Documentação Funções utilizadas

Novo Administrador

- **Descrição:** Criar um novo administrador
- **Argumentos:**
  - `"username_admin":` string do nome do Administrador
  - `"email_admin":` string do email do Administrador
  - `"password_admin":` string da password do Administrador
  - `"nome_Restaurante":` string do nome do restaurante a qual o Administrador pertence
- **Função:**

```
CREATE OR REPLACE FUNCTION createAdministrador(username_admin text,email_admin text,
password_admin text, nome_Restaurante text) returns boolean AS $$
DECLARE IDRestaurante_variable int;
DECLARE ExistsEmail int;
DECLARE ExistsUsername int;
BEGIN
SELECT COUNT (*) from Administrador where email_administrador= email_admin into
ExistsEmail;
IF(ExistsEmail > 0) THEN
   RETURN FALSE;
END IF;
SELECT COUNT (*) from Administrador where username_administrador= username_admin into
ExistsUsername;
IF(ExistsUsername > 0) THEN
   RETURN FALSE;
END IF;
select getIdRestaurante(nome_Restaurante) into IDRestaurante_variable;
IF(IDRestaurante_variable = 0) THEN
    RETURN FALSE;
END IF;
insert into Administrador(id_restaurante,username_administrador, email_administrador,
password_administrador)
   values ( IDRestaurante_variable,username_admin, email_admin, password_admin);
RETURN TRUE;
END;
$$ LANGUAGE plpgsql;
```

Verificar o Email

- **Descrição:** Verificar se o email do administrador é valido
- **Argumentos:**
  - `"email_admin":` string do email do administrador
- **Função:**

```
CREATE OR REPLACE FUNCTION VerifyEmail(email_admin text) returns boolean AS $$
DECLARE ExistsEmail int;
BEGIN
SELECT COUNT (*) from Administrador where email_administrador= email_admin into
```

```
ExistsEmail;
IF(ExistsEmail > 0) THEN
   RETURN TRUE;
ELSE
    RETURN FALSE;
END IF;
END;
$$ LANGUAGE plpgsql;
```

Verificar o Username

- **Descrição:** Verificar se o username do administrador é valido
- **Argumentos:**
  - `"username_admin":` string do username do administrador
- **Função:**

```
CREATE OR REPLACE FUNCTION VerifyUsername(username_admin text) returns boolean AS $$
DECLARE ExistsUsername int;
BEGIN
SELECT COUNT (*) from Administrador where username_administrador= username_admin into
ExistsUsername;
IF(ExistsUsername > 0) THEN
   RETURN TRUE;
ELSE
    RETURN FALSE;
END IF;
END;
$$ LANGUAGE plpgsql;
```

LOGIN

Verificar o Login

- **Descrição:** Verificar se o Login é valido
- **Argumentos:**
  - `"email_login":` string do email do login
  - `"password_login":` string da password do login
- **Função:**

```
CREATE OR REPLACE FUNCTION VerificarLogin(email_login text,password_login text)
returns boolean AS $$
BEGIN
if((select count(*) from Administrador where Administrador.email_administrador =
email_login and Administrador.password_administrador = password_login) <1) then
RETURN FALSE;
else
RETURN TRUE;
end if;
END;
$$ LANGUAGE plpgsql;
```

TRANSAÇÕES

Mostrar todas as transações de um Restaurante

- **Descrição:** Mostrar todas as transações do Restaurante que queremos visualizar
- **Argumentos:**
  - `"email_admin":` string do email do Administrador
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllTransacoes(EmailAdmin text) RETURNS TABLE (
      ID_TRANSACAO INT,
      NOME_CLIENTE text,
      VALOR_TRANSACAO int,
      DATA_TRANSACAO TIMESTAMP WITH TIME ZONE

)  AS $$
DECLARE IDRestaurante int;
DECLARE IDAdmin int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    return query
    select
transacao.id_transacao,cliente.nome_cliente,transacao.valor_transacao,transacao.data_tra
 from transacao join consumo on transacao.id_consumo = consumo.id_consumo join cliente
on consumo.id_cliente = cliente.id_cliente where transacao.id_restaurante =
IDRestaurante;
END;
$$ LANGUAGE plpgsql;
```

Criar uma transação

- **Descrição:** Criar uma transação
- **Argumentos:**
  - `"email_admin":` string do email do Administrador
- **Função:**

