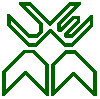
**Universidade Eduardo Mondlane**

**Faculdade de Ciências**

Departamento de Matemática e Informática

Trabalho Semestral - **BD II**

**Kliniek – Sistema de Gestão de Processos Clinicos**

Curso: Informática - 3˚Ano

**Discentes:**

* **Carlos Alberto Massavanhane Junior**
* **Paulo Amosse Cuambe**

**O Docente:**

dr. Jordão Uache

**Maputo, Novembro de 2019**

**Índice:**

[I. Contextualização 1](#_Toc24955402)

[1.1. Introdução 1](#_Toc24955403)

[1.2. Descrição do problema 1](#_Toc24955404)

[II. Descrição da solução proposta 2](#_Toc24955405)

[III. Material e Métodos 2](#_Toc24955406)

[IV. Desenvolvimento do sistema 3](#_Toc24955407)

[4.1. Diagrama de Entidade e Relacionamento 3](#_Toc24955408)

[4.2. Scripts de criação de tabelas e/ou Views 3](#_Toc24955409)

[*4.3.* Scripts de criação de *Triggers* 8](#_Toc24955410)

[4.4. Scripts de criação de Funções 9](#_Toc24955411)

[4.5. Scripts de criação de Procedimentos 15](#_Toc24955412)

[4.6. Diversos 20](#_Toc24955413)

[4.6.1. Sequências 20](#_Toc24955414)

[4.6.2. Backups 22](#_Toc24955415)

[V. Referências Bibliográficas 23](#_Toc24955416)

# Contextualização

## Introdução

O Antendimento Integrado de Criança e Adoelescente é um consutório médico que se dedica ao atendimento de crianças, adoelescentes, jovens e adultos nas áreas de pediatria, genecologia, psicologia entre outros. Cada paciente possuí um processo clinico que é aberto a quão da marcação da sua primeira consulta. Cada médico atende a uma especialidade. Para além das consultas, o AICA oferece serviços de exame. Tanto para as consultas assim como para os exames, os serviços podem ser normal, urgente ou especial. O atendimento dos serviços supracitados é realizado por um recepcionista.

## Descrição do problema

Actualmente todos os processos de gestão são efectuados manualmente, o que torna o atendimento bem como a própria gestão á nível interno moroso e constragedor.

Um dos principais no AICA é a gestão dos processos clinicos de seus pacientes pois actualmente os processos são armazenados num armário fisíco e cada vez que o paciente vai realizar um serviço (consulta ou exame) o funcionário de serviço (recepcionista) leva bastabte tempo a procura do mesmo processo o que faz com que os pacientes fiquem bastante tempo em espera.

Ouro constragimento enfrentado pelo AICA é na gestão das dispoibilidades dos médicos bem como na gestão dos serviços por eles realizados num determinado periodo, de forma a saber-se qual será o seu rendimento.

# Descrição da solução proposta

O Sistema Gestão de Processos Clinicos-Registo Médico Electrónico para o Atendimento Integral da Criança e Adolescentes - AICA a ser desenvolvido pela PaCaJo Lda., tem como objectivo automatizar os processos, disponibilizar e reduzir os custos de tempo de acesso e fornecimento de informação, criar condições para a gestão de conhecimento em tempo real sobre os diversos processos internos do AICA.

Importa referir que, para além do objectivo acima descrito o sistema visa ainda eliminar o uso de interfaces manuais, optimizar o fluxo da informação e a qualidade da mesma dentro da organização de forma eficiente, optimizar o processo de tomada de decisão, eliminar a redundância de actividades, reduzir o tempo de resposta, reduzir as incertezas, incorporação de melhores práticas aos processos internos e garantir maior aproveitamento do tempo nos processos de gestão.

Serviu como ponto de partida a necessidade de estabelecer a comunicação entre os colaboradores e seus departamentos, transparência nos processos de tomada de decisão, assim como prover uma ferramenta de gestão de informação e de fácil acesso em tempo real.

