U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Bases de Dados 2018

Home Gym

A mobile application for
workout challenges

Angelo Miguel Tenreiro Teixeira , up201606516
Henrique Melo Lima , up201606525
Rui Pedro Moutinho Moreira Alves , up201606746

07 – Março - 2018

Index

What is home gym?

Home Gym is a mobile application for challenging other people to do workout with you in a fun and interactive way. You get rewards by completing challenges and get feedback on how well you did, to help you improve for next time!

Our idea with this project is to create a Database to manage all the user and app information, keeping track of all the user's statistics, all on-going challenges, types of exercises, user reviews, and others (this subject will be specified in the following chapter).

Project's Specification

Any person using the application is a **User**. The user connects to the app with its facebook account, being characterized by his facebook ID, nickname, and by his score, calculated from the scores of all the challenges the user has participated in.

A **User** can participate in more than one **Challenge** at a time. Each Challenge is composed by an ID, by a start and ending date and information about if the challenge is public [(1)] or not. In each challenge there can be 2 or more participants. When the Challenge is created, it is also specified in which **Week days** the **Exercise Plan** associated to the Challenge should be executed (e.g. , the Challenge consists in completing an exercise plan every Monday, Wednesday and Friday from 15-03-2018 to 20-04-2018).

Associated to each challenge the user is participating in are stored its **Participation Details**, composed by the user's score in that challenge, the rating the user gives to that Challenge (after challenge is completed / after the user gives up) and the various **Executions** of the exercise plan related to that Challenge. Each Execution consists in the duration and the date the user completed that execution (so that the user can keep track of improvements).

An **Exercise Plan** is composed by it's unique ID, a recommended cooldown (number of days that are recommended between Exercise Plan executions) and by its difficulty (calculated by the average difficulty of the **Exercises** that compose the Exercise Plan). An **Exercise Plan** can be a **Default Plan** (created by the app developers) or it can be a **Custom Plan** (created by an User). A Custom Plan can be public [(1)] or not and saves the date of the last time it was used.

An **Exercise** is composed by its unique ID, a name, a description, an image (that visually describes the exercise), a link to a video that explains how the exercise should be executed, its difficulty (rated from 1 to 5) and its **Exercise Type**, which can be *Endurance, Strength, Flexibility* or *Balance*