```
CREATE OR REPLACE FUNCTION createTransacao(EmailAdmin text) RETURNS boolean
AS $$
DECLARE IDRestaurante int;
DECLARE IDAdmin int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    insert into Transacao (id_restaurante, id_consumo, valor_transacao,
data_transacao) values (1, 1, 1, '7/16/2019');

END;
$$ LANGUAGE plpgsql;
```

- **Descrição:** Apagar uma transação
- **Argumentos:**
  - `"IDTransacao":` int do ID da Transação
- **Função:**

```
CREATE OR REPLACE FUNCTION DeleteTransacao(IDTransacao int) RETURNS BOOLEAN
AS $$
BEGIN
    DELETE FROM transacao where transacao.id_transacao = IDTransacao;
    RETURN TRUE;
END;
$$ LANGUAGE plpgsql;
```

Mostrar top 5 Clientes com mais transações

- **Descrição:** Mostrar top 5 Clientes com maior número de transações
- **Função:**

```
CREATE OR REPLACE FUNCTION getTopTransacoes_XML() RETURNS XML AS $BODY$
DECLARE output XML;
BEGIN
    DROP VIEW if exists TopTransacoesClientes;
    CREATE VIEW TopTransacoesClientes AS select
cliente.nome_cliente,count(transacao.id_transacao) as numero_transacoes
    from cliente join cartao on cliente.id_cliente = cartao.id_cliente join consumo on
cartao.id_cartao = consumo.id_cartao join transacao on transacao.id_consumo =
consumo.id_consumo
    group by cliente.nome_cliente
    order by numero_transacoes desc
    limit 5;
     SELECT query_to_xml('select * from TopTransacoesClientes', true, false, '') INTO
output;
    RETURN output;
END;
$BODY$
LANGUAGE plpgsql
```

Clientes

Mostrar todos Clientes

- **Descrição:** Mostrar todos os Clientes existentes
- **Função:**

```
CREATE OR REPLACE FUNCTION getallClientes()
  RETURNS SETOF public.cliente AS $BODY$
BEGIN
  return query--<< this was missing
    SELECT *
    FROM cliente;
```

```
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar Cliente específico

- **Descrição:** Mostrar cliente específico
- **Argumentos:**
  - `"idc":` inteiro do id do Cliente
- **Função:**

```
CREATE OR REPLACE FUNCTION getClientes(IN idc int)
 RETURNS TABLE (
      ID_CLIENTE INT,
      NOME_CLIENTE text,
      DATA_CARTAO TIMESTAMP
)
BEGIN
  return query--<< this was missing
    select cliente.ID_CLIENTE,cliente.NOME_CLIENTE,cartao.DATA_CRIACAO_CARTAO from
cliente join cartao on cartao.id_cliente = cliente.id_cliente where cartao.id_cliente
= idc;

END;
```

Mostrar Cliente de um restaurante

- **Descrição:** Mostrar clientes de um dado restaurante
- **Argumentos:**
  - `"EmailAdmin":` string do email do Admin
- **Função:**

```
CREATE OR REPLACE FUNCTION getallClientesFromRestaurante(EmailAdmin text) RETURNS
TABLE (
      ID_CLIENTE INT,
      NOME_CLIENTE text,
      DATA_CARTAO TIMESTAMP
)  AS $$
DECLARE IDRestaurante int;
DECLARE IDAdmin int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    return query
    select cliente.ID_CLIENTE,cliente.NOME_CLIENTE,cartao.DATA_CRIACAO_CARTAO from
cliente join cartao on cartao.id_cliente = cliente.id_cliente where
cartao.id_restaurante = IDRestaurante;
END;
$$ LANGUAGE plpgsql;
```

Create na tabela Clientes

- **Descrição:** Criação de um novo cliente
- **Argumentos:**
  - `"nome_cliente":` string do nome do Cliente
  - `"emailAdmin":` string do email do Admin (de modo a referenciar o restaurante)
- **Função:**

```
CREATE OR REPLACE FUNCTION createCliente(nome_cliente text,emailAdmin text)
  RETURNS BOOLEAN AS $BODY$
  DECLARE idCliente int;
  DECLARE idRestaurante int;
  DECLARE IDAdmin int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
  insert into Cliente (nome_cliente)
  values (nome_cliente) RETURNING id_cliente INTO idCliente;

  insert into Cartao (id_restaurante,id_cliente)
  values (idRestaurante,idCliente);
  RETURN TRUE;

END;
$BODY$
LANGUAGE plpgsql
```

