Relatório

Armazenamento e Acesso a Dados

Aluno: João Pedro Pereira Rodrigues – Nº 20629
Aluno: Jorge Alexandre Pereira Rodrigues – Nº 20632
Aluno: Pedro Miguel Dias Martins – Nº20630
Aluno: Telmo Jorge Moura Paiva – Nº 20614

Professor: Hugo Fernando da Cunha Freitas

Curso Técnico Superior Profissional em Desenvolvimento Web e Multimédia

Vila Nova de Famalicão, 15 de Junho, 2021

Resumo

O presente relatório descreve o processo de criação de um sistema de base de dados referente a uma empresa do setor da construção, no qual serão abordadas o contexto do negócio, o modelo relacional, o Código SQL para construção da base de dados bem como as queries, vistas, funções e procedimentos.

Palavras-Chave:

Base de Dados;

PostgresSQL;

Query;

Construção;

Lista de Abreviaturas e Siglas

SQL - Structured Query Language

PK – Primary Key

FK – Foreign Key

PGSQL – PostGreSQL

Índice de Figuras

Índice

Introdução

No âmbito da unidade curricular de administração e acesso de dados, do curso de desenvolvimento web e multimédia, foi proposto a elaboração de um trabalho prático no qual consiste na criação de uma base de dados de um modelo de negócio à escolha pelo grupo.

Foi definido como primeiro objetivo o modelo de negócio a tratar na base de dados bem como os dados que seriam relevantes tratar nesta base de dados.

Após a definição destes pontos, foi elaborado o modelo relacional, no qual foram evidenciadas as relações entre as diversas entidades, assim como a cardinalidade e existência. Além disso foram detalhadas num diagrama os atributos incluindo as chaves primárias e estrangeiras.

Por fim será apresentado o código SQL para a construção da Base de dados, bem como as queries, vistas, funções e procedimentos que suportam a base de dados criada.

Contextualização

Motivação e objetivos

Este desafio passou por idealizar uma estrutura de base de dados para o setor que fosse apelativo ao grupo, mas também pelos desafios que essa escolha iria originar.

Estrutura do Documento

Este documento está estruturado pela introdução e pelo presente capítulo que tem como objetivos contextualizar o trabalho, isto é, explicar a motivação e a descrição do modelo de negócio.

No seguinte capítulo será apresentado o modelo relacional, o código para construção da base de dados bem como as quereis, vistas, funções e procedimentos que foram desenvolvidos.

Modelo de negócio

Uma empresa do ramo da Construção Civil, especializada em obras de moradias pretende implementar uma base de dados para ter maior organização dos seus projetos.

Pretende, portanto, um sistema que lhe permita saber a cada momento qual a evolução dos projetos, assim como a distribuição dos recursos humanos da sua empresa, nomeadamente através das equipas presentes nos projetos.

Esta empresa distribui a sua carteira de obras pelas diversas zonas do país, de Norte a Sul, tendo definido a distribuição dos mesmos pelos diversos distritos do País, sendo que cada projeto tem a si designado uma equipa de funcionários, sendo que estes possuem um cargo específico.

As equipas de funcionários integram diversas fases do projeto e estão alocadas a diferentes especialidades. Assim temos diferentes constituintes para as equipas e estas vão rodando entre os diversos projetos da empresa, consoante a especialidade e fase em que se encontram.

Foram também definidos os seguintes pressupostos a fim de contextualizar as escolhas para as relações entre entidades e criação de tabelas:

- Os projetos da empresa são todos semelhantes, ou seja, são sempre moradias.
- Todos os projetos têm as mesmas especialidades, nomeadamente:
  - Arquitetura
  - Fundação
  - Estrutura
  - Pichelaria
  - Eletricidade
- Todas as especialidades têm 3 Fases:
  - Arranque
  - Produção
  - Acabamentos
- As equipas estão fixadas especificamente para 1 fase e 1 especialidade, ou seja, apenas podem fazer a Fase "Arranque", Especialidade "Pichelaria", por exemplo.
- Existem assim 3 x 5 = 15 equipas, nomeadas de A a O que cobrem todas as especialidade e fases.
- Os funcionários podem pertencer a mais do que uma equipa, e o seu cargo é irrelevante para a equipa inserida.
- As tabelas "*Nome_Proprio*", "*Apelido*", "*localidade*" e "*cargo*" foram apenas criadas para efeitos de geração de dados aleatórios para o preenchimento dos dados de funcionários/*Users*. Podíamos ter criado FK para estas mas optamos por não o fazer.

