

Gestão de uma Pizzaria e Sistema de Encomendas

Trabalho Prático de Base de Dados

Realizado por P6 G5:

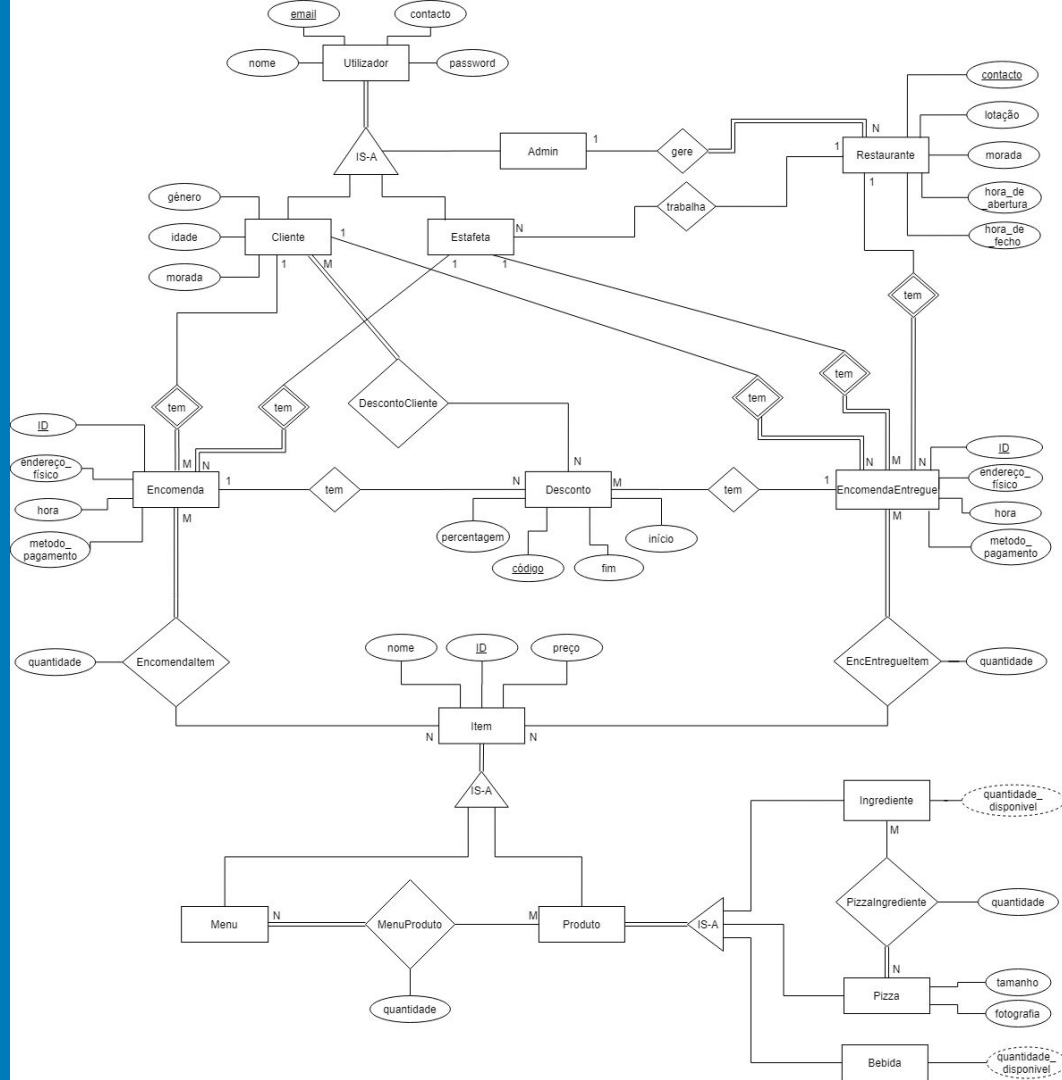
Leandro Silva 93446

Mário Silva 93430



DER

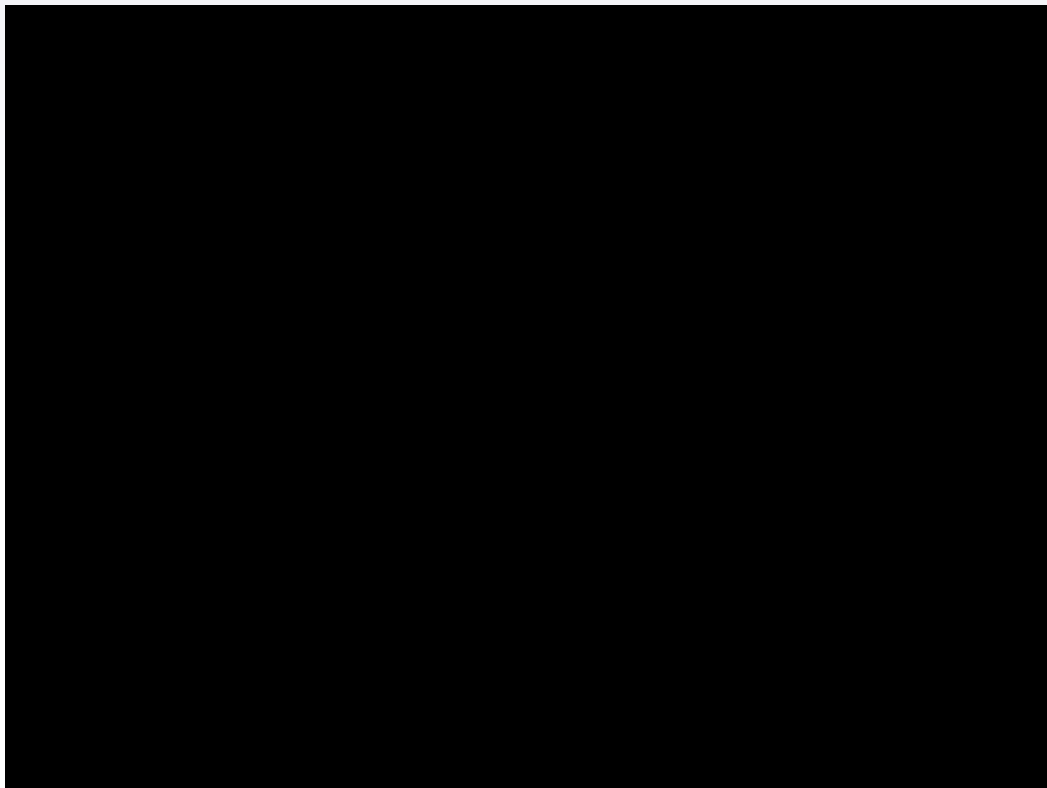
Desenho relacional do modelo da nossa base de dados.



► Funcionalidades Principais

- ▶ O cliente pode ver e filtrar uma lista de itens disponíveis a encomendar.
- ▶ O cliente pode fazer e pedir uma encomenda, podendo usar descontos promocionais.
- ▶ O estafeta pode ver e controlar as encomendas e deve dar um feedback sobre o sucesso da entrega da encomenda.
- ▶ O administrador tem os meios para gerir a base de dados.
- ▶ O administrador pode ver estatísticas do seu comércio.

▶ Demo



▶ Índices

- ▶ Não foram usados índices para além dos clustered criados por defeito para este contexto.
- ▶ Importante notar que recorreremos à propriedade IDENTITY na chave primária da tabela *Encomenda*, e em mais nenhuma tabela.

Triggers

- ▶ Protegem a consistência e integridade dos dados.