Update na tabela Clientes

- **Descrição:** Fazer mudanças em todos os campos da tabela Clientes
- **Argumentos:**
  - `"idc":` inteiro do id do Cliente
  - `"novo_nome":` string do id do Cliente
- **Função:**

```
CREATE OR REPLACE FUNCTION updateCliente(IN idc int, novo_nome text)
  RETURNS SETOF public.cliente AS $BODY$
BEGIN
    UPDATE cliente SET nome_cliente = novo_nome
    WHERE cliente.id_cliente = idc;

END;
$BODY$
LANGUAGE plpgsql
```

Eliminar Cliente

- **Descrição:** Eliminar um Cliente
- **Argumentos:**
  - `"idc":` inteiro do id do Cliente

- **Função:**

```
CREATE OR REPLACE FUNCTION deleteCliente(IN idc int)
  RETURNS SETOF public.cliente AS $BODY$
BEGIN
  --return query--<< this was missing
    DELETE from cliente
    WHERE cliente.id_cliente = idc;

END;
$BODY$
LANGUAGE plpgsql
```

Desfidelizar um cliente de um restaurante

- **Descrição:** Desfidelizar um cliente de um dado restaurante, ou seja apagar o cartão do mesmo
- **Argumentos:**
  - `"EmailAdmin":` string do email do Admin
  - `"idCliente":` int do ID do Cliente
- **Função:**

```
CREATE OR REPLACE FUNCTION deleteCartao(EmailAdmin text,idCliente int)
  RETURNS BOOLEAN AS $BODY$
  DECLARE idRestaurante int;
  DECLARE IDAdmin int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    DELETE from cartao
    WHERE cartao.id_restaurante=IDRestaurante and cartao.id_cliente = idCliente;
    RETURN TRUE;
END;
$BODY$
LANGUAGE plpgsql
```

Ementa

Mostrar todas as Ementas

- **Descrição:** Mostra todas as ementas por dia de semana de um dado restaurante
- **Argumentos:**
  - `"EmailAdmin":` string do email do administrador
  - `"DiaDaSemana":` inteiro do dia da semana
- **Função:**

```
CREATE OR REPLACE FUNCTION getallEmentas(EmailAdmin text,DiaDaSemana int)
  RETURNS TABLE (
      Dia_da_Semana int,
      Prato_de_Carne text,
```

```
      Prato_de_Peixe text,
      Entrada text,
      Bebida text,
      Sobremesa text
)  AS $BODY$
      DECLARE IDAdmin int;
      DECLARE IDRestaurante int;
      DECLARE IDEmenta int;
      DECLARE _Dia_da_Semana int;
      DECLARE _Prato_de_Carne text;
      DECLARE _Prato_de_Peixe text;
      DECLARE _Entrada text;
      DECLARE _Bebida text;
      DECLARE _Sobremesa text;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    select ementa.id_ementa into IDEmenta from ementa where ementa.id_restaurante =
IDRestaurante and ementa.dia_da_semana =DiaDaSemana;
    select getPratoCarne_ementa(IDEmenta) into _Prato_de_Carne;
    select getPratoPeixe_ementa(IDEmenta) into _Prato_de_Peixe;
    select getEntrada_ementa(IDEmenta) into _Entrada;
    select getBebida_ementa(IDEmenta) into _Bebida;
    select getSobremesa_ementa(IDEmenta) into _Sobremesa;
    DROP TABLE IF EXISTS tbl;
    CREATE TEMP TABLE tbl AS SELECT
DiaDaSemana,_Prato_de_Carne,_Prato_de_Peixe,_Entrada,_Bebida ,_Sobremesa;
    RETURN QUERY SELECT * FROM tbl;

END;
$BODY$
LANGUAGE plpgsql
```

Mostrar uma Ementa

- **Descrição:** Mostra informação de uma dada ementa
- **Argumentos:**
  - `"ide":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getEmentas(IN ide int)
  RETURNS SETOF public.ementa AS $BODY$
BEGIN
  return query--<< this was missing
    SELECT *
    FROM ementa
    WHERE ementa.id_ementa = ide;

END;
$BODY$
LANGUAGE plpgsql
```

Mostra pratos de carne de uma ementa

- **Descrição:** Mostra pratos de carne de uma dada ementa
- **Argumentos:**
  - `"IDEmenta":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getPratoCarne_ementa(IDEmenta int)
RETURNS text  AS $BODY$
DECLARE NomeProduto text;
BEGIN
  select produto.nome_produto from produto_na_ementa join produto on
produto_na_ementa.id_produto = produto.id_produto
  where produto_na_ementa.id_ementa = IDEmenta and produto.tipo_produto = 3 into
NomeProduto;
  return NomeProduto;

END;
$BODY$
LANGUAGE plpgsql
```