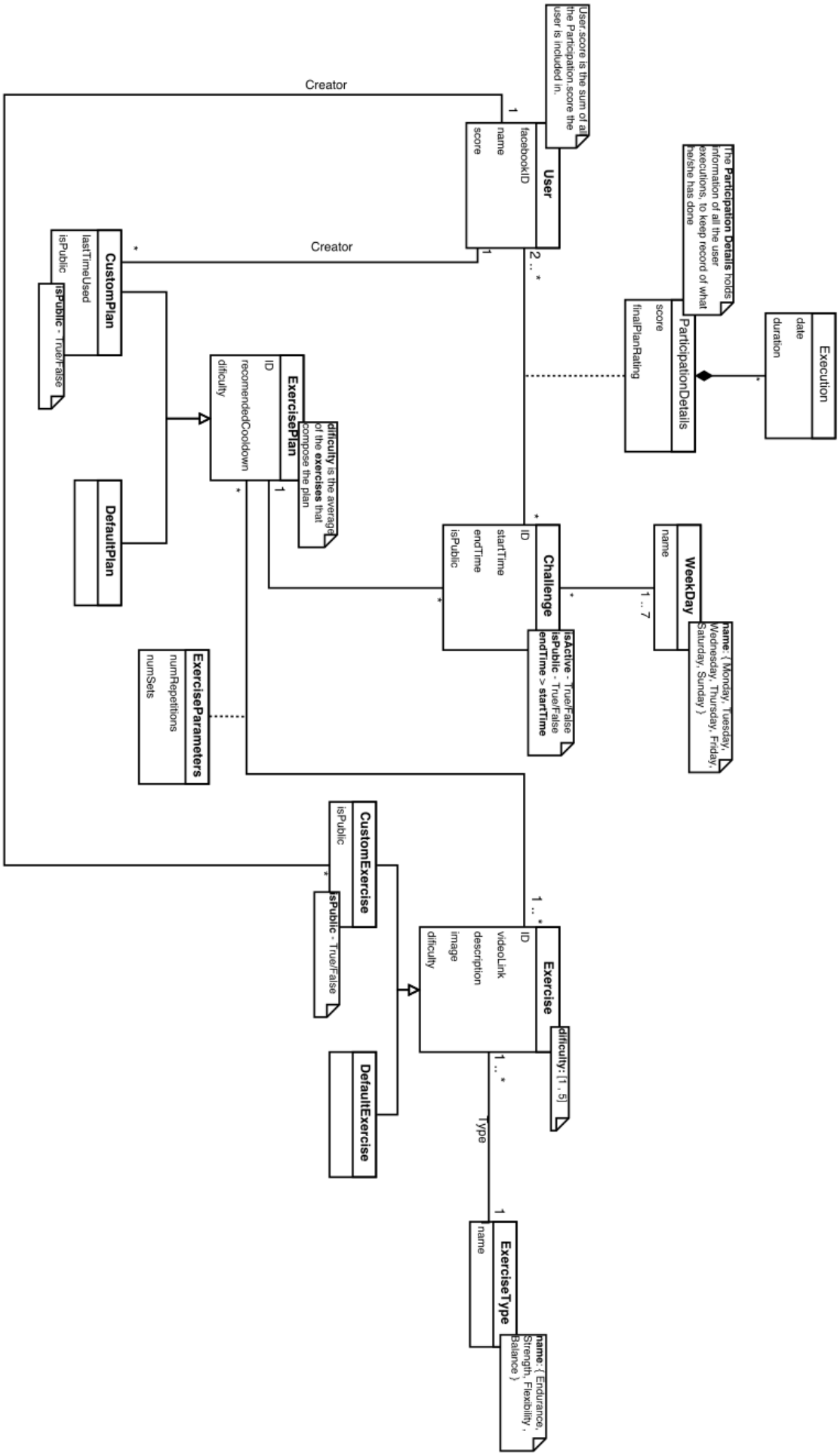
An exercise can be associated to many different exercise plans, and an Exercise Plan is composed by one or more Exercises. For each Exercise in an Exercise Plan, there are **Exercise Parameters**, that is, the number of

repetitions and number of sets for that Exercise execution (e.g. , 3 sets of 10 repetitions of push-ups , 3*10 = 30 push-ups).

