CENTRO UNIVERSITÁRIO UNIVATES
CURSO DE GRADUAÇÃO ENGENHARIA DE SOFTWARE

# TRABALHO INDIVIDUAL - ENTREGA 3/3

Augusto Fritz

Lajeado, junho de 2017

# Restaurante

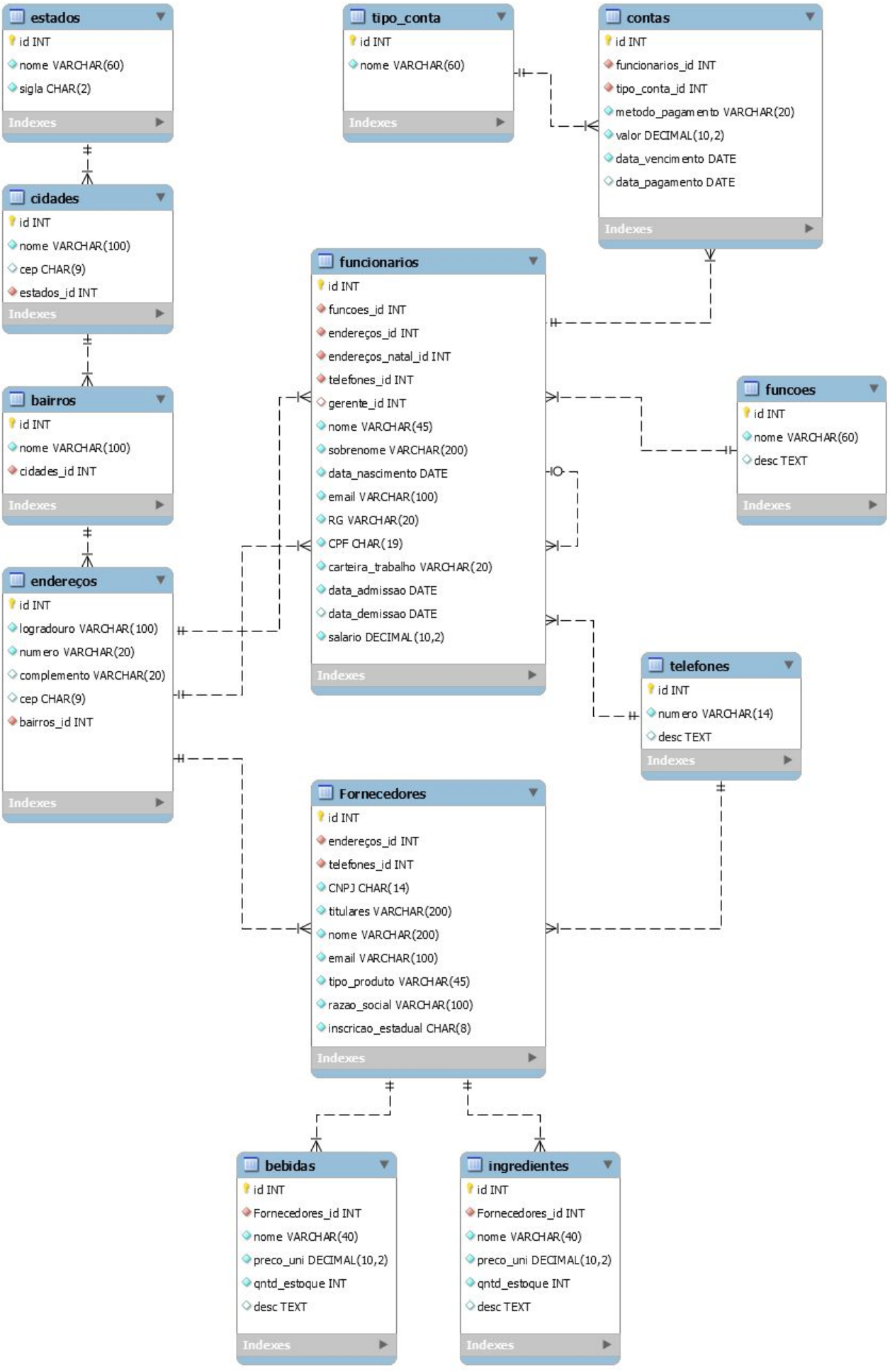
Para os funcionários, o restaurante pretende armazenar as informações de: Nome, data de nascimento, endereço, endereço natal, telefone(s), e-mail, RG e CPF, nº carteira de trabalho, data de admissão, data de demissão (se houver), função (Cozinheiro, garçom, atendente, etc…) e o seu respectivo salário. Os funcionários podem possuir ou serem gerentes.

Para o cadastro de fornecedores, será necessário cadastrar o CNPJ, o(s) titular(es), nome, endereço, telefone(s), e-mail, tipo de produto fornecido (ingrediente ou bebida), razão social e a inscrição estadual.

Contará com um sistema de estoque para os ingredientes e bebidas, possuindo o seu nome, preço unitário, fornecedor, quantidade atual em estoque e uma breve descrição se necessário.

O sistema irá registrar todas as contas a serem pagas (água, luz, gás, etc...), funcionário realizou o pagamento, forma de pagamento (O pagamento poderá ser feito através de dinheiro ou cartão), data de pagamento, vencimento e o valor a ser pago.