Atividades Realizadas

Modelo Relacional

Tendo em conta o descritivo do modelo de negócio e respetivos pressupostos tomados, presentes no capítulo 0, foi elaborado o seguinte modelo relacional apresentado na Figura 1:

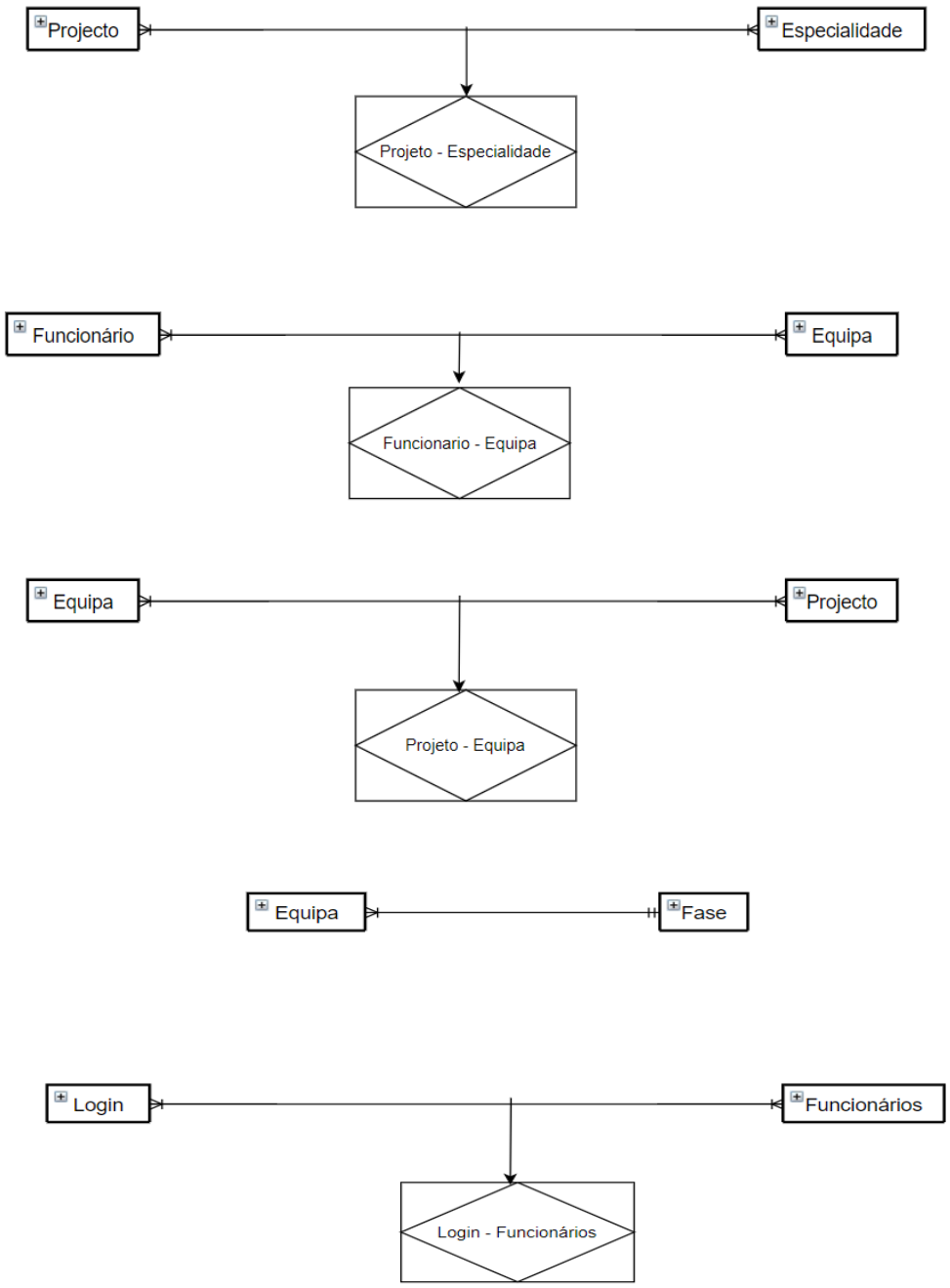

Figura 1 – Modelo relacional

Sucintamente, este modelo relacional descreve as relações existentes entre projeto e especialidade, funcionário e equipa, equipa e projeto, equipa e fase e login e funcionários.

Na seguinte Figura 2 são apresentadas as entidades para a base de dados:

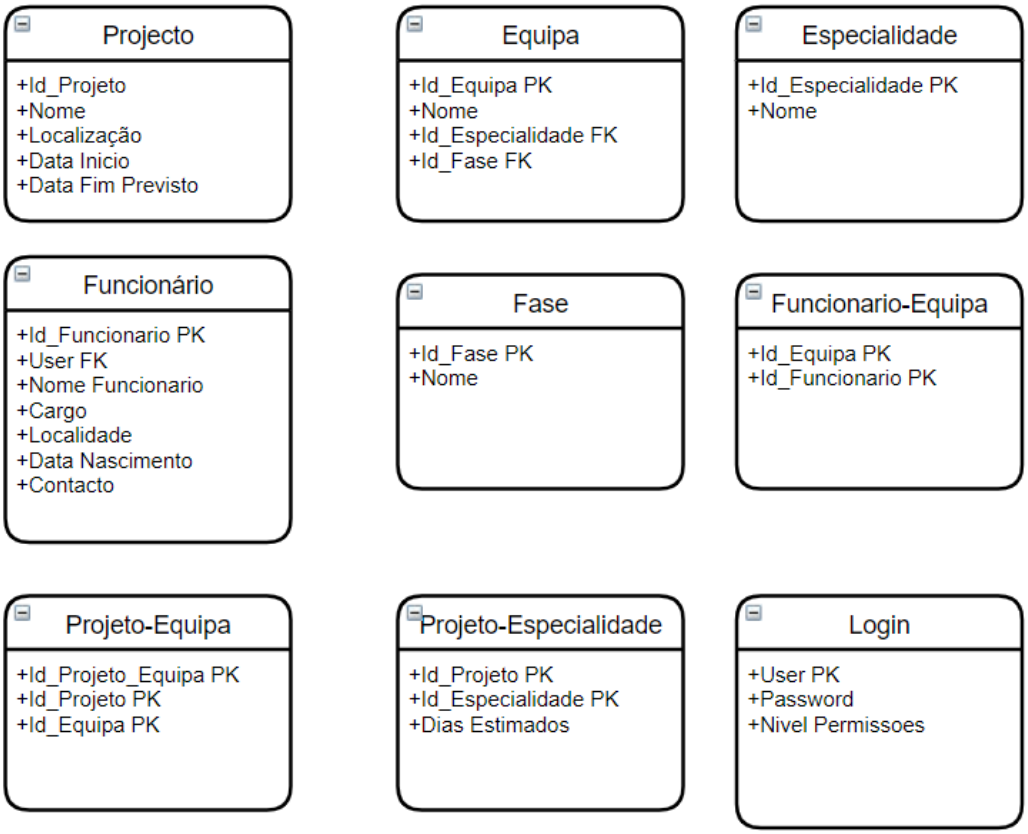

Figura 2 – Entidades

Código SQL para construção da base de dados

Tendo em conta a definição do modelo relacional e respetivas entidades indicadas no capítulo 0, procedeu-se a elaboração do código para a construção da base de dados.

Como tal seguiu-se a seguinte ordem de criação das tabelas:

- Login;
- Funcionários;
- Fase;
- Projetos;
- Especialidades;

- Equipas;
- Funcionário – Equipa;
- Projeto – Equipa;
- Projeto – Especialidade;
- Registo Diário;

Foram também criadas as seguintes tabelas para a implementação dos “procedures” de geração automática de funcionários/logins:

- Nome_proprio
- Apelido
- Localidade
- Cargo

No Anexo 1, é apresentado o código elaborado para a construção das tabelas da base de dados.

Seguidamente foram inseridos alguns dados nas várias tabelas para garantir o correto funcionamento das mesmas. No Anexo 2 é apresentada a inserção de dados efetuada para testar a base de dados criada, bem como testar as funções, procedimentos e vistas elaboradas e detalhadas nos capítulos 0 a 0.

Queries

Atendendo aos pressupostos indicados no capítulo 0, foram elaboradas as seguintes queries para atender algumas questões que seriam relevantes, tais como:

- Quantos dias foram usados na totalidade do projeto 1?
- Quantos dias foram usados para a especialidade de pichelaria no projeto 1 até ao momento?
- Nome dos funcionários com mais de 40 anos por ordem ascendente?
- Nome do funcionário e nome da equipa onde o funcionário é de Setúbal?
- Quais os dias estimados da especialidade arquitetura do projeto "moradia3"?
- Que funcionários trabalham na mesma zona do projeto?
- Diga o nome dos funcionários que trabalharam no dia 21/junho/2021?

Para atender a estas questões foram elaboradas as seguintes queries apresentadas na seguinte Figura 3 à Figura 9:

```
select count(1)
from registoDiario
where id_projeto = 1;
```

Figura 3 – Query para quantos dias foram usados na totalidade do projeto 1

```
-- 2. Quantos dias foram usados para a especialidade de pichelaria no projeto 1 até ao momento?
select count(1)
from registoDiario
inner join equipas e on e.id = registoDiario.id_equipa
inner join especialidade esp on e.id_especialidade = esp.id
where esp.nome = 'Pichelaria';
-- 2.Alternativa
select count(1)
from registoDiario r, equipas eq, especialidade esp
where r.id_equipa = eq.id
    and eq.id_especialidade = esp.id
    and esp.nome = 'Pichelaria';
```

Figura 4 -Query para quanto dias foram usados para a especialidade de pichelaria no projeto 1 até ao momento

```
select nome
from funcionarios
where (extract(year from now()) - extract(year from dataNascimento)) > 40
order by dataNascimento asc;
```

Figura 5 - Nome dos funcionários com mais de 40 anos por ordem ascendente

```
-- Nome do funcionário e nome da equipa onde o funcionário é de setubal:
select f.nome, e.nome
from funcionarios f, equipas e, funcionario_equipa fe
where f.id = fe.id_funcionario
  and e.id = fe.id_equipa
and f.localidade = 'Setubal';
```

Figura 6 - Nome do funcionário e nome da equipa onde o funcionário é de Setúbal

```
select diasEstimados
from projeto_especialidade
inner join especialidade e on e.id = projeto_especialidade.id_especialidade
inner join projetos p on p.id = projeto_especialidade.id_projeto
where p.nome='moradia3' and e.nome = 'Arquitetura';
```

Figura 7 - Quais os dias estimados da especialidade arquitetura do projeto "moradia3"

```
select  distinct funcionarios.nome, funcionarios.localidade,p.nome, p.localizacao
from funcionarios
inner join funcionario_equipa fe on funcionarios.id = fe.id_funcionario
inner join equipas e on e.id = fe.id_equipa
inner join projeto_equipa pe on e.id = pe.id_equipa
inner join projetos p on p.id = pe.id_projeto
where funcionarios.localidade = p.localizacao;
```