- ▶ Foram usados 3 triggers `instead of delete` e 1 trigger `instead of insert`:
 - ▶ `delDesconto`
 - ▶ `delEncomenda`
 - ▶ `delEstafeta`
 - ▶ `insDescontoCliente`

Cursor

```
update Pizaria.Estafeta set res_contato = null where email=@email;
--cursor para colocar o melhor estafeta em cada uma das encomendas
DECLARE c CURSOR STATIC
FOR select ID from Pizaria.Encomenda where estafeta_email=@email;

OPEN c;
FETCH NEXT FROM c into @ID;

WHILE @@FETCH_STATUS = 0
BEGIN
    update Pizaria.Encomenda set estafeta_email = (select Pizaria.FindBestEstafeta()) where ID=@ID
    FETCH NEXT FROM c into @ID;
END;
CLOSE c;
DEALLOCATE c;
```

Stored Procedures

- ▶ Permitem uma execução mais rápida e podem ser usadas como mecanismos de segurança.
- ▶ Foram usadas as seguintes procedures:
 - ❖ filterItem
 - ❖ insDesconto
 - ❖ insEncomenda
 - ❖ insEstafeta
 - ❖ insRestaurante
 - ❖ insUtilizador
 - ❖ login
 - ❖ register
 - ❖ tranShopCart
 - ❖ updRestaurante
 - ❖ updStock
 - ❖ statsRestaurants

TranShopCart

Trata de todo o processo por detrás da criação da encomenda.