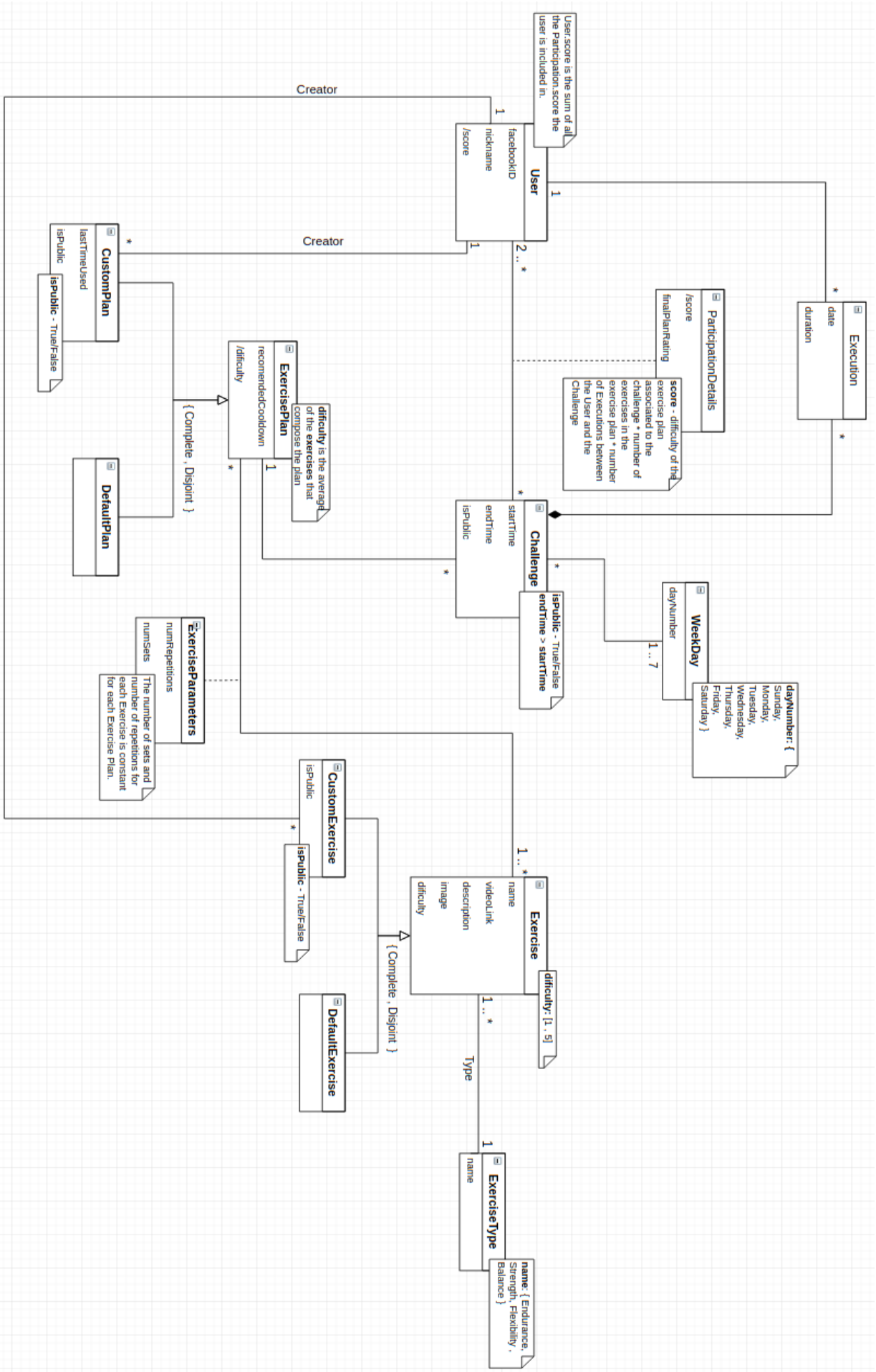
An **Exercise** can, simillarly to an Exercise Plan, be a **Default Exercise** (created by the developers) or it can be a **Custom Exercise** (created by an User). A Custom Exercise can be public [1] or not.

(1) – public : visible to other Users

Initial Conceptual Model



Reviewed Conceptual Model



The conceptual models (initial and reviewed) displayed above can also be found in a file attached to the submission for an easier view.
Initial Model: HomeGym_InitialConceptualModel.pdf
Reviewed Model: HomeGym_ReviewedConceptualModel.png

RELATIONAL MODEL

User (userID, nickname, userScore)
userID → nickname, userScore
userID is the *primary key*
userScore is a *derived attribute*

Challenge (challengeID, startTime, endTime, isPublic, exercisePlan→ExercisePlan)
challengeID → startTime, endTime, isPublic, exercisePlan
challengeID is the *primary key*
challengeID and exercisePlan are *foreign keys*

ParticipationDetails (userID→User, challengeID→Challenge, participationScore, finalPlanRating)
user, challenge → participationScore, finalPlanRating
user and challenge are the *composite primary key*
user and challenge are *foreign keys*
participationScore is a *derived attribute* (as explained in a future chapter)

Execution (idexecutionID date, duration, user→User, challenge→Challenge)
executionID → date, duration, user, challenge
executionID is the *primary key*
user and challenge are *foreign keys*

WeekDay (weekDayID, dayName)
weekDayID → dayName
dayName → weekDayID
weekDayID is the *primary key*

ExercisePlan (exercisePlanID, recomendedCooldown, difficulty)
exercisePlanID → recomendedCooldown, difficulty
exercisePlanID is the *primary key*
difficulty is a *derived attribute* (as explained in a future chapter)

CustomPlan (exercisePlanID→ExcercisePlan, lastTimeUsed, isPublic, creator→User)
exercisePlanID → lastTimeUsed, isPublic, creator
exercisePlanID is the *primary key*
exercisePlanID and creator are *foreign keys*

DefaultPlan (exercisePlanID→ExercisePlan)
exercisePlanID is the *primary key*
exercisePlanID is a *foreign key*