Figura 8 - Que funcionários trabalham na mesma zona do projeto

```
select  distinct funcionarios.nome
from funcionarios
inner join funcionario_equipa fe on funcionarios.id = fe.id_funcionario
inner join registodiario r on fe.id_equipa = r.id_equipa
where to_char(r.data, 'YYYY-MM-DD') = '2021-06-21';
```

Figura 9 - Diga o nome dos funcionários que trabalharam no dia 21/junho/2021

Funções

Atendendo aos pressupostos indicados no capítulo 0, foram elaboradas as seguintes funções com os objetivos de simplificar a forma de obtenção de alguns dados bem como apoiar à elaboração de procedimentos demonstrados no próximo capítulo 0:

- *calcularDiasTrabalhadosPorEspecialidade:* esta função retorna os dias trabalhos para uma especialidade específica.
- *calcPercentagemExecutadaProj*: esta função retorna a relação entre os dias executadas e os dias totais para um determinado projeto.
- *generateRandomPassword:* esta função permite gerar uma password
- *generateDataNacimento*: esta função permite gerar uma data de nascimento aleatória
- *generateRandomContacto*: esta função permite gerar um contacto aleatório
- *generateRandomNomeFuncionario*: esta função permite gerar um Nome aleatório (Nome Próprio + Apelido)
- *generateRandomCargoFuncionario*: esta função permite gerar um cargo aleatorio
- *generateRandomLocalidade*: esta função permite gerar uma localidade aleatorio
- *generateUsername*: esta função permite gerar um username aleatorio
- *generateDataRegisto*: esta função permite gerar uma data aleatorio entre dois intervalos

Para atender a estas questões referidas foram elaboradas as seguintes funções apresentadas na seguinte Figura 10 à

```
insert into registoDiario(id_projeto, id_equipa, data)
values(id_proj,equipa,generatedataregisto( inicioIntervalo: inicioIntervalo, fimIntervalo: fimIntervalo));
```

Figura 31:

```
insert into registoDiario(id_projeto, id_equipa, data)
values(id_proj,equipa,generatedataregisto( inicioIntervalo: inicioIntervalo, fimIntervalo: fimIntervalo));
```

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
       generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

```
CREATE or replace FUNCTION calcularDiasTrabalhadosPorEspecialidade (especialidadeNome varchar, id_proj integer)
RETURNS integer
LANGUAGE plpgsql
AS
$$
DECLARE
    counter integer;
BEGIN
    select count(1)
        into counter
    from registoDiario r, equipas eq, especialidade esp
    where r.id_equipa = eq.id
        and eq.id_especialidade = esp.id
        and esp.nome = especialidadeNome
        and r.id_projeto = id_proj;
    return counter;
END;
$$;
```

Figura 10 – Função calcularDiasTrabalhadosPorEspecialidade

```
--Uso da função de calcular o numero de Dias Trabalhados por especialidade num determinado projeto (exemplo: pichelaria no projeto 2)
select distinct registoDiario.id_projeto, calcularDiasTrabalhadosPorEspecialidade( especialidadeNome: 'Pichelaria', id_proj: 2)
from registodiario
where id_projeto = 1;
```

Figura 11 – Exemplificação do uso da função calcularDiasTrabalhadosPorEspecialidade

```
CREATE or replace FUNCTION calcPercentagemExecutadaProj (idProjeto integer)
RETURNS numeric (4,1)
LANGUAGE plpgsql
AS
$$
DECLARE
    result numeric (4,1) ;
BEGIN

select
    (select count(1)
    from registoDiario rd
    where rd.id_projeto = idProjeto)/
(   Select extract(epoch from ((p.dataFimPrevisto - p.dataInicio)/3600)/24)+1
    from projetos p
    where p.id = idProjeto)*100
        into result;

    return result;
END;
$$;
```

Figura 12 – Função calcPercentagemExecutadaProj

```
--Teste do cálculo da percentagem Executada num determinado projeto (exemplo: projeto 1):

select calcpercentagemexecutadaproj( idProjeto: 1);
```

Figura 13 – Exemplificação do uso da função calcPercentagemExecutadaProj

```
CREATE FUNCTION generateRandomPassword()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
   j int4;
   result text;
   allowed text;
   allowed_len int4;
BEGIN
   allowed := '23456789abcdefghjkmnpqrstuvwxyzABCDEFGHJKMNPQRSTUVWXYZ&#%@';
   allowed_len := length(allowed);
   result := '';
   WHILE length(result) < 8 LOOP
      j := int4(random() * allowed_len);
      result := result || substr(allowed, j+1, 1);
   END LOOP;
   RETURN result;
END;
$$;
```

Figura 14 – Função generateRandomPassword

```
insert into login (userName, password, nivelPermissoes)
values(nomeUser,generateRandomPassword(),nivelPermissao);
```

Figura 15 – Exemplificação do uso da função generateRandomPassword

```
CREATE FUNCTION generateDataNacimento()
RETURNS timestamp
LANGUAGE plpgsql
    AS $$
DECLARE
    dataNascimento timestamp;
BEGIN
   select timestamp '1965-01-01 20:00:00' +
       random() * (timestamp '2003-01-01 20:00:00'-timestamp '1965-12-31 10:00:00')
    into dataNascimento;
   RETURN dataNascimento ;
END;
$$;
```