**Imagem do modelo ER se encontra presente na próxima página.**


estados
id INT
nome VARCHAR(60)
sigla CHAR(2)
Indexes
tipo_conta
id INT
nome VARCHAR(60)
Indexes
contas
id INT
funcionarios_id INT
tipo_conta_id INT
metodo_pagamento VARCHAR(20)
valor DECIMAL(10,2)
data_vencimento DATE
data_pagamento DATE
Indexes
cidades
id INT
nome VARCHAR(100)
cep CHAR(9)
estados_id INT
Indexes
funcionarios
id INT
funcoes_id INT
endereços_id INT
endereços_natal_id INT
telefones_id INT
gerente_id INT
nome VARCHAR(45)
sobrenome VARCHAR(200)
data_nascimento DATE
email VARCHAR(100)
RG VARCHAR(20)
CPF CHAR(19)
carteira_trabalho VARCHAR(20)
data_admissao DATE
data_demissao DATE
salario DECIMAL(10,2)
Indexes
bairros
id INT
nome VARCHAR(100)
cidades_id INT
Indexes
funcoes
id INT
nome VARCHAR(60)
desc TEXT
Indexes
endereços
id INT
logradouro VARCHAR(100)
numero VARCHAR(20)
complemento VARCHAR(20)
cep CHAR(9)
bairros_id INT
Indexes
telefones
id INT
numero VARCHAR(14)
desc TEXT
Indexes
Fornecedores
id INT
endereços_id INT
telefones_id INT
CNPJ CHAR(14)
titulares VARCHAR(200)
nome VARCHAR(200)
email VARCHAR(100)
tipo_produto VARCHAR(45)
razao_social VARCHAR(100)
inscricao_estadual CHAR(8)
Indexes
bebidas
id INT
Fornecedores_id INT
nome VARCHAR(40)
preco_uni DECIMAL(10,2)
qntd_estoque INT
desc TEXT
Indexes
ingredientes
id INT
Fornecedores_id INT
nome VARCHAR(40)
preco_uni DECIMAL(10,2)
qntd_estoque INT
desc TEXT
Indexes

## Querys para a inserção das tabelas e parâmetros:

```
CREATE TABLE estados(
	id INT,
	nome VARCHAR(60) NOT NULL,
	sigla CHAR(2) NOT NULL,
	CONSTRAINT pk_estados_id PRIMARY KEY (id));


CREATE TABLE cidades(
	id INT,
	nome VARCHAR(100) NOT NULL,
	cep CHAR(9),
	estados_id INT NOT NULL,
	CONSTRAINT pk_cidades_id PRIMARY KEY (id),
	CONSTRAINT fk_estados_id FOREIGN KEY (estados_id) REFERENCES estados (id));


CREATE TABLE bairros(
	id INT,
	nome VARCHAR(100) NOT NULL,
	cidades_id INT NOT NULL,
	CONSTRAINT pk_bairros_id PRIMARY KEY (id),
	CONSTRAINT fk_cidades_id FOREIGN KEY (cidades_id) REFERENCES cidades (id));


CREATE TABLE endereços(
	id INT,
	logradouro VARCHAR(100) NOT NULL,
	numero VARCHAR(20) NOT NULL,
	complemento VARCHAR(20),
	cep CHAR(9),
	bairros_id INT NOT NULL,
	CONSTRAINT pk_endereços_id PRIMARY KEY (id),
	CONSTRAINT fk_bairros_id FOREIGN KEY (bairros_id) REFERENCES bairros (id));


CREATE TABLE funcoes(
	id INT,
	nome VARCHAR(60) NOT NULL,
	descricao TEXT,
	CONSTRAINT pk_funcoes_id PRIMARY KEY (id));


CREATE TABLE telefones(
	id INT,
	numero VARCHAR(14) NOT NULL,
	descricao TEXT,
	CONSTRAINT pk_telefones_id PRIMARY KEY (id));
```

```
CREATE TABLE funcionarios(
	id INT,
	nome VARCHAR(45) NOT NULL,
	sobrenome VARCHAR(200) NOT NULL,
	data_nascimento DATE NOT NULL,
	email VARCHAR(100) NOT NULL,
	RG VARCHAR(20) NOT NULL,
	CPF CHAR(19) NOT NULL,
	carteira_trabalho VARCHAR(20) NOT NULL,
	data_admissao DATE NOT NULL,
	data_demissao DATE,
	salario DECIMAL(10,2) NOT NULL,
	funcoes_id INT NOT NULL,
	endereços_id INT NOT NULL,
	endereços_natal_id INT NOT NULL,
	telefones_id INT NOT NULL,
	gerente_id INT,
	CONSTRAINT pk_funcionarios_id PRIMARY KEY (id),
	CONSTRAINT fk_funcoes_id FOREIGN KEY (funcoes_id) REFERENCES funcoes (id),
	CONSTRAINT fk_endereços_id FOREIGN KEY (endereços_id) REFERENCES endereços
(id),
	CONSTRAINT fk_telefones_id FOREIGN KEY (telefones_id) REFERENCES telefones (id),
	CONSTRAINT fk_gerente_id FOREIGN KEY (gerente_id) REFERENCES funcionarios (id));

ALTER TABLE funcionarios ADD CONSTRAINT fk_endereços_natal_id FOREIGN KEY
(endereços_natal_id) REFERENCES endereços (id);


CREATE TABLE tipo_conta(
	id INT,
	nome VARCHAR(60) NOT NULL,
	CONSTRAINT pk_tipo_conta_id PRIMARY KEY (id));


CREATE TABLE contas (
	id INT,
	metodo_pagamento VARCHAR(20) NOT NULL,
	valor DECIMAL(10,2) NOT NULL,
	data_vencimento DATE NOT NULL,
	date_pagamento DATE NOT NULL,
	funcionarios_id INT NOT NULL,
	tipo_conta_id INT NOT NULL,
	CONSTRAINT pk_contas_id PRIMARY KEY (id),
	CONSTRAINT fk_funcionarios_id FOREIGN KEY (funcionarios_id) REFERENCES
funcionarios (id),
	CONSTRAINT fk_tipo_conta_id FOREIGN KEY (tipo_conta_id) REFERENCES tipo_conta
(id));
```

```
CREATE TABLE fornecedores (
	id INT,
	CNPJ CHAR(14) NOT NULL,
	titulares VARCHAR(200) NOT NULL,
	nome VARCHAR(200) NOT NULL,
	email VARCHAR(100) NOT NULL,
	tipo_produto VARCHAR(45) NOT NULL,
	razao_social VARCHAR(100) NOT NULL,
	inscricao_estadual CHAR(8) NOT NULL,
	endereços_id INT NOT NULL,
	telefones_id INT NOT NULL,
	CONSTRAINT pk_fornecedores_id PRIMARY KEY (id),
	CONSTRAINT fk_endereços_id FOREIGN KEY (endereços_id) REFERENCES endereços
(id),
	CONSTRAINT fk_telefones_id FOREIGN KEY (telefones_id) REFERENCES telefones (id));


CREATE TABLE bebidas (
	id INT,
	nome VARCHAR(40),
	preco_uni DECIMAL(10,2) NOT NULL,
	qntd_estoque INT NOT NULL,
	descricao TEXT,
	fornecedores_id INT NOT NULL,
	CONSTRAINT pk_bebidas_id PRIMARY KEY (id),
	CONSTRAINT fk_fornecedores_id FOREIGN KEY (fornecedores_id) REFERENCES
fornecedores (id));


CREATE TABLE ingredientes (
	id INT,
	nome VARCHAR(40),
	preco_uni DECIMAL(10,2) NOT NULL,
	qntd_estoque INT NOT NULL,
	descricao TEXT,
	fornecedores_id INT NOT NULL,
	CONSTRAINT pk_ingredientes_id PRIMARY KEY (id),
	CONSTRAINT fk_fornecedores_id FOREIGN KEY (fornecedores_id) REFERENCES
fornecedores (id));
```