Exercise (exerciseID, name, videoLink, description, imageURL, difficulty, type→ExerciseType)
exerciseID → videoLink, description, imageURL, difficulty, type
exerciseID is the *primary key*
type is a *foreign key*

CustomExercise (exerciseID, isPublic, creator→User)
exerciseID → isPublic, creator
exerciseID is the *primary key*
exerciseID and creator are *foreign keys*

DefaultExercise (exerciseID→Exercise)
exerciseID is the *primary key*
exerciseID is a *foreign key*

ExerciseParameters (exercisePlanID→ExercisePlan, exerciseID→Exercise, numRepetitions, numSets)

exercisePlanID, exerciseID → numRepetitions, numSets

exercisePlanID and exerciseID are the *composite primary key*

exercisePlanID and exerciseID are *foreign keys*

ExerciseType (exerciseTypeID, name)

exerciseTypeID → name

name → exerciseTypeID

exerciseTypeID is the *primary key*

ChallengeDay (challengeID→Challenge, weekDayID→WeekDay)

challengeID and weekDayID are the *composite primary key*

challengeID and weekDayID are *foreign keys*

FUNCTIONAL DEPENDENCIES AND NORMAL FORM ANALYSIS

In each of the relatitions described in the previous chapter, the left side of the functional dependencies is a key for that relation, that is, the closure of the attributes in the left side is **all** the attributes in that relation, as shown in the following paragraphs:

User:

$\{ facebookID \}^+ = \{ facebookID, nickname, score \}$

Challenge:

$\{ id \}^+ = \{ id, startTime, endTime, isPublic, exercisePlan \}$

ParticipationDetails:

$\{ user, challenge \}^+ = \{ user, challenge, score, finalPlanRating \}$

Execution:

$\{ id \}^+ = \{ id, date, duration, user, challenge \}$

WeekDay:

$\{ id \}^+ = \{ id, dayName \}$

ExercisePlan:

$\{ id \}^+ = \{ id, recomendedCooldown, difficulty \}$

CustomPlan:

$\{ id \}^+ = \{ id, lastTimeUsed, isPublic, creator \}$

DefaultPlan:

$\{ id \}^+ = \{ id \}$

Exercise:

$\{ id \}^+ = \{ id, name, videoLink, description, image, difficulty, type \}$

CustomExercise:

$\{ id \}^+ = \{ id, isPublic, creator \}$

DefaultExercise:

$\{ id \}^+ = \{ id \}$

ExerciseParameters:

$\{ exercisePlan, exercise \}^+ = \{ execisePlan, exercise, numRepetitions, numSets \}$

ExerciseType:

$\{ id \}^+ = \{ id, name \}$

ChallengeDay:

$\{ challenge, weekDay \}^+ = \{ challenge, weekDay \}$

Therefore, since in each of the relations the left side of the functional dependencies is a key for that relation, the relational model is in the **Boyce-Codd Normal Form,** *BCNF* (no normal form violations were found). Since the **3rd Normal Form**, *3NF*, is a super set of the BCNF, that is, every relation in the BCNF is also in the 3NF, the relational model is also in the **3rd Normal Form**.

RESTRICTIONS

User:

- **facebookID** is the primary key (key restriction , PRIMARY KEY)
- **score** is the sum of all the *ParticipationDetails.score* the user is in (using triggers, to be implemented futurely), default value=0 (DEFAULT (0)) and can't be null (NOT NULL)
- **nickname** must have [6..48] characters and must be unique (UNIQUE) and can't be null (NOT NULL)

Challenge:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **startTime** and **endTime** are both dates. *endTime* must be greater than *startTime* (check restriction, CHECK endTime >= startTime) and both can't be null (NOT NULL)
- **IsPublic** default value is *true* (DEFAULT (1)) and can't be null (NOT NULL)
- **exercisePlan** is a foreign key (referential integrity, FOREIGN KEY) and can't be null (NOT NULL)

ParticipationDetails:

- **user** and **challenge** are the composite primary key (key restriction, PRIMARY KEY(user,challenge)) and are both foreign keys (referential integrity, FOREIGN KEY)
- **finalPlanRating** default value is **Null** (Default NULL) and, if not, must ∈ [1,10] (CHECK finalPlanRating >= 1 *and* finalPlanRating <= 10)
- **score** is calculated by multiplying the difficulty of the Exercise Plan associated to the Challenge, the number of exercises associated to that plan and the number executions between the user and the challenge (using triggers, to be implemented futurely) and can't be null (NOT NULL)
- **score** "Simplified Expression":
  score = planDifficulty * numExercisesInPlan * numExecutions

WeekDay:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **dayName** can't be null (NOT NULL) and can hold the following values, with (CHECK dayName=='Monday' or dayName=='Tuesday' or …):
  - Monday
  - Tuesday
  - Wednesday
  - Thursday
  - Friday
  - Saturday
  - Sunday

ChallengeDay:

- **challenge** and **weekDay** are the composite primary key (key restriction, PRIMARY KEY(challenge,weekDay)) and are both foreign keys (referential integrity, FOREIGN KEY)

ExerciseType:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **name** cannot be null (NOT NULL) and must be unique (UNIQUE)

ExercisePlan:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **recomendedCooldown** can be null (not set) and, if set, must be greater or equal than one day (CHECK recommendedCooldown > 0)
- **difficulty** is the average of all the *Exercise.difficulty* associated to this plan (using triggers, to be implemented futurely)

DefaultPlan:

- **id** is the primary key (key restriction, PRIMARY KEY) and a foreign key (referential integrity, FOREIGN KEY)

CustomPlan:

- **id** is the primary key (key restriction, PRIMARY KEY) and a foreign key (referential integrity, FOREIGN KEY)
- **creator** is a forein key (referential integrity, FOREIGN KEY) and can't be null (NOT NULL)
- **isPublic** default value is *true* (DEFAULT (1)) and can't be null (NOT NULL)

Exercise:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **name** cannot be null (NOT NULL)
- **description** cannot be null (NOT NULL) and its default value is "No description available" (DEFAULT 'No description available')
- **difficulty** default value is 3 (DEFAULT (3)), must be within the range [1..5] (CHECK difficulty >= 1 and difficulty <= 5) and can't be null (NOT NULL)
- **type** cannot be null (NOT NULL) and is a foreign key (referential integrity, FOREIGN KEY)

DefaultExercise:

- **id** is the primary key (key restriction, PRIMARY KEY) and a foreign key (referential integrity, FOREIGN KEY)

CustomExercise:

- **id** is the primary key (key restriction, PRIMARY KEY) and a foreign key (referential integrity, FOREIGN KEY)
- **creator** is a forein key (referential integrity, FOREIGN KEY) and can't be null (NOT NULL)
- **isPublic** default value is *true* (DEFAULT (1)) and can't be null (NOT NULL)

ExerciseParameters:

- **exercisePlan** and **exercise** are the composite primary key (key restriction, PRIMARY KEY(exercisePlan,exercise)) and are both foreign keys (referential integrity, FOREIGN KEY)
- **numRepetitions** cannot be null (NOT NULL) and must be greater or equal than one (CHECK numRepetitions >= 1)
- **numSets** cannot be null (NOT NULL) and must be greater or equal than one (CHECK numSets >= 1)

Execution:

- **id** is the primary key (key restriction, PRIMARY KEY)
- **date** cannot be null (NOT NULL)
- **duration** cannot be null (NOT NULL) and must be greater or equal than one minute (CHECK duration >= 1)
- **user** cannot be null (NOT NULL) and is a foreign key (referential integrity, FOREIGN KEY)
- **challenge** cannot be null (NOT NULL) and is a foreign key (referential integrity, FOREIGN KEY)

QUERIES

The 10 implemented queries are presented and decribed below, listed in natural language.

Query 1: Obtain all users who, in a certain period of time (that is, between two specific dates, indicated in the query) are participating in at least one challenge and, if so, the number of challenges they are participating in. This query may be useful for statistics reasons or to check, for example, who will be doing a certain challenge in the following week.

Query 2: Obtain all challenge history of a certain user (indicated in the query), that is, all the challenges the user participated in, with the associated score, start and finish dates, number of participants and whether the challenge is public or not. This query is useful for statistics reasons or for a user to track his/her own progress.

Query 3: Obtain all the exercise plans who are mostly composed (more than 50%) by exercises of a specific type (for example, Strength, to be indicated in the query). This query is useful for a user that wants to know which exercise plans are indicated for the specific kind of training they are looking for.