Figura 16 – Função generateDataNascimento

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
       generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

Figura 17 – Exemplificação do uso da função generateDataNascimento

```
CREATE or replace FUNCTION generateRandomContacto()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
   j int4;
   result text;
   allowed text;
   allowed_len int4;
BEGIN
   allowed := '123456789';
   allowed_len := length(allowed);
   result := '91';
   WHILE length(result) < 9 LOOP
      j := int4(random() * allowed_len);
      result := result || substr(allowed, j+1, 1);
   END LOOP;
   RETURN result;
END;
$$;
```

Figura 18 – Função generateRandomContacto

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
       generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

Figura 19 – Exemplificação do uso da função generateRandomContacto

```
    CREATE or replace FUNCTION generateRandomNomeFuncionario()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
    i int4;
    j int4;
    result text;
BEGIN
    i := int4(random() * 19)+1;
    j := int4(random() * 19)+1;

  result := concat(
      (select nome from nome_Proprio where id = i),
      ' ',
      (select nome from apelido where id = j)
                );
   RETURN result;
END;
$$;
```

Figura 20 – Função generateRandomNomeFuncionario

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
      generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

Figura 21 – Exemplificação do uso da função generateRandomNomeFuncionario

```
CREATE or replace FUNCTION generateRandomCargoFuncionario()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
    i int4;
    result text;
BEGIN
    i := int4(random() * 9)+1;
    result := (select nome from cargo where id = i);
    RETURN result;
END;
$$;
```

Figura 22 – Função generateRandomCargoFuncionario

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
      generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

Figura 23 – Exemplificação do uso da função generateRandomCargoFuncionario

```
CREATE or replace FUNCTION generateRandomLocalidade()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
    i int4;
    result text;
BEGIN
    i := int4(random() * 17)+1;
    result := (select nome from localidade where id = i);
    RETURN result;
END;
$$;
```

Figura 24 – Função generateRandomLocalidade

```
insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
       generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
i := i + 1;
```

Figura 25 – Exemplificação do uso da função generateRandomLocalidade

```
CREATE or replace FUNCTION countFuncionariosNaEquipa(func_id int, equipa_id int)
returns int
language plpgsql
    as $$
    DECLARE
        counter int;
    BEGIN
        select  count(1)
            from funcionario_equipa fe
            join equipas e on fe.id_equipa = e.id and e.id = equipa_id
                where fe.id_funcionario = func_id
        into counter;
        return counter;
    END;
    $$;
```

Figura 26 – Função countFuncionariosNaEquipa

```
if
    (select countFuncionariosNaEquipa( func_id: i,  equipa_id: equipa_record.id) = 0)
```

Figura 27 – Exemplificação do uso da função countFuncionariosNaEquipa

```
CREATE or replace FUNCTION generateUsername()
RETURNS text
LANGUAGE plpgsql
    AS $$
DECLARE
    i int4; --
    original_username text;
    new_number int4;
    result text;
BEGIN
    i := (select count(1) from login);
    select regexp_replace((select (userName) from login limit 1 offset (i-1)), '\D','','g') into original_username;
    select cast(original_username as int4)+1 into new_number;
    select concat('ROMAR',trim(to_char(new_number,'99999'))) into result;
    RETURN result;
END;
$$;
```

Figura 28 – Função generateUsername

```
new_username := generateusername();
```

Figura 29 – Exemplificação do uso da função generateUsername

```
CREATE FUNCTION generateDataRegisto(inicioIntervalo text, fimIntervalo text)
RETURNS timestamp
LANGUAGE plpgsql
AS
$$
DECLARE
    dataRegisto timestamp;
BEGIN
   select inicioIntervalo:: timestamp +
        random() * (fimIntervalo::timestamp - inicioIntervalo::timestamp)
        into dataRegisto;
   RETURN dataRegisto;
END;
$$;
```

Figura 30 – Função generateDataRegisto

```
insert into registoDiario(id_projeto, id_equipa, data)
values(id_proj,equipa,generatedataregisto( inicioIntervalo: inicioIntervalo, fimIntervalo: fimIntervalo));
```

Figura 31 – Exemplificação do uso da função generateDataRegisto

Procedimentos

Os procedimentos elaborados nesta base de dados foi o intuito de facilitar a inserção de dados para a população de registos nas tabelas. Não têm qualquer retorno. Apenas executam o código que está dentro deles inserido.

Assim foram criados os seguintes procedimentos:

- *gerarUsuario*: Este procedimento permite gerar um nome de Usuario sequencial para a tabela de Login, apartir do tipo "Romar*xxx*", em que o *xxx* são numeros inteiros sequenciais, gerados automaticamente do último registo na tabela Login.
- *gerarFuncionario*: Este procedimento permite gerar um funcionário novo aleatório, usando as funções anteriormente criadas para geração dos diversos atributos assim como executar o procedimento *gerarUsuario* para ter uma referência para a FK username.
- *gerarEquipaFuncionario*: Este procedimento permite gerar combinações para a tabela Funcionário_Equipa, atribuindo funcionarios aleatórios a cada equipa, mantendo o racio 1/3/2, para as fases Arranque/Produção/Acabamentos.