## Querys utilizadas para os registros:

```
INSERT INTO estados VALUES
        (1,'Rio Grande do Sul','RS'),
        (2,'Santa Catarina','SC'),
        (3,'Paraná','PR'),
        (4,'São Paulo','SP'),
        (5,'Rio de Janeiro','RJ'),
        (6,'Espírito Santo','ES'),
        (7,'Bahia','BA'),
        (8,'Sergipe','SE'),
        (9,'Alagoas','AL'),
        (10,'Pernambuco','PE'),
        (11,'Paraíba','PB'),
        (12,'Rio Grande do Norte','RN'),
        (13,'Ceará','CE'),
        (14,'Piaui','PI'),
        (15,'Minas Gerais','MG');


INSERT INTO cidades(id,nome,estados_id) VALUES
        (1,'Lajeado',1),
        (2,'Blumenau',2),
        (3,'Curitiba',3),
        (4,'São Paulo',4),
        (5,'Rio de Janeiro',5),
        (6,'Linhares',6),
        (7,'Salvador',7),
        (8,'Laranjeiras',8),
        (9,'Maceió',9),
        (10,'Olinda',10),
        (11,'Campina Grande',11),
        (12,'Natal',12),
        (13,'Fortaleza',13),
        (14,'Bom Jesus',14),
        (15,'Juíz de Fora',15);


INSERT INTO bairros VALUES
        (1,'Florestal',1),
        (2,'Centro',1),
        (3,'São Cristóvão',1),
        (4,'Moinhos',1),
        (5,'Universitário',1),
        (6,'Montanha',1),
        (7,'Casa Verde',4),
        (8,'Centro',7),
        (9,'Botafogo',5),
        (10,'Lagoa Nova',12),
        (11,'Jardim Cearense',13),
        (12,'Ouro Preto',10);
```

```
INSERT INTO endereços (id,logradouro,numero,bairros_id) VALUES
	(1,'Rua 1',1,1),
	(2,'Rua 2',2,2),
	(3,'Rua 3',3,3),
	(4,'Rua 4',4,4),
	(5,'Rua 5',5,5),
	(6,'Rua 6',6,6),
	(7,'Rua 7',7,7),
	(8,'Rua 8',8,8),
	(9,'Rua 9',9,9),
	(10,'Rua 10',10,10),
	(11,'Rua 11',11,11),
	(12,'Rua 12',12,12),
	(13,'Rua 13',13,1),
	(14,'Rua 14',14,2),
	(15,'Rua 15',15,3),
	(16,'Rua 16',16,4),
	(17,'Rua 17',17,5),
	(18,'Rua 18',18,6),
	(19,'Rua 19',19,7),
	(20,'Rua 20',20,8),


INSERT INTO endereços (id,logradouro,numero,bairros_id) VALUES
	(13,'Rua 13',13,1),
	(14,'Rua 14',14,2),
	(15,'Rua 15',15,3),
	(16,'Rua 16',16,4),
	(17,'Rua 17',17,5),
	(18,'Rua 18',18,6),
	(19,'Rua 19',19,7),
	(20,'Rua 20',20,8);


INSERT INTO funcoes VALUES
	(1,'Chef'),
	(2,'Cozinheiro'),
	(3,'Auxiliar de Cozinha'),
	(4,'Caixa'),
	(5,'Gerente'),
	(6,'Garçom'),
	(7,'Faxineiro'),
	(8,'Saladeiro'),
	(9,'Churrasqueiro'),
	(10,'Recepcionista');
```

```
INSERT INTO telefones VALUES
	(1, '051 1111-1111'),
	(2, '051 2222-2222'),
	(3, '051 3333-3333'),
	(4, '051 4444-4444'),
	(5, '051 5555-5555'),
	(6, '051 6666-6666'),
	(7, '051 7777-7777'),
	(8, '051 8888-8888'),
	(9, '051 9999-9999'),
	(10, '051 1010-1010'),
	(11, '051 1111-1111'),
	(12, '051 1212-1212'),
	(13, '051 1313-1313'),
	(14, '051 1414-1414'),
	(15, '051 1515-1515'),
	(16, '051 1616-1616'),
	(17, '051 1717-1717'),
	(18, '051 1818-1818'),
	(19, '051 1919-1919'),
	(20, '051 2020-2020');


INSERT INTO fornecedores VALUES
	(1,'11111111111111','João','F1','f1@email.com','bebidas','Fornecedor 1','11111111',1,1),
	(2,'22222222222222','Maria','F2','f2@email.com','ingredientes','Fornecedor 2','22222222',2,2),
	(3,'33333333333333','Paulo','F3','f3@email.com','bebidas','Fornecedor 3','33333333',3,3),
	(4,'44444444444444','Ana','F4','f4@email.com','ingredientes','Fornecedor 4','44444444',4,4),
	(5,'55555555555555','Leandro','F5','f5@email.com','bebidas','Fornecedor 5','55555555',5,5),
	(6,'66666666666666','Salete','F6','f6@email.com','ingredientes','Fornecedor
6','66666666',6,6),
