PocketCoach

Base de dados de gestão de PTs e atletas

Licenciatura de Engenharia de Computadores e Informática

Base de dados P7G6

Gabriel Gonçalves Costa | 109050

Henrique Cristóvão Coelho | 108342

05/06/2024

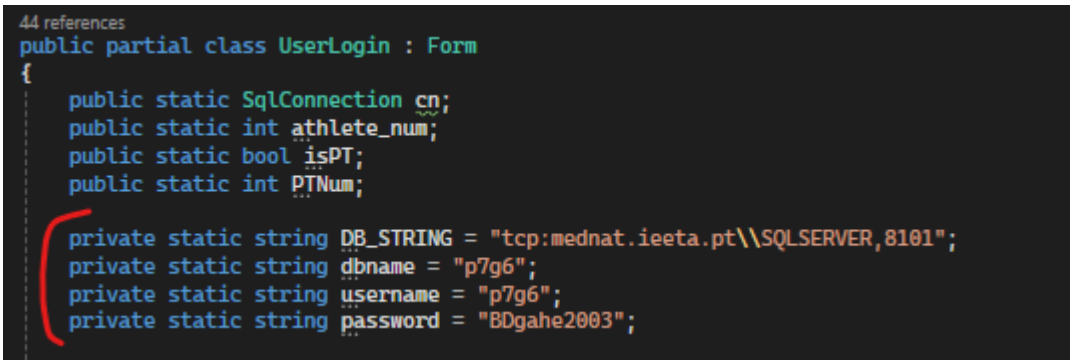
Índice

1. Introdução....3
2. Análise de Requisitos....4
3. DER....5
3.1 Diagrama Entidade Relação obtido....5
3.2 Entidades e multiplicidade....5
4. Esquema Relacional da BD....6
5. SQL DDL e SQL DML....6
6. Normalização....6
7. Índices....7
8. Triggers....8
9. Stored Procedures....9
10. UDF....13
11. Views....14
12. Conclusão....15

1. Introdução

No âmbito da unidade curricular de Base de Dados, pertencente à Licenciatura de Engenharia de Computadores e Informática, foi-nos proposta a criação de um projeto funcional, aplicável ao mundo real. O nosso grupo decidiu criar um sistema de interação entre Personal Trainers e atletas, de modo a que estes atletas possam ter uma melhor evolução na sua jornada de "fitness".

Para que seja possível testar a nossa aplicação noutra base de dados para além daquela que nos foi fornecida pelo ieeta, é necessário aceder ao ficheiro UserLogin.cs e mudar os atributos BD_STRING,dbname, username e password para as credenciais desejadas.