- *gerarRegistosDiarios*:Este procedimento permite gerar registos aleatórios para um determinado projeto entre duas datas escolhidas.

Para atender a estas questões referidas foram elaborados os seguintes procedimentos apresentados na seguinte Figura 32 a

```
call gerarRegistosDiarios( id_proj: 2, inicioIntervalo: '2021-06-01', fimIntervalo: '2021-06-30', nRegistos: 15);
call gerarRegistosDiarios( id_proj: 1, inicioIntervalo: '2021-06-01', fimIntervalo: '2021-06-30', nRegistos: 20);
call gerarRegistosDiarios( id_proj: 3, inicioIntervalo: '2021-09-01', fimIntervalo: '2021-09-30', nRegistos: 10);
call gerarRegistosDiarios( id_proj: 4, inicioIntervalo: '2021-10-01', fimIntervalo: '2021-10-30', nRegistos: 33);
```

Figura 39

```
call gerarFuncionario( numeroDeFuncionarios: 20);
```

Figura 35:

```
CREATE OR REPLACE PROCEDURE gerarUsuario(nomeUser varchar)
LANGUAGE plpgsql
AS
$$
DECLARE
    nivelPermissao integer;
BEGIN
    nivelPermissao = floor(random() * (5-1+1) + 1)::int;
    insert into login (userName, password, nivelPermissoes)
    values(nomeUser,generateRandomPassword(),nivelPermissao);
END;
$$;
```

Figura 32 – Procedimento para gerarUsuario

```
new_username := generate_username();
call gerarUsuario( nomeUser: new_username);
```

Figura 33 – Exemplificação do uso do procedimento para gerarUsuario

```
CREATE OR REPLACE PROCEDURE gerarFuncionario(numeroDeFuncionarios int)
LANGUAGE plpgsql
AS
$$
DECLARE
    i integer := 0;
    new_username text;
BEGIN
    loop
        exit when i = numeroDeFuncionarios;
        new_username := generateusername();
        call gerarUsuario( nomeUser: new_username);
        insert into funcionarios (userName, nome, cargo, localidade, dataNascimento, contacto)
        values(new_username,generateRandomNomeFuncionario(), generaterandomcargofuncionario(),
               generaterandomlocalidade(), generatedatanacimento(),generaterandomcontacto());
        i := i + 1;
    end loop;
END;
$$;
```

Figura 34 – Procedimento para gerarFuncionario

```
call gerarFuncionario( numeroDeFuncionarios: 20);
```

Figura 35 – Exemplificação do uso do procedimento para gerarFuncionario

```
CREATE OR REPLACE PROCEDURE gerarFuncionarioEquipa()
LANGUAGE plpgsql
AS
$$
DECLARE
    equipa_record record;
    i integer := 0;
    j integer := 0;
    REPETIDO bool = TRUE;
BEGIN
    for equipa_record in SELECT * FROM equipas
    loop
        IF equipa_record.id_fase = 1
            then
                i := int4(random() * (select count(1)-1 from funcionarios)+1);
                insert into funcionario_equipa(id_equipa, id_funcionario)
                values (equipa_record.id, (select f.id from funcionarios f where f.id = i));
        elsif equipa_record.id_fase = 2
            then
            loop
                exit when j = 3;
                loop
                    EXIT WHEN REPETIDO = FALSE;
                    i := int4(random() * (select count(1)-1 from funcionarios)+1);
                    if
                        (select countFuncionariosNaEquipa( func_id: i,  equipa_id: equipa_record.id) = 0)
                    then
                        insert into funcionario_equipa(id_equipa, id_funcionario)
                        values (equipa_record.id, (select f.id from funcionarios f where f.id = i));
                        REPETIDO = FALSE;
                    end if;
                end loop;
                j := j +1;
            end loop;
        else
            loop
                exit when j = 2;
                loop
                    EXIT WHEN REPETIDO = FALSE;
                    i := int4(random() * (select count(1)-1 from funcionarios)+1);
                    if
                        (select countFuncionariosNaEquipa( func_id: i,  equipa_id: equipa_record.id) = 0)
                    then
                        insert into funcionario_equipa(id_equipa, id_funcionario)
                        values (equipa_record.id, (select f.id from funcionarios f where f.id = i));
                        REPETIDO = FALSE;
                    end if;
                end loop;
                j := j +1;
            end loop;
        end if;
        j := 0;
    end loop;

END;
$$;
```

Figura 36 – Procedimento para gerarFuncionarioEquipa

```
call gerarFuncionarioEquipa();
```

Figura 37 – Exemplificação do uso do procedimento para gerarFuncionarioEquipa

```
CREATE OR REPLACE PROCEDURE gerarRegistosDiarios(id_proj integer, inicioIntervalo text,
                            fimIntervalo text, nRegistos integer)
LANGUAGE plpgsql
AS
$$
DECLARE
    equipa int4 := 0;
    i int4 = 0;
BEGIN
    if inicioIntervalo::timestamp > fimIntervalo::timestamp
    then i = nRegistos;
    end if;
    loop
        exit when i = nRegistos;
        equipa := int4(random() * 14)+1;
        insert into registoDiario(id_projeto, id_equipa, data)
        values(id_proj,equipa,generatedataregisto( inicioIntervalo: inicioIntervalo, fimIntervalo: fimIntervalo));
        i := i + 1;
    end loop;
END;
$$;
```