	(7,'77777777777777','Pedro','F7','f7@email.com','bebidas','Fornecedor 7','77777777',7,7),
	(8,'88888888888888','Vitória','F8','f8@email.com','ingredientes','Fornecedor
8','88888888',8,8),
	(9,'99999999999999','Thomas','F9','f9@email.com','bebidas','Fornecedor 9','99999999',9,9),
	(10,'10101010101010','Milena','F10','f10@email.com','ingredientes','Fornecedor
10','10101010',10,10);


INSERT INTO bebidas (id,nome,preco_uni,qntd_estoque,fornecedores_id) VALUES
	(1, 'Coca Cola (lata)', 3.50 , 50, 1),
	(2, 'Pepsi (late)', 3.50, 60, 3),
	(3, 'Suco Laranja', 2.50, 35, 5),
	(4, 'Suco Uva', 2.50, 38, 7),
	(5, 'Agua (600ml)', 2.50, 54, 9);


INSERT INTO ingredientes (id,nome,preco_uni,qntd_estoque,fornecedores_id) VALUES
	(1, 'Carne rês', 25, 50, 2),
	(2, 'Carne frago', 20, 60, 4),
	(3, 'Tomate', 1.50, 80, 6),
```

```
	(4, 'Cebola', 1.50, 80, 8),
	(5, 'Alface', 3, 20, 10);


INSERT INTO funcionarios (id, nome, sobrenome, data_nascimento, email, RG, CPF,
carteira_trabalho, data_admissao, salario, funcoes_id, endereços_id, endereços_natal_id,
telefones_id) VALUES
	(1, 'Augusto', 'Fritz', '19980115', 'a.fritz@email.com', '1111111111',
'111.111.111-11','1111111111','20160620','4000.00',1,15,15,11),
	(2, 'Maria', 'Santos', '19920620','m.santos@email.com','2222222222',
'222.222.222-22','2222222222','20170226','2500.00',5,13,8,12);


INSERT INTO funcionarios (id, nome, sobrenome, data_nascimento, email, RG, CPF,
carteira_trabalho, data_admissao, salario, funcoes_id, endereços_id, endereços_natal_id,
telefones_id,gerente_id) VALUES
	(3, 'João', 'Filho', '19940824', 'jp.filho@email.com','3333333333',
'333.333.333-33','3333333333','20160924','2000.00',6,14,9,13,2),
	(4, 'Veronica', 'Silva', '19950414', 've.silva@email.com', '4444444444', '444.444.444-44',
'4444444444', '20170129','1500.00',7,15,11,14,2),
	(5, 'Vitório', 'Silva', '19950414', 'vi.silva@email.com', '5555555555', '555.555.555-55',
'5555555555', '20170129','1800.00',4,15,11,15,2),
	(6, 'Milena', 'Montanha', '19900728', 'm.montanha@email.com', '6666666666',
'666.666.666-66', '6666666666', '20161205','1600.00',10,16,16,16,2),
	(7, 'Pedro', 'Sampaio', '19890326', 'p.sampaio@email.com', '7777777777',
'777.777.777-77','7777777777','20160620','3000.00',9,17,8,17,1);


INSERT INTO funcionarios VALUES
	(8, 'Marina', 'Santana', '19951010', 'm.santana@email.com', '8888888888',
'888.888.888-88','8888888888','20160620','20160820','2900.00',8,18,7,18,1),
	(9, 'Gustavo', 'Cadeira', '19931107', 'g.cadeira@email.com', '9999999999',
'999.999.999-99','9999999999','20160615','20170220','2500.00',3,19,5,19,1),
	(10, 'Carol', 'Medeiros', '19920421', 'c.medeiros@email.com', '1010101010',
'101.010.101-01','1010101010','20160729','20161205','2950.00',2,20,20,20,1);


INSERT INTO tipo_conta VALUES
	(1,'Água'),
	(2,'Energia'),
	(3,'Aluguel'),
	(4,'Telefone'),
	(5,'Multa'),
	(6,'Reparos');


INSERT INTO contas VALUES
	(1, 'Cartão', '800.00', '20170420','20170417', 3, 2),
	(2, 'Cartão', '1500.00', '20170422','20170417', 4, 3),
	(3, 'Cartão', '100.00', '20170419','20170413', 5, 4),
	(4, 'Dinheiro', '150.00', '20170510','20170502', 6, 5),
```

```
(5, 'Dinheiro', '400.00', '20170428','20170424', 2, 6),
(6, 'Cartão', '600.00', '20170415', '20170406', 2, 1),
(7, 'Dinheiro', '1500.00', '20170322', '20170315', 1, 3),
(8, 'Dinheiro', '300.00', '20170310', '20170309', 1, 5),
(9, 'Cartão', '1000.00', '20170320','20170315',4,2),
(10,'Dinheiro','120.00', '20170319','20170312',5,4);
```