Mostra pratos de peixe de uma ementa

- **Descrição:** Mostra pratos de peixe de uma dada ementa
- **Argumentos:**
  - `"IDEmenta":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getPratoPeixe_ementa(IDEmenta int)
RETURNS text  AS $BODY$
DECLARE NomeProduto text;
BEGIN
  select produto.nome_produto from produto_na_ementa join produto on
produto_na_ementa.id_produto = produto.id_produto
  where produto_na_ementa.id_ementa = IDEmenta and produto.tipo_produto = 4 into
NomeProduto;
  return NomeProduto;

END;
$BODY$
LANGUAGE plpgsql
```

Mostra entradas de uma ementa

- **Descrição:** Mostra entradas de uma dada ementa
- **Argumentos:**
  - `"IDEmenta":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getEntrada_ementa(IDEmenta int)
RETURNS text  AS $BODY$
DECLARE NomeProduto text;
BEGIN
  select produto.nome_produto from produto_na_ementa join produto on
produto_na_ementa.id_produto = produto.id_produto
  where produto_na_ementa.id_ementa = IDEmenta and produto.tipo_produto = 0 into
NomeProduto;
  return NomeProduto;

END;
$BODY$
LANGUAGE plpgsql
```

Mostra bebidas de uma ementa

- **Descrição:** Mostra bebidas de uma dada ementa
- **Argumentos:**
  - `"IDEmenta":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getBebida_ementa(IDEmenta int)
RETURNS text  AS $BODY$
DECLARE NomeProduto text;
BEGIN
  select produto.nome_produto from produto_na_ementa join produto on
produto_na_ementa.id_produto = produto.id_produto
  where produto_na_ementa.id_ementa = IDEmenta and produto.tipo_produto = 1 into
NomeProduto;
  return NomeProduto;

END;
$BODY$
LANGUAGE plpgsql
```

Mostra sobremesas de uma ementa

- **Descrição:** Mostra sobremesas de uma dada ementa
- **Argumentos:**
  - `"IDEmenta":` int do id da ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getSobremesa_ementa(IDEmenta int)
RETURNS text  AS $BODY$
DECLARE NomeProduto text;
BEGIN
  select produto.nome_produto from produto_na_ementa join produto on
produto_na_ementa.id_produto = produto.id_produto
  where produto_na_ementa.id_ementa = IDEmenta and produto.tipo_produto =2 into
NomeProduto;
  return NomeProduto;
```

```
END;
$BODY$
LANGUAGE plpgsql
```

Criação de uma ementa

- **Descrição:** Criar uma ementa associada a um dia da semana e a um restaurante
- **Argumentos:**
  - `"EmailAdmin":` string do email do administrador
  - `"DiaSemana":` inteiro do dia da semana
  - `"carne":` inteiro do id do prato de carne
  - `"peixe":` inteiro do id do prato de peixe
  - `"entrada":` string do id da entrada
  - `"bebida":` inteiro do id da bebida
  - `"sobremesa":` inteiro do id da sobremesa
- **Função:**

```
CREATE OR REPLACE FUNCTION createEmenta(EmailAdmin text,DiaSemana int,carne int,peixe
int,entrada int,bebida int,sobremesa int) RETURNS boolean
AS $$
DECLARE IDRestaurante int;
DECLARE IDAdmin int;
DECLARE IDEmenta int;
BEGIN
    IDAdmin = getIDAdminWithEmail(EmailAdmin);
    IDRestaurante = getIDRestauranteComIDAdministrador(IDAdmin);
    insert into Ementa (dia_da_semana, id_restaurante) values
(DiaSemana,IDRestaurante) RETURNING id_ementa INTO IDEmenta;
    insert into produto_na_ementa (id_ementa, id_produto) values (IDEmenta, carne);
    insert into produto_na_ementa (id_ementa, id_produto) values (IDEmenta, peixe);
    insert into produto_na_ementa (id_ementa, id_produto) values (IDEmenta, entrada);
    insert into produto_na_ementa (id_ementa, id_produto) values (IDEmenta, bebida);
    insert into produto_na_ementa (id_ementa, id_produto) values (IDEmenta,
sobremesa);
    RETURN TRUE;

END;
$$ LANGUAGE plpgsql;
```