Query 4: Obtain, for a specific user (to be indicated in the query) all the exercises that he has executed, with the respective frequency (how many times he has executed that exercise), their type and the exercise type frequency (how many times he has executed exercises of that type). In addition, there are also obtained the exercises he has not yet performed, but that belong to the exercise types the user usually performs. This query is useful for suggesting new exercises to a user and also for a user to keep track of what kind of exercises he/she has been doing, so that they know which exercises to prioritize/avoid.

Query 5: Obtain the application top 3 user podium (the 3 users with the best score). This query is useful for displaying weekly or monthly rankings, showing the best players in the present moment.

Query 6: Obtain all the challenges in the application, ordered by rating in descending order and, for the tied ones, ordered by number of votes in descending order. This query is usefullfor statistics reasons and to show users the challenges that are trending in the application.

Query 7: Obtain, for a specific exercise (to be indicated in the query), all the users who have performed that specific exercise and how many repetitions were perform by him/her. This query is useful for showing users who else is doing that specific exercise to help their exercise choices.

Query 8: Obtain all the exercise plans that contain an exercise with a simillar name to the one indicated in the query. This query is useful for Users that want to find exercise that contain, for example, “Jumps” in the name. This way, users can track down all exercises that involve jumping (for this example).

Query 9: Obtain, for a specific user (to be indicated in the query) all the Custom Exercise Plans the he/she created and how customized they are, that is, the percentage of user made exercises it contains. This query is useful to check which users are contributing to the application by inovating with new exercise plans with their own exercises and for other statistics reasons.

Query 10: Obtain all the custom exercises ordered by their popularity (that is, how many times they were executed), how many times they were executed and their creator. This query is useful for users that want to track which custom made exercises are trending and to check which users are contributing to the app community with the best exercises.

Side note: All the above queries that mention a parameter “to be indicated in the query”, already have an example parameter written in the query. However, this parameters may be changed to query for different information.

Allthough there were only required 10 queries for this submission, there were implemented a few more that were considered strong candidates to make the “top 10” queries. These queries can be found in the folder **Other Queries**, attached to the submission.

TRIGGERS

The 3 implemented triggers are presented and decribed below, listed in natural language.

Trigger 1: This trigger updates an ExercisePlan's difficulty based on the average of the Exercises associated to it. To do so, after an ExerciseParameters tuple (linking an Exercise and an ExercisePlan) is inserted on the data base, the ExercisePlan difficulty is updated.

Trigger 2: This trigger updates the Participation Score of a certain User in a certain Challenge (a ParticipationScore tupple) based on the number of executions of the Challenge, its difficulty and the number of Exercises of the ExercisePlan associated to the Challenge. To do so, after an Execution tuple is inserted on the data base, the respective user associated to it is updated, based on the Challenge associated to the inserted Execution.

Trigger 3: This trigger updates a User's score based on the sum of all the ParticipationDetails of the challenges it has participated in. To do so, after a ParticipationDetails tuple is updated on the data base, the respective user's score is updated.

To run the above triggers, the script 'criar.sql' should be read, followed by the 'gatilhoN_adiciona.sql' scripts, and then followed by the 'povoar.sql' scripts. After that, the triggers can be tested using the 'gatilhoN_verifica' scripts and removed using the 'gatilhoN_remove.sql' scripts.

The triggers aboved described are also dependant to each other. They should all the added to the data base in their correct order (Trigger 1, followed by Trigger 2, followed by Trigger 3) in order for data to make sense and be correct.

The triggers above described are triggered after an insertion of a tuple in the respective relation (except Trigger 3, which is triggered after an update). However, there should also be implemented triggers to update the data after data updating and deleting. Those triggers were not implemented, since only three triggers were required.

All the Triggers implement restrictions that were specified in the previous delivery, being them:

- ExercisePlan.difficulty is the average of the difficulties of all Exercises associated to it (Trigger 1)
- ParticipationDetails.score is equal to the number of executions times the difficulty of the exercise challenge times the number of exercises in the plan (Trigger 2)
- User.score is equal to the sum of the scores of the Challenges he/she has participated in (Trigger 3)