**Consultas:**

1. Nome dos funcionários em ordem alfabética, onde todos os funcionários que tem como gerente o funcionário 1, receberam um aumento de 10%, enquanto os que possuem como gerente o funcionário 2, receberão um aumento de 15%.

```
SELECT nome , gerente_id, salario, CAST ((salario * 1.10) AS DECIMAL (10,2)) AS "novo_salario"
FROM funcionarios
WHERE gerente_id = 1
UNION
SELECT nome , gerente_id, salario, CAST ((salario * 1.15) AS DECIMAL (10,2)) AS "novo_salario"
FROM funcionarios
WHERE gerente_id = 2
ORDER BY nome;
```

| nome character... | gerente... integer | salario numeric ... | novo_sa... numeric ... |
|---|---|---|---|
| Carol | 1 | 2950 | 3245 |
| Gustavo | 1 | 2500 | 2750 |
| João | 2 | 2000 | 2300 |
| Marina | 1 | 2900 | 3190 |
| Milena | 2 | 1600 | 1840 |
| Pedro | 1 | 3000 | 3300 |
| Veronica | 2 | 1500 | 1725 |
| Vitório | 2 | 1800 | 2070 |

2. Todos os funcionários que foram contratados em 2016.

```
SELECT nome ||' '|| sobrenome AS "Nome Completo", data_admissao
FROM funcionarios
WHERE data_admissao BETWEEN '20160101' AND '20161231'
ORDER BY data_admissao;
```

| Nome Completo text | data_admissao date |
|---|---|
| Gustavo Cadeira | 2016-06-15 |
| Pedro Sampaio | 2016-06-20 |
| Augusto Fritz | 2016-06-20 |
| Marina Santana | 2016-06-20 |
| Carol Medeiros | 2016-07-29 |
| João Filho | 2016-09-24 |
| Milena Montanha | 2016-12-05 |

3. Todas as contas com o nome completo dos respectivos funcionários que pagaram.

```
SELECT contas.id, funcionarios.nome ||' '|| funcionarios.sobrenome AS "Nome_Completo"
FROM contas
INNER JOIN funcionarios ON contas.funcionarios_id = funcionarios.id
ORDER BY id;
```

| id integer | Nome_Com... text |
|---|---|
| 1 | João Filho |
| 2 | Veronica Silva |
| 3 | Vitório Silva |
| 4 | Milena Mont... |
| 5 | Maria Santos |
| 6 | Maria Santos |
| 7 | Augusto Fritz |
| 8 | Augusto Fritz |
| 9 | Veronica Silva |
| 10 | Vitório Silva |

4. Nome de todos os funcionários, com o id das contas que já pagaram.

```
SELECT funcionarios.nome ||' '|| funcionarios.sobrenome AS "Nome_Completo", contas.id
FROM funcionarios
LEFT JOIN contas ON funcionarios.id=contas.funcionarios_id;
```

| Nome_Completo text | id integer |
|---|---|
| João Filho | 1 |
| Veronica Silva | 2 |
| Vitório Silva | 3 |
| Milena Montanha | 4 |
| Maria Santos | 5 |
| Maria Santos | 6 |
| Augusto Fritz | 7 |
| Augusto Fritz | 8 |
| Veronica Silva | 9 |
| Vitório Silva | 10 |
| Carol Medeiros | |
| Marina Santana | |
| Gustavo Cadeira | |
| Pedro Sampaio | |

5. Todos os fornecedores de bebidas.

```
SELECT id, nome, razao_social, tipo_produto
FROM fornecedores
WHERE tipo_produto IN ('bebidas');
```

| id integer | nome characte... | razao_social character va... | tipo_pro... characte... |
|---|---|---|---|
| 1 | F1 | Fornecedor 1 | bebidas |
| 3 | F3 | Fornecedor 3 | bebidas |
| 5 | F5 | Fornecedor 5 | bebidas |
| 7 | F7 | Fornecedor 7 | bebidas |
| 9 | F9 | Fornecedor 9 | bebidas |

6. Todos os funcionários que se mudaram do endereço natal.

```
SELECT id, nome||' '|| sobrenome, endereços_id, endereços_natal_id
FROM funcionarios
WHERE endereços_id != endereços_natal_id;
```

| id integer | ?column? text | endereç... integer | endereç... integer |
|---|---|---|---|
| 2 | Maria Santos | 13 | 8 |
| 3 | João Filho | 14 | 9 |
| 4 | Veronica Silva | 15 | 11 |
| 5 | Vitório Silva | 15 | 11 |
| 7 | Pedro Sampaio | 17 | 8 |
| 8 | Marina Santana | 18 | 7 |
| 9 | Gustavo Cadeira | 19 | 5 |

7. Funcionários os quais os nomes comecem em A ou V.

```
SELECT nome
FROM funcionarios
WHERE nome LIKE 'A%' OR nome LIKE 'V%';
```

| nome characte... |
|---|
| Augusto |
| Veronica |
| Vitório |

8. Nomear todas as cidades com seus respectivos bairros, existindo ou não no banco.

```
SELECT cidades.nome, bairros.nome
FROM bairros
RIGHT JOIN cidades ON cidades.id = bairros.cidades_id
ORDER BY cidades.id;
```

| nome character varying | nome characte... |
|---|---|
| Lajeado | Florestal |
| Lajeado | Centro |
| Lajeado | São Crist... |
| Lajeado | Moinhos |
| Lajeado | Universit... |
| Lajeado | Montanha |
| Blumenau | |
| Curitiba | |
| São Paulo | Casa Ver... |
| Rio de Janeiro | Botafogo |
| Linhares | |
| Salvador | Centro |
| Laranjeiras | |
| Maceió | |
| Olinda | Ouro Preto |
| Campina Grande | |
| Natal | Lagoa N... |
| Fortaleza | Jardim C... |
| Bom Jesus | |
| Juíz de Fora | |

9. Fornecedores e o respectivo produto.

```
SELECT f.id, f.nome, f.tipo_produto, b.nome
FROM fornecedores f, bebidas b
WHERE b.fornecedores_id = f.id
UNION
SELECT f.id, f.nome, f.tipo_produto, i.nome
FROM fornecedores f, ingredientes i
WHERE i.fornecedores_id = f.id
ORDER BY id;
```

| id integer | nome characte... | tipo_produto character v... | nome character varying |
|---|---|---|---|
| 1 | F1 | bebidas | Coca Cola (lata) |
| 2 | F2 | ingredientes | Carne rês |
| 3 | F3 | bebidas | Pepsi (late) |
| 4 | F4 | ingredientes | Carne frago |
| 5 | F5 | bebidas | Suco Laranja |
| 6 | F6 | ingredientes | Tomate |
| 7 | F7 | bebidas | Suco Uva |
| 8 | F8 | ingredientes | Cebola |
| 9 | F9 | bebidas | Agua (600ml) |
| 10 | F10 | ingredientes | Alface |