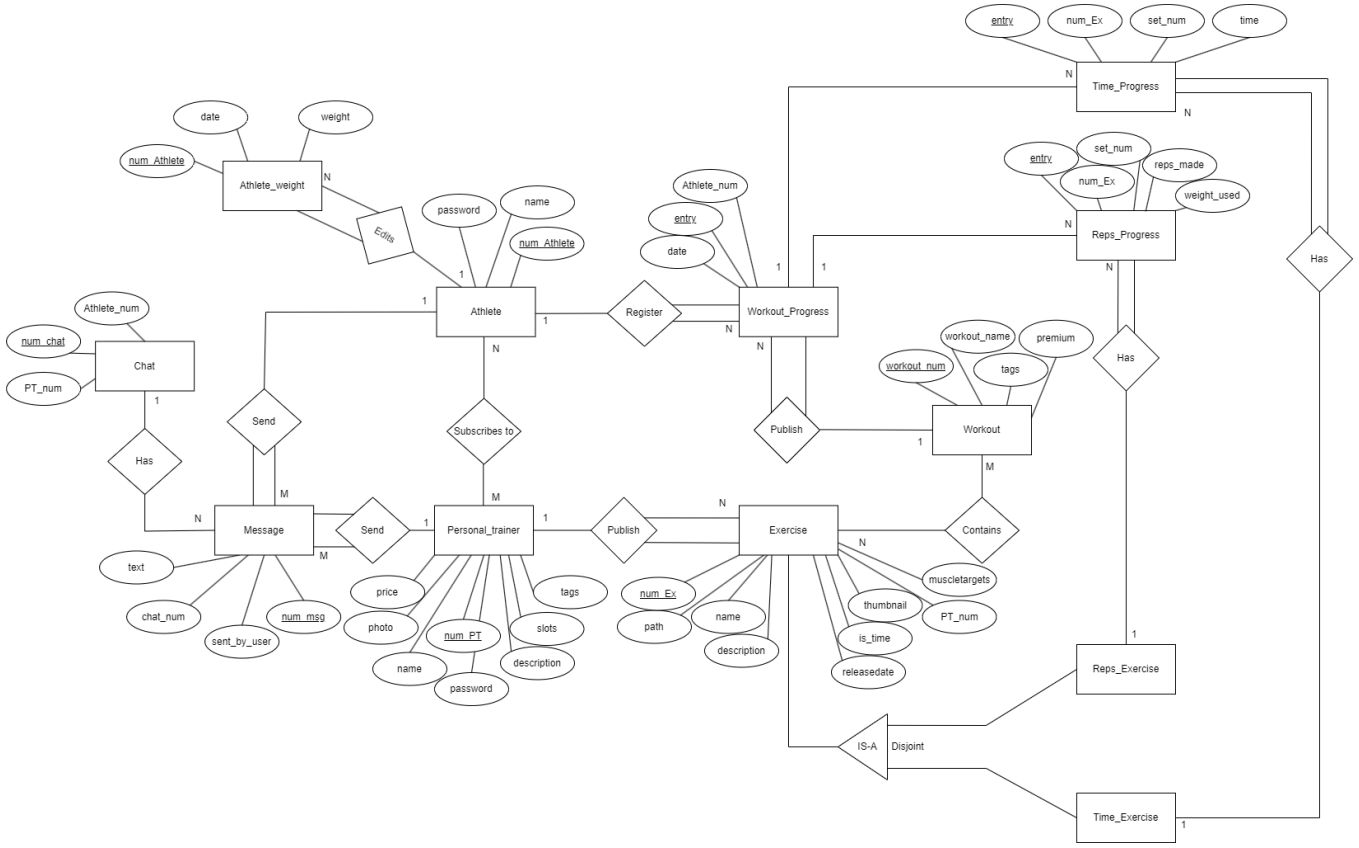
2. Análise de Requisitos

O nosso tema tem os seguintes requisitos:

- O Personal Trainer deve conseguir criar workouts específicos para cada atleta;
- Cada workout tem uma variedade de exercícios;
- O atleta deve ter referências para erros comuns;
- Os atletas devem ser capazes de controlar o próprio peso. Se o atleta for premium, o instrutor deverá ter acesso;
- O atleta deve ser capaz de registar quantas repetições ele faz em cada exercício;
- O Personal Trainer deve ser capaz de compartilhar vídeos e comunicar com os atletas (inscritos nele) para tirar dúvidas.

