BD - P7G6

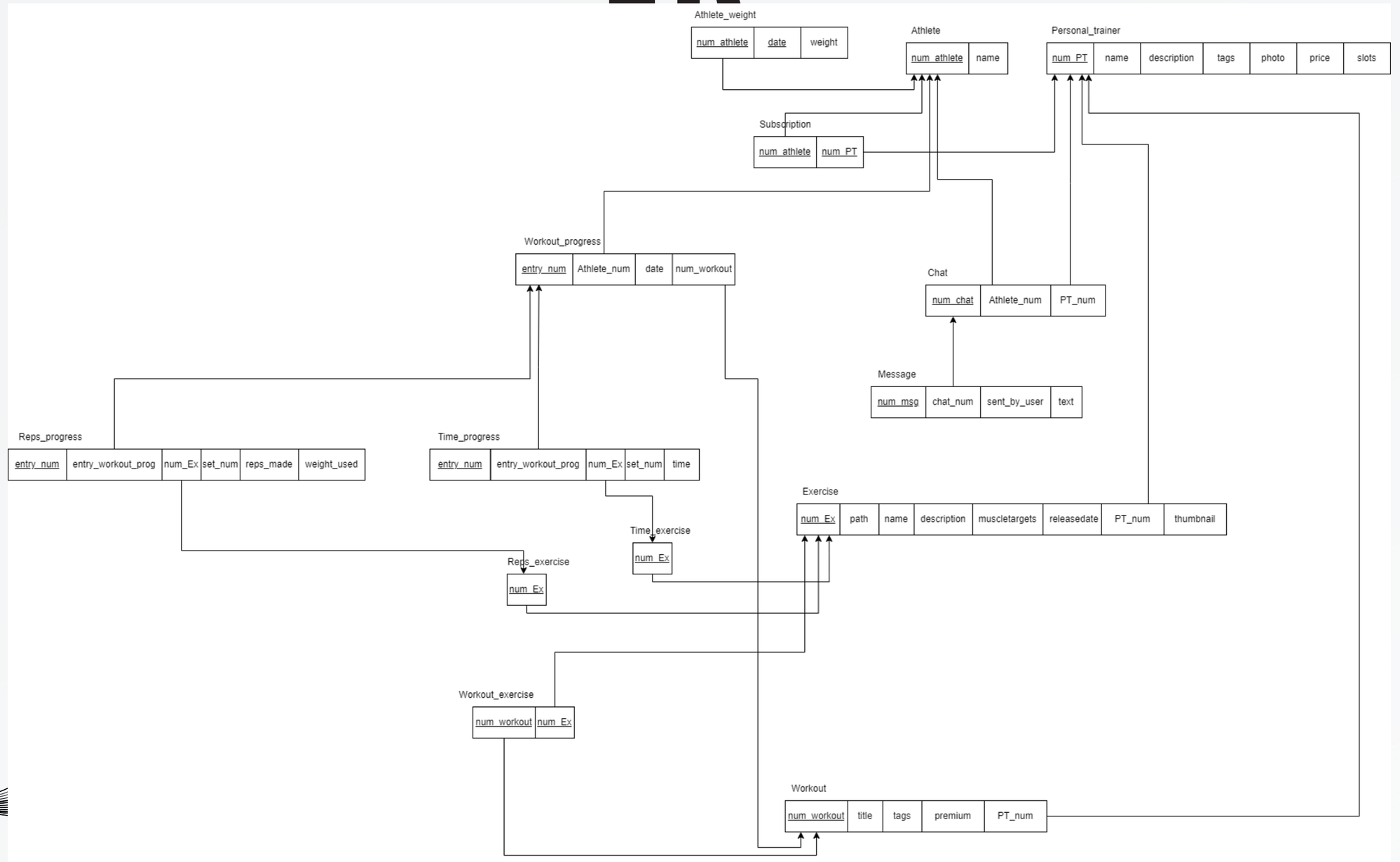
POCKET COACH

GABRIEL COSTA Nº109050, HENRIQUE COELHO Nº108342

INTRODUÇÃO

- Consiste numa aplicação de sistema de suporte a exercício físico individual e remoto
- Os PTs podem criar workouts com exercícios diferenciados para os atletas
- Os atletas podem visualizar estes workouts e registar o seu progresso em cada exercício do respetivo workout
- Os atletas podem subscrever-se aos PTs, de modo a terem acesso aos workouts premium

ER


Athlete_weight
num_athlete
date
weight
Athlete
num_athlete
name
Personal_trainer
num_PT
name
description
tags
photo
price
slots
Subscription
num_athlete
num_PT
Workout_progress
entry_num
Athlete_num
date
num_workout
Chat
num_chat
Athlete_num
PT_num
Message
num_msg
chat_num
sent_by_user
text
Reps_progress
entry_num
entry_workout_prog
num_Ex
set_num
reps_made
weight_used
Time_progress
entry_num
entry_workout_prog
num_Ex
set_num
time
Exercise
num_Ex
path
name
description
muscletargets
releasedate
PT_num
thumbnail
Time_exercise
num_Ex
Reps_exercise
num_Ex
Workout_exercise
num_workout
num_Ex
Workout
num_workout
title
tags
premium
PT_num