# Material e Métodos

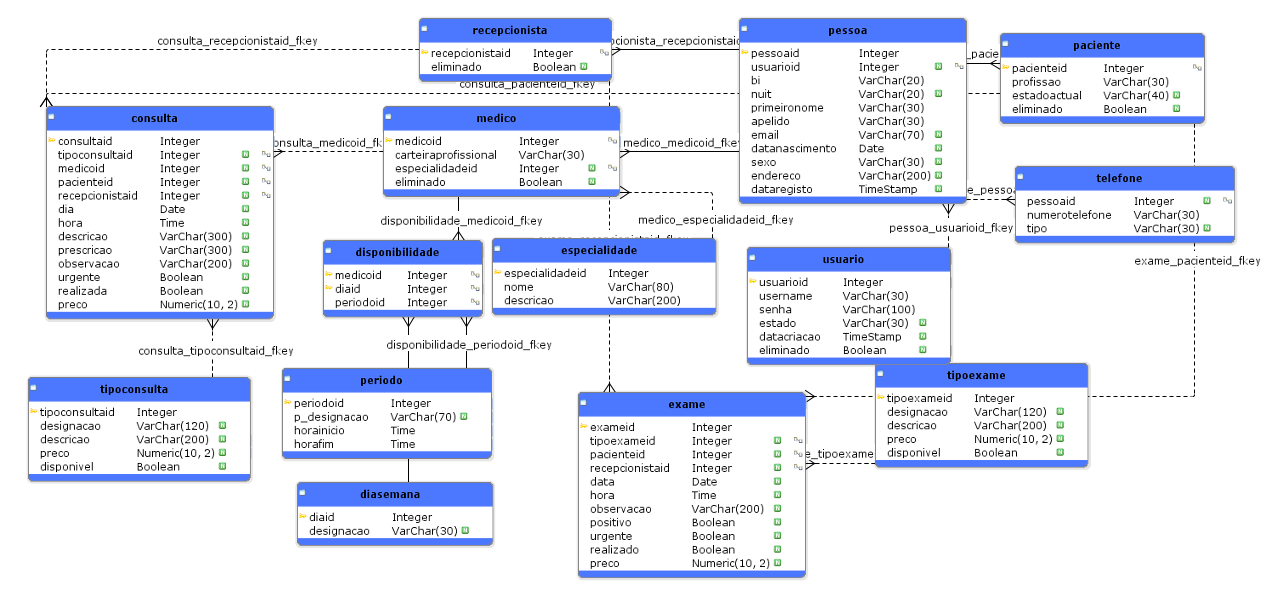
Para elaboração deste projecto, foram realizadas pesquisas sobre a documentação do Sistema de Gestão de Base de Dados (SGBD) em referências electrónicas. Foram ainda usados os slides apresentados durante as aulas de Base de Dados (um e dois).

Em termos de tecnologias usadas, foi usado o SGDB foi o PostgreSQL, para a geração do Diagrama de Entidade Relacionamento foi usado o DBWrench e para a manipulação da base de dados foi usado o PGAdmin, o SublimeText e o JetBrainsDataGrid.

O método usado para a realização do trabalho foi o similar ao *XtremeProgramming(*Programação Extrema) que consistiu particularmente no uso da técnica de desenvolvimento par a par por parte dos integrantes do projecto.

# Desenvolvimento do sistema

## Diagrama de Entidade e Relacionamento



## Scripts de criação de tabelas e/ou Views

*-- Criando Tabelas*

**create** **table** usuario(

usuarioid serial **primary** **key**,

username varchar(30) **unique** **not** **null**,

senha varchar(100) **not** **null**,

estado varchar(30) **Default** 'activo',

dataCriacao **Timestamp** **DEFAULT** (**current\_timestamp** **AT** TIME **ZONE** 'UTC'),

eliminado boolean **default** **false**

);

**create** **table** pessoa (

pessoaid serial **primary** **key**,

usuarioid int **references** usuario,

bi varchar(20) **unique** **not** **null**,

nuit varchar(20) **unique**,

primeiroNome varchar(30) **not** **null**,

apelido varchar(30) **not** **null**,

email varchar(70) **unique**,

dataNascimento Date,

sexo varchar(30),

endereco varchar(200),

dataRegisto **Timestamp** **DEFAULT** (**current\_timestamp** **AT** TIME **ZONE** 'UTC')

);

**create** **table** telefone(

pessoaid int **references** pessoa(pessoaid) **on** **delete** **cascade**,

numeroTelefone varchar(30) **not** **null**,

tipo varchar(30) **default** 'pessoal'

);

**create** **table** paciente(

pacienteid int **references** pessoa(pessoaid) **primary** **key** ,

profissao varchar(30) **not** **null**,

estadoActual varchar(40),

eliminado boolean **default** **false**

);

**create** **table** recepcionista(

recepcionistaid int **references** pessoa(pessoaid) **primary** **key**,

eliminado boolean **default** **false**

);

