Base de Dados

Apresentação de Projeto

Business Manager

P5G4

Bernardo Pinto, 105926
José Mendes, 107188

Introdução

Este projeto tem por base a criação de uma Base de Dados de um sistema de gestão de um website. Além disso foi também utilizado Windows Forms C# (.NET Framework) permitindo criar uma interface simples para uma demonstração da utilização da Base de Dados.

As entidades principais são:

- Manager
- Staff
- Item
- Order
- Transport
- Supplier
- Store

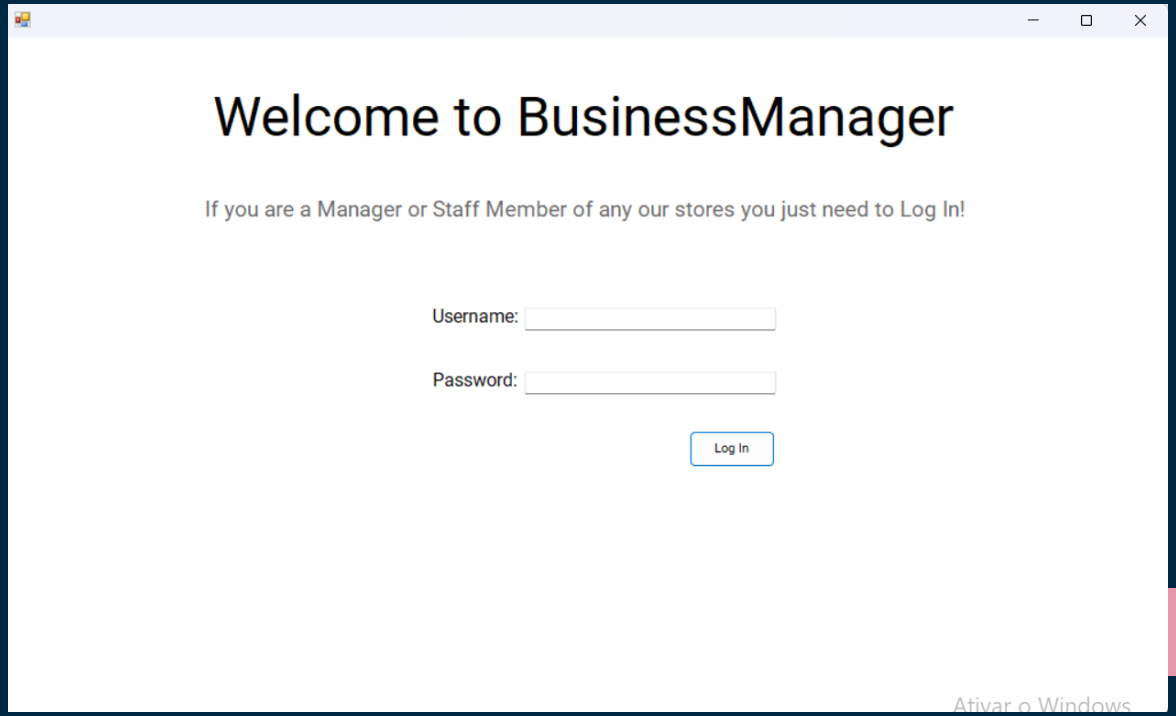
Funcionalidades do Sistema

Gestão de pedidos realizados por um Costumer

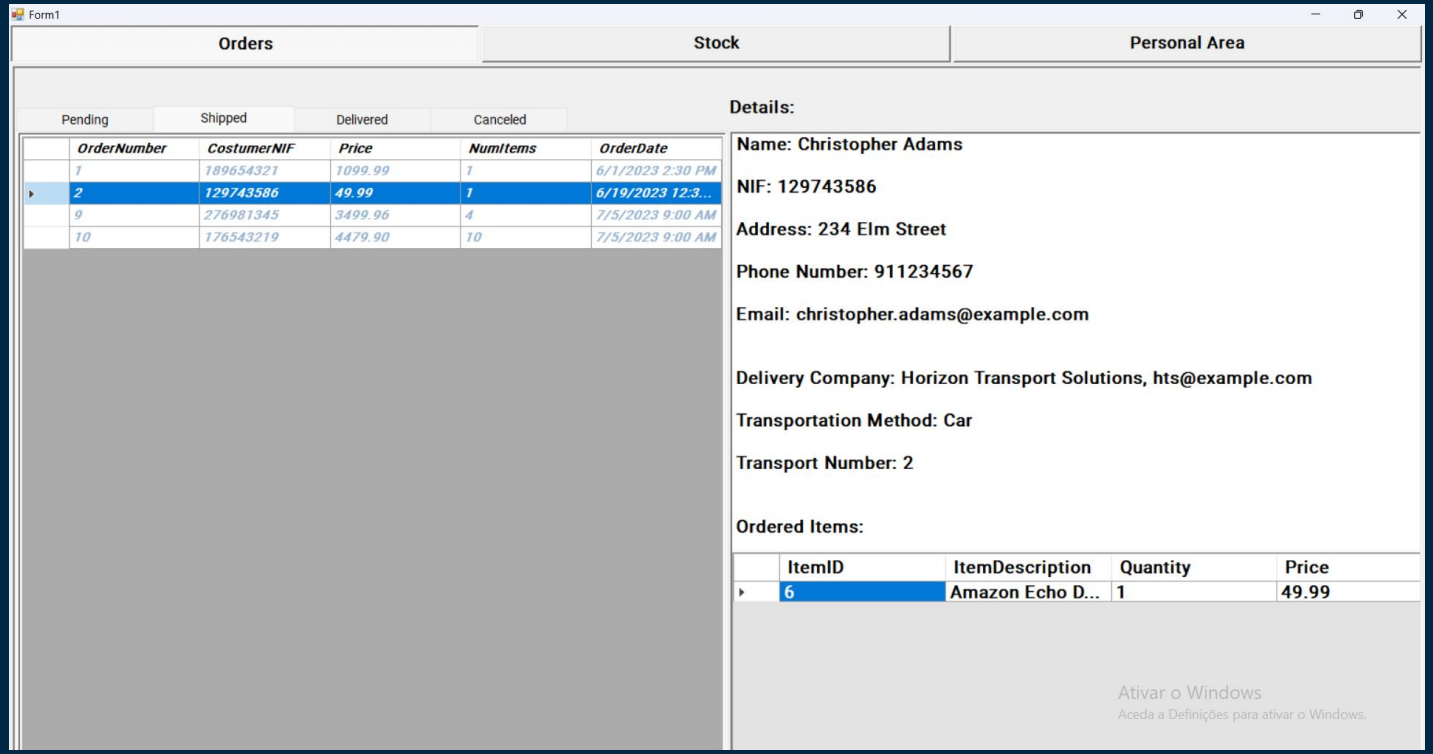
Comunicação com Fornecedores e Transportes