Triggers

- Trigger para verificar se um personal trainer tem ainda vagas disponíveis e atualizar as vagas após inserir

```
GO
CREATE TRIGGER check_PT_slots ON subscription
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @num_PT INT, @slots INT;
    SELECT @num_PT = num_PT FROM inserted;

    SELECT @slots = slots FROM personal_trainer WHERE num_PT = @num_PT;

    IF @slots <= 0
    BEGIN
        RAISERROR ('Sem slots disponíveis para este Personal Trainer.', 16, 1);
        ROLLBACK TRAN; -- Anula a inserção
    END
    ELSE
    BEGIN
        UPDATE personal_trainer SET slots = slots - 1 WHERE num_PT = @num_PT;
    END
END
GO
```

Triggers

- Trigger para apagar todos os dados relacionados a um personal trainer ao apagá-lo

```
GO
CREATE TRIGGER delete_related_PT_data ON personal_trainer
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @num_PT INT;
    SELECT @num_PT = num_PT FROM deleted;

    DECLARE @num_ex TABLE (num_ex INT);
    INSERT INTO @num_ex
    SELECT num_ex FROM exercise WHERE PT_num = @num_PT;

    DELETE FROM subscription WHERE num_PT = @num_PT;

    -- Before deleting chat, delete all related data
    DELETE FROM message WHERE chat_num IN (SELECT num_chat FROM chat WHERE PT_num = @num_PT);
    DELETE FROM chat WHERE PT_num = @num_PT;

    -- Before deleting exercise, delete all related data
    DELETE FROM workout_exercise WHERE num_workout IN (SELECT num_workout FROM workout WHERE PT_num = @num_PT);
    DELETE FROM time_progress WHERE num_ex IN (SELECT num_ex FROM @num_ex);
    DELETE FROM reps_progress WHERE num_ex IN (SELECT num_ex FROM @num_ex);
    DELETE FROM reps_exercise WHERE num_ex IN (SELECT num_ex FROM @num_ex);
    DELETE FROM time_exercise WHERE num_ex IN (SELECT num_ex FROM @num_ex);
    DELETE FROM workout_progress WHERE num_workout IN (SELECT num_workout FROM workout WHERE PT_num = @num_PT);
    DELETE FROM exercise WHERE PT_num = @num_PT;

    DELETE FROM workout WHERE PT_num = @num_PT;

    DELETE FROM personal_trainer WHERE num_PT = @num_PT;
END
```

Triggers

- Trigger para apagar todos os dados relacionados a um atleta ao apaga-lo

```
GO
CREATE TRIGGER delete_related_athlete_data ON athlete
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @num_athlete INT;
    SELECT @num_athlete = num_athlete FROM deleted;

    DELETE FROM athlete_weight WHERE num_athlete = @num_athlete;
    DELETE FROM subscription WHERE num_athlete = @num_athlete;

    -- Before deleting chat, delete all related data
    DELETE FROM message WHERE chat_num IN (SELECT num_chat FROM chat WHERE Athlete_num = @num_athlete);
    DELETE FROM chat WHERE Athlete_num = @num_athlete;

    -- Before deleting workout_progress, delete all related data
    DELETE FROM time_progress WHERE entry_num IN (SELECT entry_num FROM workout_progress WHERE Athlete_num = @num_athlete);
    DELETE FROM reps_progress WHERE entry_num IN (SELECT entry_num FROM workout_progress WHERE Athlete_num = @num_athlete);
    DELETE FROM workout_progress WHERE Athlete_num = @num_athlete;

    DELETE FROM athlete WHERE num_athlete = @num_athlete;
END
```

Stored Procedures

- SP para obter os exercícios de um workout

```
GO
CREATE PROCEDURE GetWorkoutExercises
    @WorkoutId INT
AS
BEGIN
    SELECT
        we.num_ex AS ExerciseId,
        e.name AS ExerciseName,
        e.description AS ExerciseDescription,
        e.muscletargets AS MuscleTargets,
        e.releasedate AS ReleaseDate,
        e.path AS ExercisePath,
        e.thumbnail AS ExerciseThumbnail
    FROM
        workout_exercise we
    JOIN
        exercise e ON we.num_ex = e.num_ex
    WHERE
        we.num_workout = @WorkoutId
END
GO
```