**create** **table** medico(

medicoid int **references** pessoa(pessoaid) **primary** **key**,

carteiraProfissional varchar(30) **unique** **not** **null**,

especialidadeid int **references** especialidade **on** **delete** **cascade**,

eliminado boolean **default** **false**

);

**create** **table** especialidade(

especialidadeid serial **primary** **key**,

nome varchar(80) **not** **null** **unique**,

descricao varchar(200) **not** **null**

);

*--*

**insert** **into** especialidade (nome, descricao)

**values** ('Pediatria', 'Cuida de crancas.'),

('Genecologista', 'Cuida do sistema reprodutor feminino.'),

('Psicologo', 'Saude mental.'), ('Generalista', 'Cuida das doencas do povo');

*--Gestao de Horarios*

**create** **table** diaSemana(

diaid serial **primary** **key**,

designacao varchar(30)

);

**insert** **into** diaSemana (designacao) **values**

('domingo'), ('segunda-feira'), ('terca-feira'), ('quarta-feira'),

('quinta-feira'), ('sexta-feira'), ('sabado');

**create** **table** periodo(

periodoid serial **primary** **key**,

p\_designacao varchar(70),

horainicio Time **not** **null**,

horafim Time **not** **null**

);

**insert** **into** periodo (p\_designacao, horainicio, horafim) **values**

('normal', '07:00', '15:30'),

('noite', '15:00', '23:30'), ('plantao', '23:00', '07:30');

**create** **table** disponibilidade (

medicoid int **references** medico **not** **null**,

diaid int **references** diaSemana **not** **null**,

periodoid int **references** periodo **not** **null**,

**Primary** **key** (medicoid, diaid)

);

*-- Marcacao de consultas*

**create** **table** tipoConsulta(

tipoconsultaid serial **primary** **key**,

designacao varchar(120),

descricao varchar(200),

preco numeric(10,2),

disponivel boolean **default** **true**

);

**insert** **into** tipoConsulta (designacao, descricao, preco) **values**

('Normal', 'Marcada no intervalo das 7 as 12', 1200),

('Urgente', 'Marcada fora do horario normal', 1500),

('Especial', 'Marcada no instante da consulta.', 2000);

**create** **table** tipoExame(

tipoexameid serial **primary** **key**,

designacao varchar(120),

descricao varchar(200),

preco numeric(10,2),

disponivel boolean **default** **true**

);

**insert** **into** tipoExame (designacao, descricao, preco) **values**

('Normal', 'Marcada no intervalo das 7 as 12', 1200),

('Urgente', 'Marcada fora do horario normal', 1500),

('Especial', 'Marcada no instante da consulta.', 2000);

**CREATE** **TABLE** exame

(

exameid integer **NOT** **NULL**,

tipoexameid integer,

pacienteid integer,

recepcionistaid integer,

enfermeiroid integer,

**data** date,

hora time **without** time **zone**,

observacao character varying(200) **COLLATE** pg\_catalog."default",

positivo boolean **DEFAULT** **false**,

urgente boolean **DEFAULT** **false**,

realizado boolean **DEFAULT** **false**,

preco numeric(10,2),

**CONSTRAINT** exame\_pkey **PRIMARY** **KEY** (exameid),

**CONSTRAINT** exame\_data\_hora\_key **UNIQUE** (**data**, hora)

,

**CONSTRAINT** exame\_pacienteid\_fkey **FOREIGN** **KEY** (pacienteid)

**REFERENCES** **public**.paciente (pacienteid) **MATCH** **SIMPLE**

**ON** **UPDATE** **NO** ACTION

**ON** **DELETE** **NO** ACTION,

**CONSTRAINT** exame\_recepcionistaid\_fkey **FOREIGN** **KEY** (recepcionistaid)

**REFERENCES** **public**.recepcionista (recepcionistaid) **MATCH** **SIMPLE**

**ON** **UPDATE** **NO** ACTION

**ON** **DELETE** **NO** ACTION,

**CONSTRAINT** enfermeiro\_enfermeiroid\_fkey **FOREIGN** **KEY** (enfermeiroid)

**REFERENCES** **public**.enfermeiro (enfermeiroid) **MATCH** **SIMPLE**

**ON** **UPDATE** **NO** ACTION

**ON** **DELETE** **NO** ACTION,

**CONSTRAINT** exame\_tipoexameid\_fkey **FOREIGN** **KEY** (tipoexameid)

**REFERENCES** **public**.tipoexame (tipoexameid) **MATCH** **SIMPLE**

**ON** **UPDATE** **NO** ACTION

**ON** **DELETE** **NO** ACTION

)