Visualização e gestão do Stock da Loja

Interface Login



Interface Staff



Interface Manager

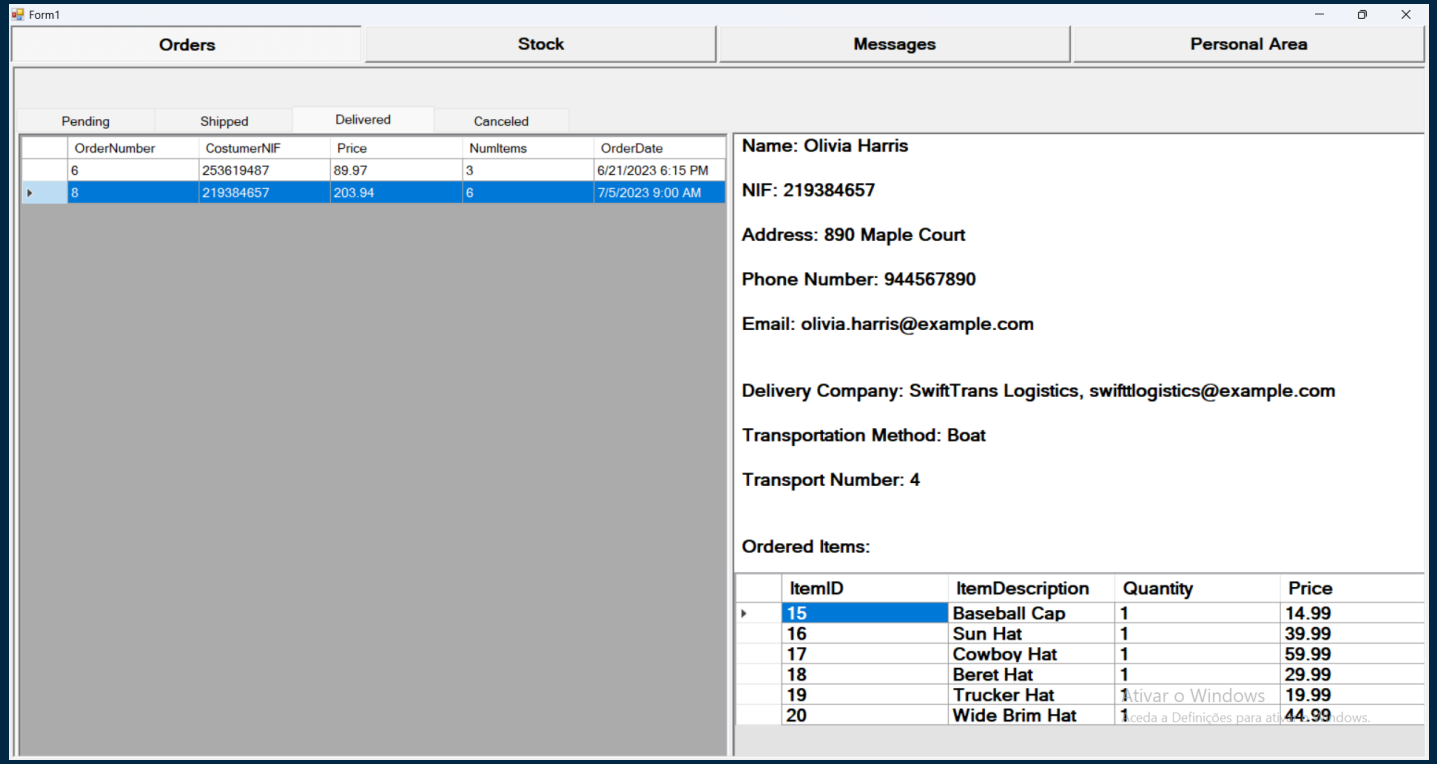
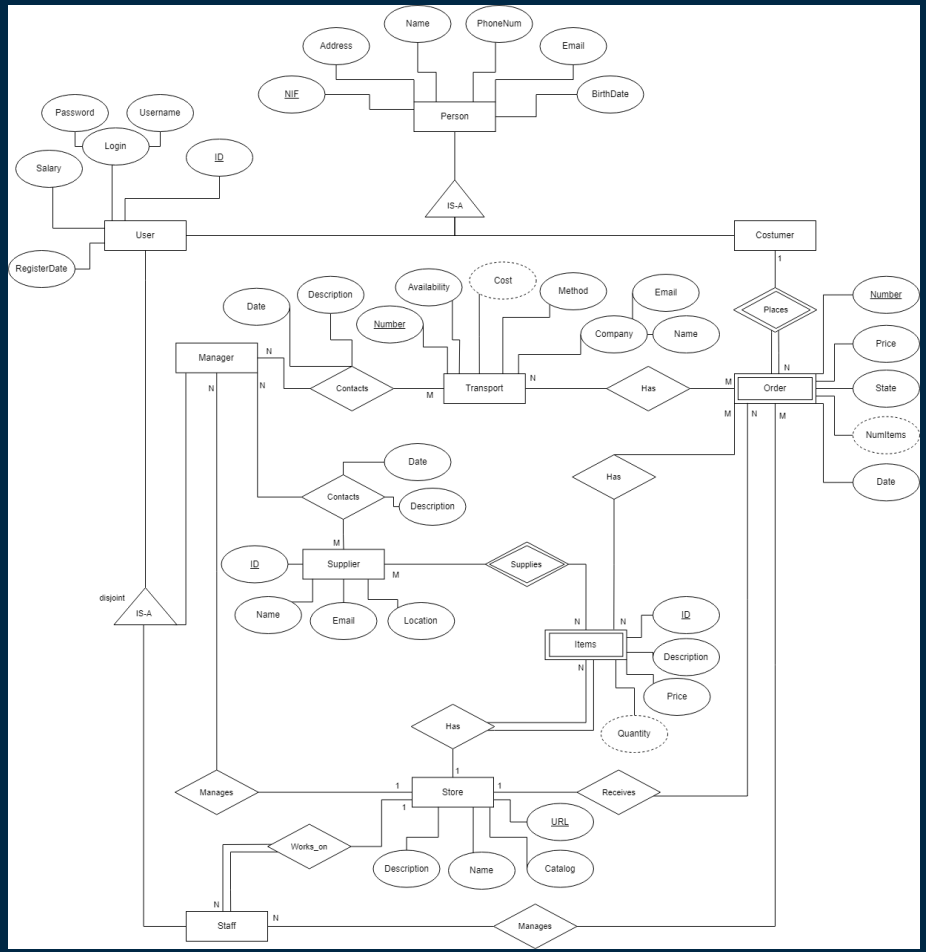
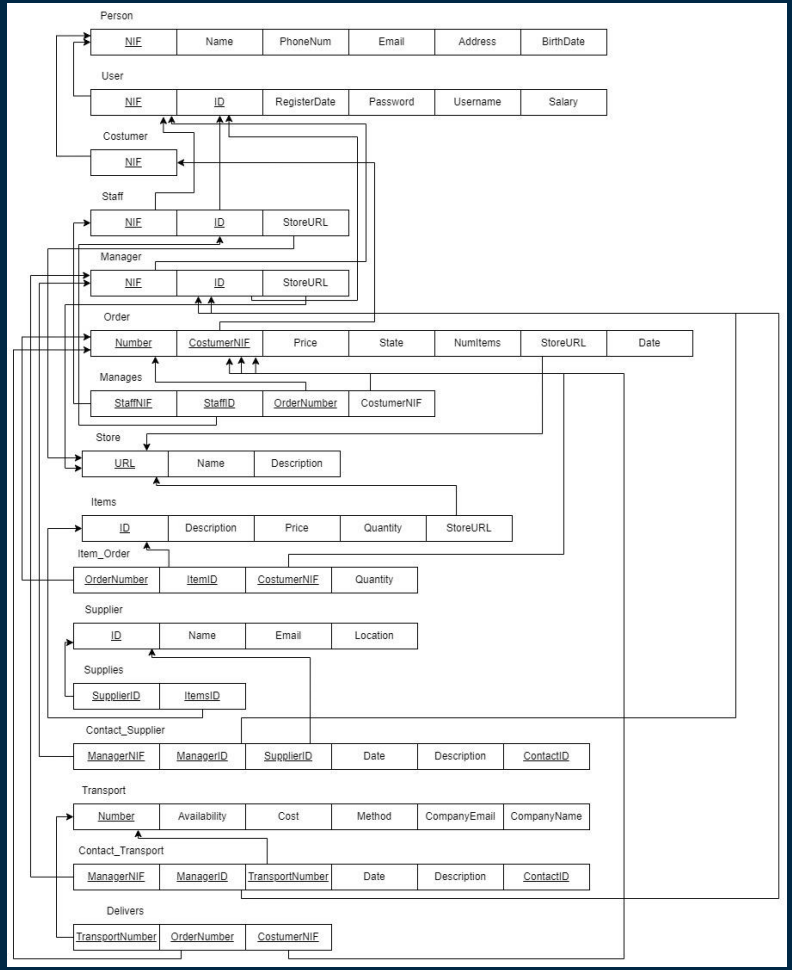