10. Endereço completo dos funcionários 1, 3, 7.

```
SELECT f.nome, e.logradouro, e.numero, b.nome, c.nome, es.sigla
FROM funcionarios f, endereços e, bairros b, cidades c, estados es
WHERE f.endereços_id = e.id AND b.id = e.bairros_id AND c.id = b.cidades_id AND es.id =
c.estados_id;
```

| nome character... | logradou... character... | numero character... | nome character va... | nome character... | sigla character |
|---|---|---|---|---|---|
| Marina | Rua 18 | 18 | Montanha | Lajeado | RS |
| Pedro | Rua 17 | 17 | Universitário | Lajeado | RS |
| Milena | Rua 16 | 16 | Moinhos | Lajeado | RS |
| Vitório | Rua 15 | 15 | São Cristóvão | Lajeado | RS |
| Veronica | Rua 15 | 15 | São Cristóvão | Lajeado | RS |
| Augusto | Rua 15 | 15 | São Cristóvão | Lajeado | RS |
| João | Rua 14 | 14 | Centro | Lajeado | RS |
| Maria | Rua 13 | 13 | Florestal | Lajeado | RS |
| Gustavo | Rua 19 | 19 | Casa Verde | São Paulo | SP |
| Carol | Rua 20 | 20 | Centro | Salvador | BA |

**Consultas Complexas:**

1) Média do valor das contas pagas pelos funcionários com a presença de seus respectivos nomes.

```
SELECT c.funcionarios_id, f.nome AS "Nome", ROUND (AVG (c.valor) , 2) AS "Média"
FROM contas c INNER JOIN funcionarios f ON c.funcionarios_id = f.id
GROUP BY c.funcionarios_id, f.nome
ORDER BY c.funcionarios_id ASC;
```

| funcionari... integer | Nome character varying (45) | Média numeric |
|---|---|---|
| 1 | Augusto | 900.00 |
| 2 | Maria | 500.00 |
| 3 | João | 800.00 |
| 4 | Veronica | 1250.00 |
| 5 | Vitório | 110.00 |
| 6 | Milena | 150.00 |

2) Maior salário dentre os funcionários que possuem um gerente.

```
SELECT f.gerente_id, MAX (f.salario)
FROM funcionarios f
WHERE gerente_id IS NOT NULL
GROUP BY f.gerente_id;
```

| gerente_id integer | max numeric |
|---|---|
| 1 | 3000.00 |
| 2 | 2000.00 |

3) A data de admissão mais antiga para o empregado de cada gerente.

```
SELECT f.gerente_id, MIN(f.data_admissao)
FROM funcionarios f
WHERE f.data_demissao IS NULL
AND f.gerente_id IS NOT NULL
GROUP BY f.gerente_id;
```

| gerente_id integer | min date |
|---|---|
| 1 | 2016-06-20 |
| 2 | 2016-09-24 |

4) Todas as cidades com mais de um bairro registrado.

```
SELECT c.nome , COUNT (b.nome) AS "Bairros"
FROM cidades c INNER JOIN bairros b ON c.id = b.cidades_id
GROUP BY c.nome
HAVING COUNT(b.nome) > 1;
```

| nome character varying (100) | Bairros bigint |
|---|---|
| Lajeado | 6 |

5) Funcionários que recebem menos que a média salarial.

```
SELECT f.id, f.nome, f.salario
FROM funcionarios f
WHERE f.salario < (SELECT AVG(f2.salario)
                   FROM funcionarios f2)
ORDER BY f.id ASC;
```

| id<br>integer | nome<br>character varying (45) | salario<br>numeric (... |
|---|---|---|
| 3 | João | 2000.00 |
| 4 | Veronica | 1500.00 |
| 5 | Vitório | 1800.00 |
| 6 | Milena | 1600.00 |

6) Estados que contenham uma cidade com a letra "J" registrada no sistema. Listar o estado, sigla e o nome da cidade.

```
SELECT e.nome AS "estado", e.sigla, c.nome AS "cidade"
FROM estados e INNER JOIN cidades c ON e.id = c.estados_id
 WHERE c.id IN (SELECT id
FROM cidades c2
 WHERE c2.nome LIKE '%j%')
```

| estado<br>character varying (60) | sigla<br>character (2) | cidade<br>character varying (100) |
|---|---|---|
| Rio Grande do Sul | RS | Lajeado |
| Sergipe | SE | Laranjeiras |

7) Fornecedores de bebidas que o produto vendido é mais caro que a média das bebidas.

```
SELECT f.nome, b.nome, b.preco_uni
FROM fornecedores f INNER JOIN bebidas b ON f.id = b.fornecedores_id
GROUP BY f.nome, b.nome, b.preco_uni
HAVING b.preco_uni >
(SELECT AVG(b3.preco_uni)
FROM bebidas b3)
```

| nome<br>character varying (200) | nome<br>character varying (40) | preco_uni<br>numeric (... |
|---|---|---|
| F1 | Coca Cola (lata) | 3.50 |
| F3 | Pepsi (late) | 3.50 |

8) Contas que o funcionário o qual a pagou possui a letra "M" no sobrenome.

```
SELECT c.id AS "ID conta", c.funcionarios_id AS "ID Funcionario", f.nome, f.sobrenome
FROM contas c INNER JOIN funcionarios f ON  f.id = c.funcionarios_id
WHERE f.id IN (SELECT f2.id
                        FROM funcionarios f2
                        WHERE f2.sobrenome LIKE '%M%')
```

| ID conta integer | ID Funcio... integer | nome character varying (45) | sobrenome character varying (200) |
|---|---|---|---|
| 4 | 6 | Milena | Montanha |