**create** **table** consulta(

consultaid serial **primary** **key**,

tipoconsultaid int **references** tipoconsulta,

medicoid int **references** medico,

pacienteid int **references** paciente,

recepcionistaid int **references** recepcionista,

dia Date,

hora Time,

descricao varchar(300),

prescricao varchar(300),

observacao varchar(200),

urgente boolean **default** **false**,

realizada boolean **default** **false**,

**unique** (medicoid, dia, hora)

);

**alter** **table** exame **add** preco numeric(10, 2);

**alter** **table** consulta **add** preco numeric(10, 2);

*-- Tabelas de Auditoria*

**CREATE** **TABLE** **public**.auditoria\_precos\_consultas

(

tipoconsultaid integer,

valor\_antigo numeric(10,2),

valor\_actual numeric(10,2),

data\_alteracao **timestamp** **without** time **zone** **DEFAULT** timezone('UTC'::text, **CURRENT\_TIMESTAMP**)

)

**CREATE** **TABLE** **public**.auditoria\_precos\_exames

(

tipoexameid integer,

valor\_antigo numeric(10,2),

valor\_actual numeric(10,2),

data\_alteracao **timestamp** **without** time **zone** **DEFAULT** timezone('UTC'::text, **CURRENT\_TIMESTAMP**)

)

## Scripts de criação de *Triggers*

*-- Auditoria de preco que permite saber quem, quando e qual foi alteração de preços feita para um determinado tipo de consulta*

**CREATE** **TRIGGER** trigger\_preco\_consulta **Before** **INSERT** **or** **update** **ON** tipoconsulta

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** auditprecoconsulta();

*--Auditoria de preço de Exames que permite saber quem, quando e qual foi alteração de preços feita para um determinado tipo de exame*

**CREATE** **TRIGGER** trigger\_preco\_exame **Before** **INSERT** **or** **update** **ON** tipoexame

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** auditprecoexame();

*--trigger para validar a data da marcacao de um exame, ou seja, não permite marcar exame para uma data passada*

**CREATE** **TRIGGER** trigger\_data\_exame **Before** **insert** **ON** exame

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** dataexame();

*--trigger valida data da marcação de uma consulta*

**CREATE** **TRIGGER** trigger\_data\_consulta **Before** **insert** **ON** consulta

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** dataconsulta();

*--trigger que permite determinar o preço de uma consulta a quão da sua marcação de acordo com o seu tipo.*

**CREATE** **TRIGGER** trigger\_alterar\_preco\_consulta **After** **INSERT** **ON** consulta

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** preencherprecoconsulta();

*--trigger que permite de um exame a quão da sua marcação de acordo com o seu tipo.*

**CREATE** **TRIGGER** trigger\_alterar\_preco\_exame **After** **INSERT** **ON** exame

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** preencherprecoexame();

*--trigger que impede que homens marquem uma consulta de genecologia*

**CREATE** **TRIGGER** trigger\_impedir\_genicologia\_homens **Before** **insert** **ON** consulta

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** impedir\_genicologia\_homens();

*-- Trigger para impedir marcao de consulta de pediatria para maiores de 12 anos*

**CREATE** **TRIGGER** trigger\_impedir\_pediatria **Before** **insert** **ON** consulta

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** impedir\_pediatria();

*-- Trigger para impedir inserção de medico menor de 18 anos*

**CREATE** **TRIGGER** trigger\_impedir\_medico\_menoridade **Before** **insert** **ON** medico

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** impedir\_menoridade();

*-- Trigger para impedir inserção de enfermeiro menor de 18 anos*

**CREATE** **TRIGGER** trigger\_impedir\_enfermeiro\_menoridade **Before** **insert** **ON** enfermeiro

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** impedir\_menoridade();

*-- Trigger para impedir inserção de recepcionista menor de 18 anos*

**CREATE** **TRIGGER** trigger\_impedir\_recepcionista\_menoridade **Before** **insert** **ON** recepcionista

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** impedir\_\_menoridade();

## Scripts de criação de Funções

*--função que permite saber quantas consultas existem por realizar*

**CREATE** **OR** **REPLACE** **FUNCTION** **public**."contaConsultaPorRealizar"(

)

**RETURNS** integer

**LANGUAGE** 'plpgsql'

COST 100

**VOLATILE**

**AS** $BODY$**declare**

contador integer;