Stored Procedures

- SP para obter o progresso de todos os exercícios de um workout para um atleta

```
GO
CREATE PROCEDURE GetWorkoutExerciseProgressForAthlete
    @num_workout INT,
    @num_athlete INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Seleciona o progresso dos exercícios de time
    SELECT
        we.num_ex AS ExerciseID,
        e.name AS ExerciseName,
        tp.set_num AS SetNumber,
        tp.time AS Time,
        NULL AS RepsMade,
        NULL AS WeightUsed
    FROM
        workout_exercise we
    INNER JOIN
        exercise e ON we.num_ex = e.num_ex
    INNER JOIN
        time_exercise te ON e.num_ex = te.num_ex
    INNER JOIN
        workout_progress ep ON ep.Athlete_num = @num_athlete
    INNER JOIN
        time_progress tp ON ep.entry_num = tp.entry_num AND tp.num_ex = te.num_ex
    WHERE
        we.num_workout = @num_workout

    UNION ALL

    -- Seleciona o progresso dos exercícios de reps
    SELECT
        we.num_ex AS ExerciseID,
        e.name AS ExerciseName,
        rp.set_num AS SetNumber,
        NULL AS Time,
        rp.reps_made AS RepsMade,
        rp.weight_used AS WeightUsed
    FROM
        workout_exercise we
    INNER JOIN
        exercise e ON we.num_ex = e.num_ex
    INNER JOIN
        reps_exercise re ON e.num_ex = re.num_ex
    INNER JOIN
        workout_progress ep ON ep.Athlete_num = @num_athlete
    INNER JOIN
        reps_progress rp ON ep.entry_num = rp.entry_num AND rp.num_ex = re.num_ex
    WHERE
        we.num_workout = @num_workout;
END
GO
```

UDFs

- UDF para um dado workout_progress, devolve os progressos nos exercícios associados (diferentes tipos de progressos)

```
CREATE FUNCTION GetWorkoutProgressDetails(@entry_num INT)
RETURNS TABLE
AS
RETURN
(
    SELECT
        rp.set_num AS SetNumber,
        rp.reps_made AS RepsMade,
        rp.weight_used AS WeightUsed,
        NULL AS Time
    FROM
        reps_progress rp
    WHERE
        rp.entry_workout_prog = @entry_num

    UNION ALL

    SELECT
        tp.set_num AS SetNumber,
        NULL AS RepsMade,
        NULL AS WeightUsed,
        tp.time AS Time
    FROM
        time_progress tp
    WHERE
        tp.entry_workout_prog = @entry_num
);
```

UDFs

- UDF que faz a estimativa de duração do workout de um athlete

```
GO
CREATE FUNCTION GetWorkoutDuration(@num_workout INT, @num_athlete INT)
RETURNS INT
AS
BEGIN
    DECLARE @Duration INT = 0;

    -- Variável temporária para armazenar os resultados combinados
    DECLARE @CombinedResults TABLE (
        ExerciseID INT,
        SetNumber INT,
        Time INT,
        RepsMade INT,
        WeightUsed INT
    );

    -- Inserir resultados do time_progress
    INSERT INTO @CombinedResults (ExerciseID, SetNumber, Time, RepsMade, WeightUsed)
    SELECT
        tp.num_ex AS ExerciseID,
        tp.set_num AS SetNumber,
        tp.time AS Time,
        NULL AS RepsMade,
        NULL AS WeightUsed
    FROM
        workout_progress wp
    INNER JOIN
        time_progress tp ON wp.entry_num = tp.entry_workout_prog
    WHERE
        wp.num_workout = @num_workout
        AND wp.Athlete_num = @num_athlete;

    -- Inserir resultados do reps_progress
    INSERT INTO @CombinedResults (ExerciseID, SetNumber, Time, RepsMade, WeightUsed)
    SELECT
        rp.num_ex AS ExerciseID,
        rp.set_num AS SetNumber,
        NULL AS Time,
        rp.reps_made AS RepsMade,
        rp.weight_used AS WeightUsed
    FROM
        workout_progress wp
    INNER JOIN
        reps_progress rp ON wp.entry_num = rp.entry_workout_prog
    WHERE
        wp.num_workout = @num_workout
        AND wp.Athlete_num = @num_athlete;

    -- Calcular a duração total
    SELECT
        @Duration = @Duration +
        CASE
            WHEN cr.Time IS NOT NULL THEN cr.Time
            ELSE 30
        END
    FROM
        @CombinedResults cr;

    RETURN @Duration;
END
GO
```

Views

- View para exibir os 3 personal trainers mais baratos

```
GO
CREATE VIEW Top3CheapestPTs AS
SELECT TOP 3
    num_PT,
    name,
    description,
    tags,
    photo,
    price,
    slots
FROM
    personal_trainer
ORDER BY
    price ASC;
GO
```

Indexes

Um atleta precisará de “saber” a que Personal Trainers está subscrito para ter acesso aos workouts

```
CREATE INDEX idx_subscription_num_athlete ON subscription(num_athlete);
```

```
CREATE INDEX idx_exercise_PT_num ON exercise(PT_num);

CREATE INDEX idx_workout_PT_num ON workout(PT_num);
```

Um personal trainer utilizará constantemente os exercicios criados por si e um atleta consultará os workouts dos seus PTs

DEMO

THANK YOU FOR LISTENING