9) Mostrar o endereço de todos os funcionários os quais possuem o salário maior que a média salarial.

```
SELECT f.id, f.nome, e.logradouro, e.numero
FROM funcionarios f INNER JOIN endereços e ON e.id = f.endereços_id,
vw_func_salarioMMA vw
WHERE f.salario > vw.media
 ORDER BY f.id ASC;
```

| id integer | nome character varying (45) | logradouro character varying (100) | numero character varying (20) |
|---|---|---|---|
| 1 | Augusto | Rua 15 | 15 |
| 2 | Maria | Rua 13 | 13 |
| 7 | Pedro | Rua 17 | 17 |
| 8 | Marina | Rua 18 | 18 |
| 9 | Gustavo | Rua 19 | 19 |
| 10 | Carol | Rua 20 | 20 |

10) Telefones dos funcionários com salário menor que a média. Não contar funcionários já demitidos.

```
SELECT f.id, f.nome, t.numero
FROM funcionarios f INNER JOIN telefones t ON f.telefones_id = t.id, vw_func_salarioMMA
vw
WHERE f.data_demissao IS NULL
AND f.salario > vw.media
ORDER BY f.id ASC;
```

| id integer | nome character varying (45) | numero character varying (14) |
|---|---|---|
| 1 | Augusto | 051 1111-1111 |
| 2 | Maria | 051 1212-1212 |
| 7 | Pedro | 051 1717-1717 |

11) Salário do último funcionário a ser contratado e o salário do último funcionário a ser demitido.

```
SELECT f.nome, f.salario
```

```
FROM funcionarios f, vw_func_data_admissao vw1, vw_func_data_demissao vw2
WHERE f.data_admissao = vw1.tarde
OR f.data_demissao = vw2.tarde;
```

| nome<br>character varying (45) | salario<br>numeric (... |
|---|---|
| Maria | 2500.00 |
| Gustavo | 2500.00 |

12) Nome completo, telefone e endereço de todos funcionários contratados após a última demissão.

```
SELECT f.id, vw1.nome, vw1.telefone, vw1.endereço
FROM vw_func_nome_tel_end vw1, vw_func_data_demissao vw2, funcionarios f
WHERE f.data_admissao > vw2.tarde
AND f.nome ||' '|| f.sobrenome = vw1.nome
ORDER BY f.id;
```

| id<br>integer | nome<br>text | telefone<br>character varying (14) | endereço<br>text |
|---|---|---|---|
| 2 | Maria Santos | 051 1212-1212 | Rua 13 nº13 |

**VIEWS:**

Salário máximo, mínimo e a média:

```
CREATE VIEW vw_func_salarioMMA (maximo, minimo, media) AS
SELECT MAX (f.salario), MIN (f.salario), AVG (f.salario)
FROM funcionarios f;
```

Data de demissão mais antiga e mais nova:

```
CREATE VIEW vw_func_data_demissao(cedo, tarde) AS
SELECT MIN (f.data_demissao), MAX(f.data_demissao)
FROM funcionarios f;
```

Data de admissão mais antiga e mais nova:

```
CREATE VIEW vw_func_data_admissao(cedo, tarde) AS
SELECT min(f.data_admissao), MAX(f.data_admissao)
FROM funcionarios f;
```

Nome, telefone e endereço (logradouro + número) de todos os funcionários:

```
CREATE VIEW vw_func_nome_tel_end (nome, telefone, endereço) AS
SELECT f.nome ||' '|| f.sobrenome AS "Nome Completo", t.numero, e.logradouro ||' nº'||
e.numero
FROM funcionarios f INNER JOIN telefones t ON t.id = f.telefones_id, endereços e
WHERE f.endereços_id = e.id;
```

## Querys utilizadas para as Stored Procedures:

1. Stored Procedure que, quando ativada, adiciona 1000 novos registros a tabela “endereços”.

```
CREATE OR REPLACE FUNCTION addbairros ()
	RETURNS INTEGER
	AS $$
	DECLARE
		quant INTEGER;
		bairro INTEGER;
		total INTEGER;
	BEGIN
		quant := (SELECT COUNT (id)
			  FROM endereços);
			quant := quant + 1;
			total := quant + 999;
			WHILE quant <= total
			LOOP
				bairro := CAST (RANDOM()*11+1 AS INTEGER);
				INSERT INTO endereços
				(id, logradouro, numero, bairros_id)
				VALUES
				(quant,'Rua ' || quant, quant, bairro);
				quant := quant + 1;
			END LOOP;
			RETURN 0;
		END;
	$$ LANGUAGE plpgsql;

SELECT addbairros();
```

2. Stored Procedure que verifica se o funcionário já pagou alguma conta pela empresa. Se sim, aumentar o salário em R$200.

```
CREATE OR REPLACE FUNCTION exer2 (INT)
	RETURNS INTEGER
	AS $$
	DECLARE
		salarioatual DECIMAL(10,2);
		exer2cursor CURSOR FOR
```

```
        SELECT *
        FROM contas;
        BEGIN
        IF EXISTS (SELECT *
                        FROM contas
                        WHERE funcionarios_id = $1) THEN
                                salarioatual := (SELECT salario
                                                FROM funcionarios
                                                WHERE id = $1);
                                UPDATE funcionarios
                                SET salario = salarioatual + 200
                                WHERE id = $1;
                        RETURN 1;
                ELSE
                        RETURN 0;
        END IF;
        END;
        $$ LANGUAGE plpgsql;

SELECT *
FROM exer2(2);
```

3. Stored Procedure que procura e lê todas as informações de funcionários através do ID do gerente.

```
CREATE OR REPLACE FUNCTION selectfunc (gerente INTEGER)
        RETURNS SETOF funcionarios
        AS $$
        DECLARE
                selfunc RECORD;
        BEGIN
                FOR selfunc IN (SELECT *
                                FROM funcionarios
                                WHERE gerente_id = gerente) LOOP
                                RETURN NEXT selfunc;
                END LOOP;
        END;
        $$ LANGUAGE plpgsql;

SELECT *
FROM selectfunc(2);
```