**begin**

**select** **count**(\*) **into** contador **from** consulta **where** realizada=**false**;

**return** contador;

**end**;$BODY$;

*--função que permite saber quantos exames existem por realizar*

**CREATE** **OR** **REPLACE** **FUNCTION** **public**."contaExamePorRealizar"(

)

**RETURNS** bigint

**LANGUAGE** 'plpgsql'

**AS** $BODY$**declare**

contador integer;

**begin**

**select** **count**(\*) **into** contador **from** exame **where** realizado=**false**;

**return** contador;

**end**;$BODY$;

*--função que verifica a validade de uma certa data marcada para exame e retorna um trigger.*

**CREATE** **FUNCTION** **public**.dataexame()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

diferenca int;

**BEGIN**

**select** **Extract** (**day** **from** (**select** **new**.**data** - now())) **into** diferenca;

if(diferenca < 0) **then**

RAISE **EXCEPTION** 'Data invalida! A data deve ser superior a data actual';

**end** if;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função para verificar idade medico e retorna um trigger*

**CREATE** **FUNCTION** **public**.idademedico()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

diferenca int;

**BEGIN**

**select** trunc(**extract** (**day** **from** (**select** now() - to\_date(datanasc, 'YYYY-MM-DD')))/365) **into** idade;

if(idade < 18) **then**

RAISE **EXCEPTION** 'Idade invalida! O médico não pode ser menor de idade(18)';

**end** if;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função que verifica a validade de uma certa data marcada para consulta e retorna um trigger.*

**CREATE** **FUNCTION** **public**.dataconsulta()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

diferenca int;

**BEGIN**

**select** **Extract** (**day** **from** (**select** **new**.dia - now())) **into** diferenca;

if(diferenca < 0) **then**

RAISE **EXCEPTION** 'Data invalida! A data deve ser superior a data actual';

**end** if;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função que faz auditoria dos da alteração dos preços de um tipo de consulta, podendo se saber quando foi alterada, qual era o preco anterior e retorna um trigger.*

**CREATE** **FUNCTION** **public**.auditprecoconsulta()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**INSERT** **INTO** auditoria\_precos\_consultas(

tipoconsultaid, valor\_antigo, valor\_actual, data\_alteracao)

**VALUES** (**old**.tipoconsultaid, **old**.preco, **new**.preco, NOW());

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função que faz auditoria dos da alteração dos preços de um tipo de exame, podendo se saber quando foi alterada, qual era o preco anterior e retorna um trigger.*

**CREATE** **FUNCTION** **public**.auditprecoexame()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

COST 100

**VOLATILE** **NOT** LEAKPROOF

**AS** $BODY$

**BEGIN**

**INSERT** **INTO** auditoria\_precos\_exames(

tipoexameid, valor\_antigo, valor\_actual, data\_alteracao)

**VALUES** (**old**.tipoexameid, **old**.preco, **new**.preco, NOW());

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função que preenche o preço de uma determinada consulta de acordo com o seu tipo durante a marcação que retorna um trigger.*

*-- Preencher preco da consulta*

**CREATE** **FUNCTION** **public**.preencherprecoconsulta()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

tipoc int;

pr numeric(10,2);

**BEGIN**

tipoc := **new**.tipoconsultaid;

**select** preco **into** pr **from** tipoconsulta **where** tipoconsultaid = tipoc;

**update** consulta **set** preco = pr **where** preco **is** **null**;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função que preenche o preço de um determinado de acordo com o seu tipo durante a marcação que retorna um trigger.*

*-- Preencher preco da exame*

**CREATE** **FUNCTION** **public**.preencherprecoexame()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

tipoe int;

pr numeric(10,2);

**BEGIN**

tipoe := **new**.tipoexameid;

**select** preco **into** pr **from** tipoexame **where** tipoexameid = tipoe;

**update** exame **set** preco = pr **where** preco **is** **null**;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- Impedir Marcacao de genicologia para homens, uma vez que esta consulta é apenas para individuos do sexo femenino*

*-- Impedir Marcacao de genicologia para homens*

**CREATE** **FUNCTION** **public**.impedir\_genicologia\_homens()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

esp int;

s varchar(40);

**BEGIN**

**select** especialidadeid **into** esp **from** medico **where** medicoid = **new**.medicoid;

**select** sexo **into** s **from** pessoa **inner** **join** paciente **on** pessoaid=pacienteid **where** pessoaid=**new**.pacienteid;

if s = 'masculino' **or** s = 'Masculino' **then**

if esp = 2 **then**

RAISE **EXCEPTION** 'Nao eh permitido realizar consultas de genicologias para homens';