Trigger Remoção ementa da semana passada

- **Descrição:** Trigger executado cada vez que uma nova ementa é criada e que elimina a ementa antiga
- **Trigger:**

```
CREATE TRIGGER RemoverEmentaSemanaPassada_Trigger
  AFTER INSERT
  ON ementa
  FOR EACH ROW
  EXECUTE PROCEDURE RemoverEmentaSemanaPassada();
```

```
CREATE OR REPLACE FUNCTION RemoverEmentaSemanaPassada()
  RETURNS trigger AS
$BODY$
BEGIN
   IF EXISTS (SELECT 1 FROM ementa WHERE ementa.dia_da_semana = new.dia_da_semana and
ementa.id_restaurante = new.id_restaurante and ementa.id_ementa != new.id_ementa) THEN
       DELETE FROM ementa where ementa.dia_da_semana = new.dia_da_semana and
ementa.id_restaurante = new.id_restaurante and ementa.id_ementa != new.id_ementa;
   END IF;

   RETURN NULL;
END;
$BODY$ LANGUAGE plpgsql;
```

Update na tabela Ementa

- **Descrição:** Fazer mudanças em todos os campos da tabela Ementa
- **Argumentos:**
  - `"ide":` inteiro do id do Cliente
  - `"novo_diasemana":` string do id do Cliente
  - `"idr":` inteiro do id do Cliente
- **Função:**

```
CREATE OR REPLACE FUNCTION updateEmenta(IN ide int, novo_diasemana int, idr int)
  RETURNS SETOF public.ementa AS $BODY$
BEGIN
  --return query--<< this was missing
    UPDATE ementa SET dia_da_semana = novo_diasemana, id_restaurante = idr
    WHERE ementa.id_ementa = ide;

END;
$BODY$
LANGUAGE plpgsql
```

Eliminar Ementa

- **Descrição:** Eliminar uma Ementa
- **Argumentos:**
  - `"idc":` inteiro do id do Cliente
- **Função:**

```
CREATE OR REPLACE FUNCTION deleteEmentas(IN ide int)
  RETURNS SETOF public.ementa AS $BODY$
BEGIN
  --return query--<< this was missing
    DELETE from ementa
    WHERE ementa.id_ementa = ide;

END;
$BODY$
LANGUAGE plpgsql
```

Produtos

Mostrar todos Produtos

- **Descrição:** Mostrar todos os Produtos da tabela Produtos
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllProdutos() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar produtos de uma ementa

- **Descrição:** Mostrar todos os produtos numa dada ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getProdutos(IDEmenta int) RETURNS SETOF public.produto AS
$BODY$
BEGIN
  return query
    SELECT *
    FROM produto join produto_na_ementa on produto.ID_PRODUTO =
produto_na_ementa.ID_PRODUTO
    join ementa on produto_na_ementa.ID_EMENTA = ementa.ID_EMENTA
    WHERE ementa.ID_ementa = IDEmenta;
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar todas Entradas

- **Descrição:** Mostrar todas as Entradas que existem
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllEntradas() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 0;
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar todas Bebidas

- **Descrição:** Mostrar todas as Entradas que existem
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllBebidas() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 1;
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar todas Sobremesas

- **Descrição:** Mostrar todas as Sobremesas que existem na Ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllSobremesas() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 2;
END;
$BODY$
LANGUAGE plpgsql
```

Mostrar todos Pratos de Carne

- **Descrição:** Mostrar todos os tipos de Prato de Carne que existem na Ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllCarne() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 3;
END;
$BODY$
LANGUAGE plpgsqlCREATE OR REPLACE FUNCTION getAllEntradas() RETURNS SETOF
public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 0;
END;
```

```
$BODY$
LANGUAGE plpgsql
```

Mostrar todos Pratos de Peixe

- **Descrição:** Mostrar todos os tipos de Prato de Peixe que existem na Ementa
- **Função:**

```
CREATE OR REPLACE FUNCTION getAllPeixe() RETURNS SETOF public.produto AS $BODY$
BEGIN
  return query
    SELECT *
    FROM produto
    WHERE produto.tipo_produto = 4;
END;
$BODY$
LANGUAGE plpgsql
```

