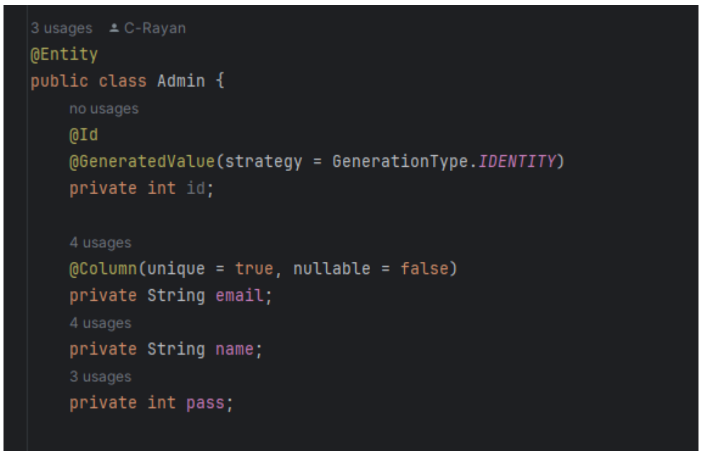
```java
package org.example;

import ...

87 usages   NKabaso +1 *
@Entity
public class Availability {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    5 usages
    @ManyToOne
    @JoinColumn(name = "trainer_id", nullable = true)
```

*Figure 5* Availability class



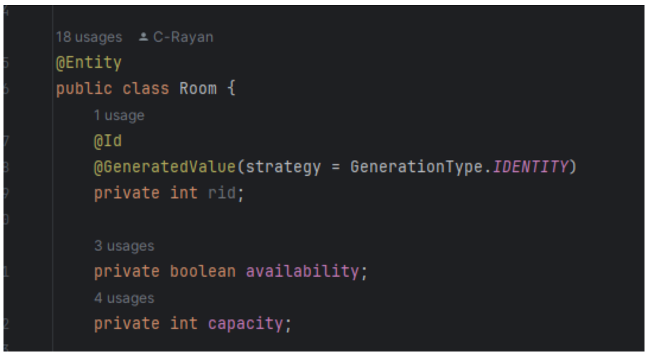
```java
C-Rayan +1
@Entity
@Table(indexes = @Index(name = "idx_traemail", columnList = "email"))
public class Trainer {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    4 usages
    @Column(unique = true, nullable = false)
    private String email;
    4 usages
    private String name;
    3 usages
    private int pass;
    3 usages
    @OneToMany
    private List<Availability> availabilities;
    4 usages
    @OneToMany(mappedBy = "trainer")
    private List<Member> clients;
```

*Figure 7: Trainer class*



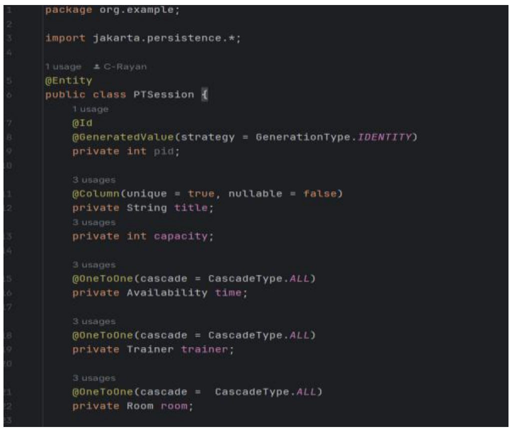
```java
3 usages   C-Rayan
@Entity
public class Admin {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    4 usages
    @Column(unique = true, nullable = false)
    private String email;
    4 usages
    private String name;
    3 usages
    private int pass;
```

*Figure 6:Admin class*



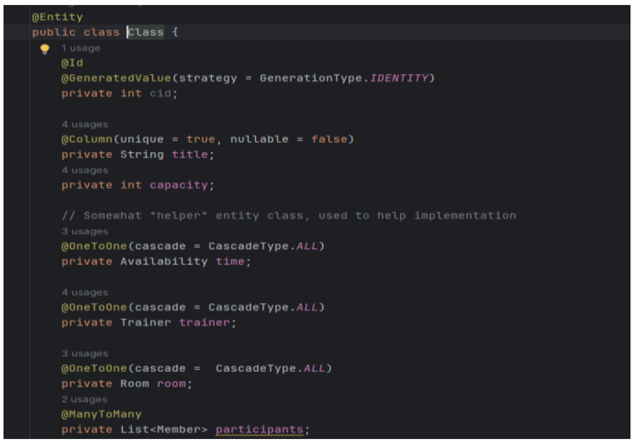
```java
18 usages   C-Rayan
@Entity
public class Room {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int rid;

    3 usages
    private boolean availability;
    4 usages
    private int capacity;
```

*Figure 9: Room class*



```java
package org.example;

import jakarta.persistence.*;

1 usage   C-Rayan
@Entity
public class PTSession {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int pid;

    3 usages
    @Column(unique = true, nullable = false)
    private String title;
    3 usages
    private int capacity;

    3 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Availability time;

    3 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Trainer trainer;

    3 usages
    @OneToOne(cascade =  CascadeType.ALL)
    private Room room;
```

*Figure 8:PTSession class*



```java
@Entity
public class Class {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int cid;

    4 usages
    @Column(unique = true, nullable = false)
    private String title;
    4 usages
    private int capacity;

    // Somewhat "helper" entity class, used to help implementation
    3 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Availability time;

    4 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Trainer trainer;

    3 usages
    @OneToOne(cascade =  CascadeType.ALL)
    private Room room;
    2 usages
    @ManyToMany
    private List<Member> participants;
```

*Figure 10: Class class*