- ▶ Procura o melhor estafeta para entregar a encomenda.
- ▶ Insere o desconto, se existente, no registo do cliente.
- ▶ Registrar a encomenda.
- ▶ Percorre todos os itens da encomenda, verificar a sua disponibilidade no stock e reduzir no stock.



```

declare @estafeta_email nvarchar(255)
set @estafeta_email = Pizaria.findBestestafeta()

begin try
begin tran
    insert into Pizaria.DescontoCliente (cli_email,des_codigo) Values (@cliente_email,@des_codigo)

    declare @last_ID int
    Exec Pizaria.insEncomenda @cliente_email=@cliente_email, @estafeta_email=@estafeta_email, @endereco_fisico=@endereco_fisico,
    @hora=@hora, @metodo_pagamento=@metodo_pagamento, @des_codigo=@des_codigo, @last_ID=@last_ID output

    declare @pos int = 0
    declare @len int = 0
    declare @item_ID int
    declare @quantidade int
    WHILE CHARINDEX(',', @lista, @pos+1)>0
    BEGIN
        set @len = CHARINDEX(',', @lista, @pos+1) - @pos
        set @item_ID = cast(SUBSTRING(@lista, @pos, @len) as int)
        set @pos = CHARINDEX(',', @lista, @pos+@len) +1
        set @len = CHARINDEX(',', @lista, @pos+1) - @pos
        set @quantidade =cast( SUBSTRING(@lista, @pos, @len) as int)
        set @pos = CHARINDEX(',', @lista, @pos+@len) +1

        declare @itemType varchar(15)
        set @itemType=Pizaria.findItemType(@item_ID)
    
```

```

if (@itemType='Menu')
begin
    IF EXISTS (select top 1 men_ID from Pizaria.MenuProduto left outer join Pizaria.Piza on pro_ID=Piza.ID
    left outer join Pizaria.Bebida on Bebida.ID=pro_ID
    left outer join Pizaria.PizaIngrediente on piz_ID=Piza.ID
    left outer join Pizaria.Ingrediente on ing_ID=Ingrediente.ID
    where Ingrediente.quantidade_disponivel - PizaIngrediente.quantidade*MenuProduto.quantidade*@quantidade < 0
    or Bebida.quantidade_disponivel - MenuProduto.quantidade*@quantidade < 0
    and men_ID=@item_ID
    )
begin
    rollback tran
    set @response='Number of Products not Available'
    return
end

update Pizaria.Ingrediente
set quantidade_disponivel = quantidade_disponivel - PizaIngrediente.quantidade*MenuProduto.quantidade*@quantidade
from Pizaria.MenuProduto join Pizaria.Piza on pro_ID=Piza.ID
join Pizaria.PizaIngrediente on piz_ID=Piza.ID
join Pizaria.Ingrediente on ing_ID=Ingrediente.ID
where men_ID=@item_ID

update Pizaria.Bebida
set quantidade_disponivel = quantidade_disponivel - MenuProduto.quantidade*@quantidade
from Pizaria.MenuProduto join Pizaria.Bebida on Bebida.ID=pro_ID
where men_ID=@item_ID
end

```

User Defined Functions

- ▶ Promovem a encapsulação da base de dados.
- ▶ Preferência de UDFs em relação às Views devido a maior eficiência.

Escalares

- ❖ getDesconto
- ❖ getEncPrice
- ❖ isEstafeta
- ❖ isAdmin
- ❖ isEmployed
- ❖ isValidDiscount
- ❖ findBestEstafeta
- ❖ findItemType

Multi-statement Table-valued

- ❖ showLowStock
- ❖ statsEncomenda

User Defined Functions

Inline Table-valued

- | | | |
|---------------------|--------------------|--------------------------|
| ❖ bebidaAvail | ❖ showAllDiscounts | ❖ showAllOrders |
| ❖ ingredienteAvail | ❖ showAllEstafetas | ❖ showOrder |
| ❖ menuAvail | ❖ showEncEntregue | ❖ showOrderClient |
| ❖ pizaAvail | ❖ showEncomenda | ❖ showOrderClientHistory |
| ❖ getEstRestaurante | ❖ showMenu | ❖ showOrderHistory |
| ❖ showRestaurante | ❖ showPiza | ❖ showOrdersToDel |

▶ Aspectos de Segurança

- ▶ Validação de campos de preenchimento na aplicação.
- ▶ Apresentação de mensagens de erro customizadas.
- ▶ Uso de SQL parametrizado.
- ▶ Codificação de dados.
- ▶ Uso de Triggers.
- ▶ Uso de Stored Procedures.
- ▶ Uso de checks na criação das tabelas.



► Conclusão



Este trabalho foi uma mais valia para aumentar o nosso discernimento nos conteúdos lecionados.