**end** if;

**end** if;

**Return** **new**;

**END**;

$BODY$;

*-- Função para impedir marcao de consulta de pediatria para maiores de 12 anos sendo esta a idade máxima para ir a consulta de pediatria*

*-- Trigger para impedir marcao de consulta de pediatria para maiores de 12 anos*

**CREATE** **FUNCTION** **public**.impedir\_pediatria()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

datanasc varchar(40);

idade int;

esp int;

**BEGIN**

**select** especialidadeid **into** esp **from** medico **where** medicoid = **new**.medicoid;

**select** datanascimento **into** datanasc **from** pessoa **inner** **join** paciente **on** pessoaid=pacienteid **where** pessoaid=**new**.pacienteid;

**select** trunc(**extract** (**day** **from** (**select** now() - to\_date(datanasc, 'YYYY-MM-DD')))/365) **into** idade;

if esp = 1 **then**

if idade > 12 **then**

Raise **Exception** 'Pediatria eh so para criancas menores de 12 anos.';

**end** if;

**end** if;

**Return** **new**;

**END**;

$BODY$;

*-- função para verificar idade de um certo funcionário e retorna um trigger*

**CREATE** **FUNCTION** idadeFuncionario()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

diferenca int;

**BEGIN**

**select** trunc(**extract** (**day** **from** (**select** now() - to\_date(datanasc, 'YYYY-MM-DD')))/365) **into** idade;

if(idade < 18) **then**

RAISE **EXCEPTION** 'Idade invalida! Não pode ser menor de idade(18)';

**end** if;

**RETURN** **NEW**;

**END**;

$BODY$;

*-- função para verificar idade de um certo funcionário e retorna um trigger*

**CREATE** **FUNCTION** impedir\_menoridade()

**RETURNS** **trigger**

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**Declare**

idade int;

**BEGIN**

**select** trunc(**extract** (**day** **from** (**select** now() - to\_date(datanasc, 'YYYY-MM-DD')))/365) **into** idade;

if(idade < 18) **then**

RAISE **EXCEPTION** 'Idade invalida! Não pode ser menor de idade(18)';

**end** if;

**RETURN** **NEW**;

**END**;

$BODY$;

## Scripts de criação de Procedimentos

*--Procedimento para marcar um exame especificando o tipo, o paciente, o recepcionista, o enfermeiro, a data e a hora.*

**CREATE** **OR** **REPLACE** **PROCEDURE** marcarExame(INT, INT, INT, INT, date, time)

**LANGUAGE** plpgsql

**AS** $$

**BEGIN**

*-- inserindo dados do exame na tabela exame*

**insert** **into** exame(tipoexameid, pacienteid, recepcionistaid, enferemeiroid, **data**, hora) **values**($1,$2,$3,$4,$5,$6);

**END**;

$$;

*--Procedimento que permite actualizar os dados de um determinado exame realizado de modo a saber se o seu resultado e observações feitas.*

**CREATE** **OR** **REPLACE** **PROCEDURE** realizarExame(INT, varchar, boolean)

**LANGUAGE** plpgsql

**AS** $$

**BEGIN**

*-- actualizando dados do exame na tabela exame*

**UPDATE** exame

**SET** observacao = $2, positivo=$3

**WHERE** exameid = $1;

**END**;

$$;

*--procedimento para marcar consulta especificando o tipo, o medico, o paciente, o recepcionista, o dia, a hora e a descricão*

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.marcarconsulta1(

integer,

integer,

integer,

integer,

date,

time **without** time **zone**,

character varying)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**insert** **into** consulta(tipoconsultaid,medicoid,pacienteid,recepcionistaid, dia, hora, descricao) **values**($1, $2,$3,$4,$5, $6, $7);

**END**;

$BODY$;

*--procedimento permite marcar um consulta como realizada podendo actualizar a prescrição e a observação*

**CREATE** **OR** **REPLACE** **PROCEDURE** realizarExame(INT, varchar, boolean)

**LANGUAGE** plpgsql

**AS** $$

**BEGIN**

*-- actualizando dados do exame na tabela exame*

**UPDATE** exame

**SET** observacao = $2, positivo=$3

**WHERE** exameid = $1;

**END**;

$$;

*--procedimento para listar consultas por realizar permitindo o recepcionista notificar tando os medicos assim como os pacientes caso haja alguma alteração*