4. Stored Procedure que, após receber duas IDs, mostra a o maior valor entre um intervalo de duas contas.

```
CREATE OR REPLACE FUNCTION exer3 (inicio INTEGER, fim INTEGER)
        RETURNS DECIMAL (10,2)
```

```
AS $$
DECLARE
	exer3row contas%ROWTYPE;
	resultado DECIMAL (10,2);
BEGIN
	IF EXISTS (SELECT *
			FROM contas
			WHERE id BETWEEN inicio AND fim) THEN
			   SELECT *
			   INTO exer3row
			   FROM contas
			   WHERE valor =(SELECT MAX(valor)
					FROM contas
					    WHERE id BETWEEN inicio AND fim);
			   resultado := exer3row.valor;
		ELSE
			   resultado := null;
		END IF;
	RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT *
FROM exer3(1,10)
```

5. Stored Procedure que mostra todas ou as informações selecionadas da tabela “contas”. As tabelas são selecionadas através do ID do funcionário que as pagou.

```
CREATE OR REPLACE FUNCTION contasfunc (funcionario INTEGER)
	RETURNS SETOF contas
	AS $$
		SELECT *
		FROM contas
		WHERE funcionarios_id = $1
	$$ LANGUAGE sql;

SELECT *
FROM contasfunc (1);
```

6. Stored Procedure que recebe o ID de dois fornecedores. Após, mostra todos os valores no intervalo demarcado entre as duas IDs.

```
CREATE OR REPLACE FUNCTION exer6 (INT,INT)
	RETURNS SETOF fornecedores
	AS $$
	DECLARE
		selectforn RECORD;
```

```
        id1 ALIAS FOR $1;
        id2 ALIAS FOR $2;
    BEGIN
        FOR selectforn IN (SELECT *
                           FROM fornecedores
                           WHERE id BETWEEN $1 AND $2 ) LOOP
            RETURN NEXT selectforn;
        END LOOP;
    END;
    $$LANGUAGE plpgsql;

SELECT id, CNPJ, tipo_produto
FROM exer6(1,4);
```

## Query para as Triggers:

1. Trigger que ocorre antes de um Insert. Esta trigger foi criada para selecionar um adicionar um “gerente_id” a um registro de funcionário caso o mesmo não possua um “gerente_id” registrado.

```
CREATE OR REPLACE FUNCTION escolhergerente()
    RETURNS TRIGGER
    AS $$
    DECLARE
        gerente INTEGER;
    BEGIN
        gerente := CAST (RANDOM()*1+1 AS INTEGER);
        IF (NEW.gerente_id IS NULL) THEN
                NEW.gerente_id = gerente;
                RETURN NEW;
            ELSE
                RETURN NULL;
        END IF;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER escolhergerente
BEFORE INSERT ON funcionarios
FOR EACH ROW
EXECUTE PROCEDURE escolhergerente();

INSERT INTO funcionarios
(id,funcoes_id,endereços_id,endereços_natal_id,telefones_id,nome,sobrenome,data_nascimento,em
ail,RG,CPF,carteira_trabalho,data_admissao,salario)
VALUES
(17,7,1,1,1,'teste','teste','19980115','teste@email.com','1','1','1','20170602',500);
```

2. Trigger que ocorre antes de um Update. Esta trigger foi criada para garantir que não haja a troca de ID dos estados da tabela “estados”.

```
CREATE OR REPLACE FUNCTION mudarestado()
	RETURNS TRIGGER
	AS $$
	BEGIN
		IF (NEW.id <> OLD.id) THEN
			RAISE NOTICE 'Não é possível trocar o id dos estados!';
			RETURN NULL;
		ELSE
			RETURN NEW;
		END IF;
	END;
	$$ LANGUAGE plpgsql;

CREATE TRIGGER mudarestado
BEFORE UPDATE ON estados
FOR EACH ROW
EXECUTE PROCEDURE mudarestado();

UPDATE estados
SET id = 8
WHERE id = 5;
```

3. Trigger que ocorre antes de um Delete. A trigger não permite a exclusão de um gerente (funcionário o qual não possui um gerente_id).

```
CREATE OR REPLACE FUNCTION checkGerente()
	RETURNS TRIGGER
	AS $$
	BEGIN
		IF OLD.gerente_id IS NULL THEN
			RAISE NOTICE 'Não é possível deletar o gerente';
			RETURN NULL;
		ELSE
			RETURN NEW;
		END IF;
	END;
	$$ LANGUAGE plpgsql;

CREATE TRIGGER checkGerente
BEFORE DELETE ON funcionarios
FOR EACH ROW
EXECUTE PROCEDURE checkGerente();

DELETE FROM funcionarios
WHERE id = 1;
```

4. Trigger que ocorre antes de um Insert ou Update . A trigger permite apenas o funcionário se tornar um gerente (trabalhar como “Gerente” ou “Chef”) se o mesmo não tiver um gerente.

```
CREATE OR REPLACE FUNCTION funcgerente()
	RETURNS TRIGGER
	AS $$
	BEGIN
		IF (NEW.gerente_id IS NOT NULL) AND ((NEW.funcoes_id = 1) OR
							(NEW.funcoes_id = 5)) THEN
			RAISE NOTICE 'O funcionario não pode receber a função de um gerente!';
			RETURN NULL;
		ELSE
			RETURN NEW;
		END IF;
	END;
	$$ LANGUAGE plpgsql;

CREATE TRIGGER funcgerente
BEFORE INSERT OR UPDATE ON funcionarios
FOR EACH ROW
EXECUTE PROCEDURE funcgerente();

UPDATE funcionarios
SET funcoes_id = 1
WHERE id=18;

INSERT INTO funcionarios
(id,funcoes_id,endereços_id,endereços_natal_id,telefones_id,nome,sobrenome,data_nascimento,em
ail,RG,CPF,carteira_trabalho,data_admissao,salario)
VALUES
(19,1,1,1,1,'teste2','teste2','19980115','teste2@email.com','1','1','1','20170602',500);
```