Figura 38 – Procedimento para gerarRegistosDiarios

```
call gerarRegistosDiarios( id_proj: 2, inicioIntervalo: '2021-06-01', fimIntervalo: '2021-06-30', nRegistos: 15);
call gerarRegistosDiarios( id_proj: 1, inicioIntervalo: '2021-06-01', fimIntervalo: '2021-06-30', nRegistos: 20);
call gerarRegistosDiarios( id_proj: 3, inicioIntervalo: '2021-09-01', fimIntervalo: '2021-09-30', nRegistos: 10);
call gerarRegistosDiarios( id_proj: 4, inicioIntervalo: '2021-10-01', fimIntervalo: '2021-10-30', nRegistos: 33);
```

Figura 39 – Exemplificação do uso do procedimento para gerarRegistosDiarios

Vistas

As vistas elaboradas têm como objetivo demonstrar os dados que sejam mais revelantes de consultar com elevada frequência, de acordo como os pressupostos indicados no capítulo 0.

Para atender a estas questões referidas foram elaboradas as seguintes vistas apresentados na seguinte Figura 40 a Figura 41:

```
CREATE OR REPLACE VIEW vista_PercentagemExecutada AS
    select nome, calcpercentagemexecutadaproj( idProjeto: p.id) as percentagemExecutada
    from projetos p;
```

Figura 40 – Vista com percentagem geral executada por projeto

```
select nome,percentagemExecutada
from vista_PercentagemExecutada vpe
where vpe.percentagemExecutada < 20;
```

Figura 41 – Exemplificação do uso da vista com percentagem geral executada por projeto

```
create or replace  view vista_DiasEstimados_Esp_Projetos
     as
select p.id as ID_Projeto,
       pe.diasEstimados as Pichelaria_estimados,
        pe2.diasEstimados as Eletricidade_estimados,
       pe3.diasEstimados as Fundacao_Estimados,
       pe4.diasEstimados as Estrutura_Estimados,
       pe5.diasEstimados as Arquitetura_Estimados

from projetos p
       join projeto_especialidade pe on p.id = pe.id_projeto
         join especialidade e
           on e.id = pe.id_especialidade and e.nome = 'Pichelaria'::text
        join projeto_especialidade pe2 on p.id = pe2.id_projeto
         join especialidade e2
           on e2.id = pe2.id_especialidade and e2.nome = 'Eletricidade'::text
       join projeto_especialidade pe3 on p.id = pe3.id_projeto
         join especialidade e3
           on e3.id = pe3.id_especialidade and e3.nome = 'Fundação'::text
       join projeto_especialidade pe4 on p.id = pe4.id_projeto
         join especialidade e4
           on e4.id = pe4.id_especialidade and e4.nome = 'Estrutura'::text
       join projeto_especialidade pe5 on p.id = pe5.id_projeto
         join especialidade e5
           on e5.id = pe5.id_especialidade and e5.nome = 'Arquitetura'::text;
```

Figura 42 – Vista com dias estimados por especialiadde por projeto

```
select * from vista_DiasEstimados_Esp_Projetos
where Pichelaria_estimados > 5;
```

Figura 43 – Exemplificação do uso da vista com percentagem geral executada por projeto

Conclusão

Lições aprendidas

Na elaboração deste trabalho prático permitiu aplicar os conceitos teóricos lecionados para se poder a estruturar uma base de dados recorrendo ao modelo relacional. Após a definição deste modelo e definição das entidades, compreende-se a sua importância para uma correta estruturação das tabelas a criar com a linguagem SQL.

Este trabalho permitiu assim abordar os vários temas relacionados com a criação e manipulação de bases de dados, isto é, a criação das tabelas, a criação de queries, funções, procedimentos e vistas.

Apreciação final

No ponto de vista global, o grupo chegou a uma apreciação global bastante positiva, pois este trabalho prático permitiu a aplicação dos conceitos lecionados na unidade curricular de armazenamento e acesso a dados num modelo de negócio do interesse do grupo.

Como melhorias futuras relativamente a base de dados apresentada temos as seguintes:

- Fixar os atributos de localidade e cargo dos funcionários, projeto à tabela de Localidade/cargo. Não foi feita a referência através de FK por pressuposto.
- Criar *View* que permita uma perspetiva sobre todos os projetos e a diferença entre os dias trabalhos e os dias estimados.
- Inclusão de tabelas de registos para controlo de materiais e equipamento.