*--listar consultas por realizar*

**CREATE** **OR** **REPLACE** **PROCEDURE** listarConsultasPorRealizar()

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**declare**

reg consulta%rowtype;

**BEGIN**

**for** reg **in** **select** \* **from** consulta **where** realizada=**false** Loop

raise notice 'Id Consulta: %', reg.consultaid;

raise notice 'Id Paciente: %', reg.pacienteid;

raise notice 'Data: %', reg.dia;

raise notice 'Hora: %', reg.hora;

raise notice 'Id médico: %', reg.medicoid;

raise notice '====================';

**end** loop;

**END**;

$BODY$

*--Listar as 10 ultimas consultas realizadas por um determinado paciente de modo que ele possa verificar o historico;*

*--Listar as 10 ultimas consultas realizadas por um determinado pacientes;*

**CREATE** **OR** **REPLACE** **PROCEDURE** listarDezUltimasConsultasRealizadas(int)

**LANGUAGE** plpgsql

**AS** $$

**declare**

reg consulta%rowtype;

**BEGIN**

**for** reg **in** **select** \* **from** consulta **where** pacienteid=$1 **and** realizada=**true** **order** **by** **data** **limit** 10

Loop

raise notice 'Id Consulta: %', reg.consultaid;

raise notice 'Data: %', reg.dia;

raise notice 'Hora: %', reg.hora;

raise notice '====================';

**end** loop;

**END**;

$$;

*--Listar os 10 ultimos exames realizados por um determinado paciente de modo que ele possa verificar o historico dos seus exames;*

*--Listar os 10 ultimos exames realizados por um determinado paciente;*

**CREATE** **OR** **REPLACE** **PROCEDURE** listarDezUltimosExamesRealizados(int)

**LANGUAGE** plpgsql

**AS** $$

**declare**

reg exame%rowtype;

**BEGIN**

**for** reg **in** **select** \* **from** exame **where** pacienteid=$1 **and** realizado=**true** **order** **by** **data** **limit** 10

Loop

raise notice 'Id Exame: %', reg.exameid;

raise notice 'Data: %', reg.**data**;

raise notice 'Hora: %', reg.hora;

raise notice '====================';

**end** loop;

**END**;

$$;

*--procedimento que permite criar uma determinada pessoa de modo a ser chamado na criacao de paciente, recpcionista, medico e enfermeiro uma vez que estes herdam boa parte dos atributos.*

*--inserirPessoa(bi, nuit, primeironome, apelido, email, sexo, endereco)*

**CREATE** **OR** **REPLACE** **PROCEDURE** inserirPessoa(varchar, varchar, varchar, varchar, varchar, varchar, varchar)

**LANGUAGE** plpgsql

**AS** $$

**BEGIN**

**insert** **into** pessoa (bi, nuit, primeironome, apelido, email, datanascimento,sexo, endereco)

**values** ($1, $2, $3, $4, $5, now(),$6, $7);

**END**;

$$

*--procedimento que permite inserir um determinado paciente e que reaproveita o metodo para inserir uma determinada pessoa.é importante pois permite numa so instrução criar uma pessoa e um paciente*

*--inserirPaciente(bi, nuit, primeironome, apelido, email, sexo, endereco, profissao)*

**CREATE** **OR** **REPLACE** **PROCEDURE** inserirPaciente(varchar, varchar, varchar, varchar, varchar, varchar, varchar, varchar)

**LANGUAGE** plpgsql

**AS** $$

**declare**

pid integer;

**BEGIN**

**call** inserirPessoa($1, $2, $3, $4, $5,$6, $7);

**select** pessoaid **into** pid **from** pessoa **where** bi=$1;

**insert** **into** paciente(pacienteid, profissao) **values**(pid, $8);

**END**;

$$

*--procedimento que permite inserir um determinado médico e que reaproveita o metodo para inserir uma determinada pessoa.é importante pois permite numa so instrução criar uma pessoa e um medico*

**CREATE** **OR** **REPLACE** **PROCEDURE** inserirMedico(varchar, varchar, varchar, varchar, varchar, varchar, varchar, varchar, int)**LANGUAGE** plpgsql **AS**

$$**declare** pid integer;

**BEGIN**

**call** inserirPessoa($1, $2, $3, $4, $5,$6, $7);

**select** pessoaid **into** pid **from** pessoa **where** bi=$1;

**insert** **into** medico(medicoid, carteiraprofissional, especialidadeid) **values**(pid, $8, $9);

**END**;$$

*procedimento que permite inserir um determinado recepcionista que reaproveita o metodo para inserir uma determinada pessoa.é importante pois permite numa so instrução criar uma pessoa e um recepcionista*