Update na tabela Produto

- **Descrição:** Fazer mudanças em todos os campos da tabela Produto
- **Argumentos:**
  - `"id":` inteiro do id do Produto
  - `"novo_nomeproduto":` string do nome do Produto
  - `"novo_desig":` string da designação do Produto
  - `"novo_preco":` dinheiro do preço do Produto
  - `"nova_alergia":` string da alergia do Produto
  - `"nova_quantidade":` inteiro da quanitdade do Produto
- **Função:**

```
CREATE OR REPLACE FUNCTION updateProduto(id int, novo_nomeproduto text, novo_desig
text, novo_preco money, nova_alergia text, nova_quantidade int)
  RETURNS boolean AS $BODY$
BEGIN
    UPDATE produto SET nome_produto = novo_nomeproduto, designacao_produto =
novo_desig, preco_produto = novo_preco, alergia_produto = nova_alergia,
quantidade_produto = nova_quantidade
    WHERE produto.id_produto = id;
    RETURN TRUE;

END;
$BODY$
LANGUAGE plpgsql
```

Eliminar Produto

- **Descrição:** Eliminar um Produto
- **Argumentos:**
  - `"id":` inteiro do id do Prdouto
- **Função:**

```
CREATE OR REPLACE FUNCTION deleteProduto(id int)
  RETURNS boolean AS $BODY$
BEGIN

    DELETE from produto
    WHERE produto.id_produto = id;
    return TRUE;
END;
$BODY$
LANGUAGE plpgsql
```

Novo Produto

- **Descrição:** Criar um novo administrador
- **Argumentos:**
  - `"tipo_produto ":` int do tipo do Produto
  - `"nome_produto ":` string do nome do Produto
  - `"designacao_produto":` string da designação do Produto
  - `"preco_produto":` int do preço do Produto
  - `"alergia_produto":` string da alergia do Produto
  - `"quantidade_produto":` string do quantidade do Produto
- **Função:**

```
CREATE OR REPLACE FUNCTION createProduto(tipo_produto int, nome_produto text,
designacao_produto text, preco_produto int, alergia_produto text, quantidade_produto
text)
  RETURNS boolean AS $BODY$
BEGIN

    INSERT INTO Produto(tipo_produto, nome_produto, designacao_produto, preco_produto,
alergia_produto, quantidade_produto)
    Values(tipo_produto, nome_produto, designacao_produto, preco_produto,
alergia_produto, quantidade_produto);
    return TRUE;
END;
$BODY$
LANGUAGE plpgsql
$$ LANGUAGE plpgsql;
```

Funções Adicionais

Nome do Restaurante pelo ID

- **Descrição:** Mostrar o nome do Restaurante pelo seu ID
- **Argumentos:**
  - `nome_do_Restaurante:` string do nome do Restaurante

- **Função:**

```
CREATE OR REPLACE FUNCTION getIdRestaurante(nome_do_Restaurante text) returns int AS
$$
DECLARE ID_Restaurante_return int;
BEGIN
select Restaurante.ID_Restaurante from Restaurante where Restaurante.nome_restaurante
= nome_do_Restaurante
into ID_Restaurante_return;
IF    ID_Restaurante_return THEN
return ID_Restaurante_return;
ELSE
return 0;
END IF;
END;
$$ LANGUAGE plpgsql;
```

ID do Restaurante pelo ID Administrador

- **Descrição:** Mostrar o ID do Restaurante através do ID do Administrador desse mesmo restaurante
- **Argumentos:**
  - `IDAdmin:` int do ID do Restaurante
- **Função:**

```
CREATE OR REPLACE FUNCTION getIDRestauranteComIDAdministrador(IDAdmin int) returns int
AS $$
DECLARE
returnNumeroRestaurante int;

BEGIN
    returnNumeroRestaurante := 0;

    select Restaurante.ID_Restaurante
    from Administrador inner join Restaurante on
    Administrador.ID_Restaurante = Restaurante.ID_Restaurante
     where Administrador.ID_Administrador = IDAdmin
    into returnNumeroRestaurante;

    IF returnNumeroRestaurante THEN
        return returnNumeroRestaurante;
    ELSE
        return 0;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Nome de todos os Restaurantes

- **Descrição:** Mostrar todos os nomes dos Restaurante que se econtram na Base de Dados
- **Função:**

```
CREATE OR REPLACE FUNCTION getNomeRestaurantes() returns text[] AS $$
DECLARE
returnNomeRestaurantes text[];
BEGIN
    select array( select Nome_Restaurante from Restaurante) into
returnNomeRestaurantes;
    return returnNomeRestaurantes;
END;
$$ LANGUAGE plpgsql;
```