Apêndices

Anexo 1 – Código de criação das tabelas

```
-- Criação a tabela de Login
CREATE TABLE if not exists login(
    userName varchar(20) primary key,
    password varchar(20),
    nivelPermissoes smallint
);
```


```
-- Criação a tabela de Login
CREATE TABLE if not exists login(
    userName varchar(20) primary key,
    password varchar(20),
    nivelPermissoes smallint
);
```


```
-- Criação da tabela da fase:
CREATE table if not exists fase(
 id serial primary key,
 nome varchar(20));
```

```
-- Criação da tabela de projetos:
CREATE table if not exists projetos(
id serial primary key,
nome varchar(30),
 localizacao varchar(20),
 dataInicio timestamp,
 dataFimPrevisto timestamp);
```

```
--Criação da tabela de especialidade:
CREATE table if not exists especialidade(
id serial primary key,
 nome varchar(20));
```

```
-- Criação da tabela de equipas:
CREATE table if not exists equipas(
 id serial primary key,
 nome varchar(20),
 id_especialidade integer,
 id_fase integer,
 FOREIGN KEY(id_especialidade) REFERENCES especialidade(id),
 FOREIGN KEY(id_fase) REFERENCES fase(id));
```

```
-- Criação da tabela de funcionário-equipa:
CREATE table if not exists funcionario_equipa(
id_equipa integer,
id_funcionario integer,
primary key (id_equipa,id_funcionario),
foreign key(id_equipa) references equipas(id),
foreign key (id_funcionario) references funcionarios(id));
```

```
--Criação da tabela de projeto-equipa:
CREATE table if not exists projeto_equipa(
id serial primary key,
id_projeto integer,
id_equipa integer,
foreign key(id_projeto) references projetos(id),
foreign key (id_equipa) references equipas(id));
```

```
--Criação da tabela de projeto-especialidade:
CREATE table if not exists projeto_especialidade(
id serial primary key,
id_projeto integer,
id_especialidade integer,
diasEstimados smallint,
foreign key(id_projeto) references projetos(id),
foreign key (id_especialidade) references especialidade(id) );
```

```
--Criação da tabela de Registo diário:
create table if not exists registoDiario(
id serial primary key,
id_projeto integer,
id_equipa integer,
data date,
foreign key (id_projeto) references projetos(id),
foreign key (id_equipa) references equipas(id));
```

Anexo 2 – Código para inserção de dados nas tabelas

```
-- criar login
insert into login (userName, password, nivelPermissoes)
values('joao','joao123',3);
 insert into login (userName, password, nivelPermissoes)
values('jorge','jorge123',3);
 insert into login (userName, password, nivelPermissoes)
values('pedro','pedro123',3);
```

```
-- Criar funcionarios
insert into funcionarios(userName, nome, cargo, localidade,
dataNascimento, contacto) values('joao','João','trolha','Famalicão',
'1991-05-29','911111111');
insert into funcionarios(userName, nome, cargo, localidade,
dataNascimento, contacto)
values('jorge','Jorge','picheleiro','Barcelos', '1979-07-
08','92222222');
insert into funcionarios(userName, nome, cargo, localidade,
dataNascimento, contacto) values('pedro','Pedro','gruista','Povoa ',
'1986-02-21','933333333');
```

```
-- Criar especialidade
insert into especialidade(nome) values ('Pichelaria');
insert into especialidade(nome) values ('Eletricidade');
```

```
-- Criar faseinsert into fase(nome) values('Arranque');
```

```
-- Criar equipas
insert into equipas(nome, id_especialidade, id_fase) values('A',1,1);
insert into equipas(nome, id_especialidade, id_fase) values('B',2,1);
```

```
-- Criar projeto
insert into projetos(nome, localizacao, dataInicio, dataFimPrevisto)
values('moradia1','Braga','2021-06-01','2021-06-30');
insert into projetos(nome, localizacao, dataInicio, dataFimPrevisto)
values('moradia2','Barcelos','2021-06-01','2021-06-30');
```

```
--Criar registos diários
insert into registoDiario (id_projeto, id_equipa, data)
values(1,1,'2021-06-10');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,1,'2021-06-11');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,1,'2021-06-12');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,1,'2021-06-13');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,1,'2021-06-14');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,3,'2021-06-10');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,3,'2021-06-11');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,3,'2021-06-12');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,3,'2021-06-13');
insert into registoDiario (id_projeto, id_equipa, data)
values(1,3,'2021-06-14');
insert into registoDiario (id_projeto, id_equipa, data)
values(2,1,'2021-06-10');
insert into registoDiario (id_projeto, id_equipa, data)
values(2,1,'2021-06-11');
insert into registoDiario (id_projeto, id_equipa, data)
values(2,1,'2021-06-12');
```

```
--GERAR DADOS A PARTIR DE PARTIR DE PROCEDIMENTOS:
--Gerar funcionários:
call gerarfuncionario(20);
```

```
--Gerar relação equipa-funcionário:
call gerarFuncionarioEquipa();

--Gerar registosDiários aleatórios para um determinado projeto:
call gerarRegistosDiarios(2,'2021-06-01','2021-06-30',15);
call gerarRegistosDiarios(1,'2021-06-01','2021-06-30',20);
call gerarRegistosDiarios(3,'2021-09-01','2021-09-30',10);
call gerarRegistosDiarios(4,'2021-10-01','2021-10-30',33);
```

Bibliografia

Generate random password. Disponível em: < https://www.postgresql.org/message-id/46685BBD.3070809%40hagander.net>. Acesso em: 11 Junho. 2021.

Random Function. Disponível em: < https://www.techonthenet.com/postgresql/functions/random.php > Acesso em: 11 Junho. 2021

PostgreSQL Create Procedure. Disponível em:< https://www.postgresqltutorial.com/postgresql-create-procedure/ >. Acesso em: 12 Junho 2021.