*--inserirRecepcionista(bi, nuit, primeironome, apelido, email, sexo, endereco)*

**CREATE** **OR** **REPLACE** **PROCEDURE** inserirRecepcionista(varchar, varchar, varchar, varchar, varchar, varchar, varchar)

**LANGUAGE** plpgsql

**AS** $$

**declare**

pid integer;

**BEGIN**

**call** inserirPessoa($1, $2, $3, $4, $5,$6, $7);

**select** pessoaid **into** pid **from** pessoa **where** bi=$1;

**insert** **into** recepcionista **values**(pid);

**END**;

$$

## Diversos

### Sequências

As sequências abaixo servem para gerar de forma automática o id das tabelas abaixo mencionadas.

*--sequência para incremementar o id das consultas*

**CREATE** SEQUENCE consulta\_consultaid\_seq

**INCREMENT** 1

**START** 38

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incrementar o id do diasemana*

**CREATE** SEQUENCE diasemana\_diaid\_seq

**INCREMENT** 1

**START** 7

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*----sequência para incremementar o id da especialidade*

**CREATE** SEQUENCE especialidade\_especialidadeid\_seq

**INCREMENT** 1

**START** 6

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incrementar o id do periodo*

**CREATE** SEQUENCE periodo\_periodoid\_seq

**INCREMENT** 1

**START** 3

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incremementar o id da pessoa*

**CREATE** SEQUENCE pessoa\_pessoaid\_seq

**INCREMENT** 1

**START** 11

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incrementar o id do utilizador*

**CREATE** SEQUENCE usuario\_usuarioid\_seq

**INCREMENT** 1

**START** 1

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incrementar o id do tipo de exame*

**CREATE** SEQUENCE tipoexame\_tipoexameid\_seq

**INCREMENT** 1

**START** 3

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

*--sequência para incrementar o id do tipo de consulta*

**CREATE** SEQUENCE tipoconsulta\_tipoconsultaid\_seq

**INCREMENT** 1

**START** 3

**MINVALUE** 1

**MAXVALUE** 2147483647

**CACHE** 1;

### Backups

Num ficheiro bat foi colocada a seguinte a instrução de seguida no Windows foi colocada a tarefa de executar o ficheiro semanalmente:

*--backup*

pg\_dump -U postgres -W -F t kliniek\_db > **c**:/backup/dump\_kliniek.tar

# Referências Bibliográficas

* Slide de PL/SQL - Subprogramas do professor Jordão Uache. 2019
* [http://www.postgresqltutorial.com/](http://www.postgresqltutorial.com/postgresql-create-procedure/)  visitado aos 13.11.2019
* <https://carto.com/help/working-with-data/sql-stored-procedures/> visitado aos 13.11.2019
* [https://www.guj.com.br/t/mostrar-apenas-os-ultimos-5-registros-do-banco/153725 visitado aos 14.11.2019](https://www.guj.com.br/t/mostrar-apenas-os-ultimos-5-registros-do-banco/153725%20visitado%20aos%2014.11.2019)
* <https://eager.io/blog/audit-postgres/> visitado aos 14.11.2019
* [http://www.postgresqltutorial.com/postgresql-backup-database/#](http://www.postgresqltutorial.com/postgresql-backup-database/) visitado aos 14.11.2019
* [http://www.postgresqltutorial.com/postgresql-backup-database/#](http://www.postgresqltutorial.com/postgresql-backup-database/) visitado aos 14.11.2019
* [https://www.devmedia.com.br/forum/trigger-de-auditoria/602012 visitado aos 14.11.2019](https://www.devmedia.com.br/forum/trigger-de-auditoria/602012%20visitado%20aos%2014.11.2019)
* [https://stackoverflow.com/questions/21037711/sublime-text-2-paste-with-colors-to-ms-word/39432069 visitado aos 15.11.2019](https://stackoverflow.com/questions/21037711/sublime-text-2-paste-with-colors-to-ms-word/39432069%20visitado%20aos%2015.11.2019)
* [https://github.com/n1k0/SublimeHighlight vistado aos 15.11.2019](https://github.com/n1k0/SublimeHighlight%20vistado%20aos%2015.11.2019)
* <https://www.ibm.com/developerworks/community/blogs/fd26864d-cb41-49cf-b719-d89c6b072893/entry/utilizando_cursores_no_postgresql3?lang=en> visitado aos 16.11.2019