3. DER

3.1 Diagrama Entidade Relação obtido

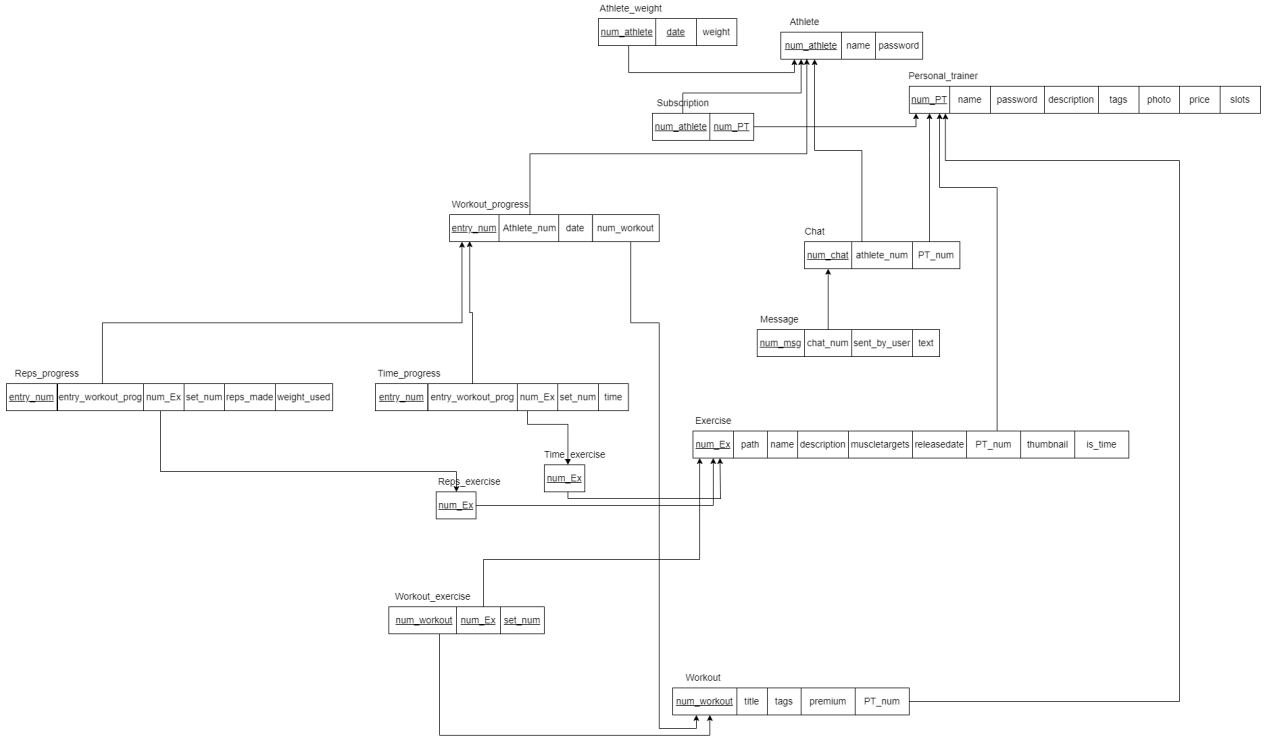


3.2 Entidades e multiplicidade

- **Personal Trainer (PT):** o atributo chave que a identifica é o ***num_PT*** e tem outros atributos (não únicos) que são price, photo, name,password, description, slots e tags;
- **Exercise:** um PT pode criar vários Exercise's, o atributo chave que a identifica é o ***num_Ex*** e tem outros atributos (não únicos) que são path, title, thumbnail, releasedate, muscletargets, description e PT_num (chave estrangeira);
- **Workout:** este é um conjunto de Exercise's, esta é criada pelo PT também, o atributo chave que a identifica é o ***workout_num*** e tem outros atributos (não únicos) que são workout_name, tags e num_ex (chave estrangeira);
- **Workout Progress:** o atributo chave que a identifica é o ***entry_num*** e tem outros atributos (não únicos) que são Athlete_num (chave estrangeira), date e num_workout (chave estrangeira)
- **Time Progress e Reps Progress:** o atributo chave que os identifica é o ***num_ex*** e contêm uma referência para o "time_exercise" (num_ex) e "reps_exercise" (num_ex), respectivamente, assim como uma referência para o workout_progress (entry_workout_prog). Por fim, têm informação do set e do progresso no exercício ("time" ou "reps_made" e "weight_used", respetivamente)

- **Time Exercise e Reps Exercise:** o atributo chave que os identifica é o ***num_ex*** e referencia o “num_ex” de “exercise”.
- **Athlete:** o atributo chave que a identifica é o ***num_athlete*** e tem outros atributos (não únicos) que são o “name” e a “password” para fazer o login e identificar o atleta.
- **Athlete Weight:** o atributo chave que a identifica é o **num_athlete** e tem outros atributos (não únicos) que são a data em que o peso foi registado e o valor do peso.
- **Chat:** Contém a informação dos elementos que participam no diálogo (athelete_num e PT_num) assim como o nº do chat para ser mais facilmente referenciado pelo “Message”
- **Message:** Contém as mensagens enviadas na conversa (text) com um identificador que indica se foi o PT ou o atleta que enviou a mensagem (sent_by_user com valor 0 ou 1), identificado pelo “num_msg”.

4. Esquema Relacional da BD



5. SQL DDL e SQL DML

Ver ficheiros **SQL_DDL.sql** e **SQL_DML.sql** na pasta **SQL_Scripts**.

6. Normalização

Consideramos que não é preciso efetuar qualquer tipo de normalização, visto que já se encontrava normalizado.

7. Índices

Uma vez que os atletas precisarão de ter a informação das suas subscrições para receber os workouts a que tem acesso (funcionalidade mais utilizada pelo atleta), criamos um índice para obter esta informação mais depressa.

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

Um personal trainer utilizará constantemente os exercícios criados por si e um atleta consultará os workouts dos seus PTs várias vezes. Desta forma, decidimos criar um índice para o número do personal trainer nos exercícios e nos workouts. Nos workouts, o índice está designado para o número do PT pois faremos a pesquisa após saber a que PTs o atleta está subscrito, usando a tabela “subscription”.

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

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

8. Triggers

Relativamente aos Triggers, implementamos três no nosso projeto/BD, tendo estes as seguintes funcionalidades:

- Um Trigger que serve para cada vez que seja feita uma subscrição de um Atleta num PT (INSERT na tabela subscription) é verificada se este ainda tem slots disponíveis, caso não tenha a inserção é anulada, mas se tiver é subtraído o número de slots disponíveis pelo respetivo PT a que está a subscrever:

```
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
```

Fig.1 - Check PT Slots

- E por fim, dois outros Triggers. O primeiro, em que, antes de ser apagado um determinado Atleta, é primeiramente apagado os dados associados a este, e só após estes terem sido apagados é que o Atleta será apagado, e um outro que fará o mesmo, mas invés de ser um Atleta será um Personal Trainer:
    - Trigger do Atleta:

```
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
```

Fig.2 - Delete Related Athlete Data

- Trigger do Personal Trainer:

```
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
```

Fig.3 - Delete Related PT Data

9. Stored Procedures

No que toca as Stored Procedures, criámos cinco, sendo cada uma delas:

- Stored Procedure para adquirir os Exercícios de um dado Workout:

```
GO
CREATE PROCEDURE GetWorkoutExercises
    @num_workout INT
AS
BEGIN
    SELECT * FROM
        workout_exercise we
    JOIN
        exercise e ON we.num_ex = e.num_ex
    WHERE
        we.num_workout = @num_workout
END
GO
```

Fig.4 - Get Workout Exercises

- Stored Procedure para adquirir todos os Workouts acessíveis a um determinado Atleta, isto inclui os Premium como também os Free:

```
CREATE PROCEDURE GetAccessibleWorkouts
    @num_athlete INT
AS
BEGIN
    SELECT w.num_workout, w.title,w.tags,w.premium,w.PT_num
    FROM athlete JOIN subscription ON athlete.num_athlete=subscription.num_athlete
        JOIN workout AS w ON subscription.num_PT=w.PT_num
    WHERE athlete.num_athlete=@num_athlete OR premium=0;

END

GO
```

Fig.5 - Get Accessible Workouts

- Um outro SP que irá devolver os Exercise Progress (Time Progress e Reps Progress) de um dado Workout Progress(que será de um respetivo Atleta):

```sql
CREATE PROCEDURE GetWorkoutExerciseProgressForAthlete
    @entry_num INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Seleciona o progresso dos exercícios de time
    SELECT  tp.num_ex AS ExerciseID,
        e.name AS ExerciseName,
        tp.set_num AS SetNumber,
        tp.time AS Time,
        NULL AS RepsMade,
        NULL AS WeightUsed
    FROM time_progress AS tp
        JOIN workout_progress ON workout_progress.entry_num=tp.entry_workout_prog
        JOIN exercise AS e ON tp.num_ex = e.num_ex
    WHERE workout_progress.entry_num = @entry_num

    UNION ALL

    SELECT rp.num_ex AS ExerciseID,
        e.name AS ExerciseName,
        rp.set_num,
        NULL AS time,
        rp.reps_made,
        rp.weight_used
    FROM reps_progress AS rp
        JOIN workout_progress ON workout_progress.entry_num=rp.entry_workout_prog
        JOIN exercise AS e ON rp.num_ex = e.num_ex
    WHERE workout_progress.entry_num = @entry_num

    ORDER BY SetNumber, ExerciseID

END

GO
```

Fig.6 - Get Workout Exercise Progress For Athlete

- Criamos também um sp para, ao retirar uma subscrição feita a um Personal Trainer, apagar as conversas entre o Personal Trainer e o atleta que quis retirar a subscrição.

```
GO
CREATE PROCEDURE DeleteSub
    @num_athlete INT,
    @num_PT INT

AS
BEGIN
    SET NOCOUNT ON;

    DELETE FROM subscription WHERE num_athlete=@num_athlete and num_PT=@num_PT;

    DECLARE @num_chat INT;
    select @num_chat=num_chat from chat where athlete_num=@num_athlete and PT_num=@num_PT;

    DELETE from message where chat_num = @num_chat;
    DELETE from chat where num_chat = @num_chat;


END
```

Fig.7 Delete Subscription

- Por fim, um SP para devolver todos os Workout Progress e respetiva informação para um dado atleta:

```
GO
CREATE PROCEDURE GetAthleteWorkoutProgressRespData
    @num_athlete INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        wp.entry_num AS WorkoutProgressID,
        w.title AS WorkoutTitle,
        w.tags AS WorkoutTags,
        w.premium AS WorkoutPremium,
        pt.name AS WorkoutPTName,
        wp.date AS Date
    FROM
        workout_progress wp
    INNER JOIN
        workout w ON wp.num_workout = w.num_workout
    INNER JOIN
        personal_trainer pt ON w.PT_num = pt.num_PT
    WHERE
        wp.Athlete_num = @num_athlete
END
```

Fig.8 - Get Athlete WorkoutProgress & Respective Data

10. UDF

Implementamos uma UDF para obter o tempo total gasto num workout, por parte de um atleta. Relativamente ao Time Exercise o tempo destes é o próprio tempo, mas no caso dos Reps Exercise estes não têm tempo, mas sim reps (repetições)

```
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
```

Fig.8 - Get Workout Duration

11. Views

Criámos uma View para dar os três PTs mais baratos:

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

Fig.9 - Top 3 Cheapest PTs

12. Conclusão

Podemos concluir que os principais objetivos da aplicação foram cumpridos. O personal trainer consegue criar workouts e exercícios.
Os atletas conseguem comunicar com os PTs a que se subscreveram. Conseguem também ver os workouts exclusivos desses PTs e ver os workouts grátis que todos os outros publicaram.
Os atletas têm a possibilidade de registar o seu progresso em cada workout e conseguem consultá-lo sempre que quiserem.