Diagrama ER


Name
PhoneNum
Address
Email
NIF
BirthDate
Person
Password
Username
Login
Salary
ID
IS-A
User
Costumer
RegisterDate
Date
Description
Availability
Cost
Method
Email
Number
Company
Name
Places
Number
Price
State
NumItems
Date
Manager
Contacts
Transport
Has
Order
Date
Contacts
Description
Has
ID
Supplier
Supplies
disjoint
IS-A
Name
Email
Location
ID
Items
Description
Price
Quantity
Has
Manages
Store
Receives
URL
Works_on
Description
Name
Catalog
Staff
Manages

Modelo ER


Person
NIF
Name
PhoneNum
Email
Address
BirthDate
User
NIF
ID
RegisterDate
Password
Username
Salary
Costumer
NIF
Staff
NIF
ID
StoreURL
Manager
NIF
ID
StoreURL
Order
Number
CostumerNIF
Price
State
NumItems
StoreURL
Date
Manages
StaffNIF
StaffID
OrderNumber
CostumerNIF
Store
URL
Name
Description
Items
ID
Description
Price
Quantity
StoreURL
Item_Order
OrderNumber
ItemID
CostumerNIF
Quantity
Supplier
ID
Name
Email
Location
Supplies
SupplierID
ItemsID
Contact_Supplier
ManagerNIF
ManagerID
SupplierID
Date
Description
ContactID
Transport
Number
Availability
Cost
Method
CompanyEmail
CompanyName
Contact_Transport
ManagerNIF
ManagerID
TransportNumber
Date
Description
ContactID
Delivers
TransportNumber
OrderNumber
CostumerNIF

Implementação das funcionalidades

- **Stored Procedures**: Maioritariamente para gerir os "inserts" em sub-entidades, garantindo a coerência do sistema

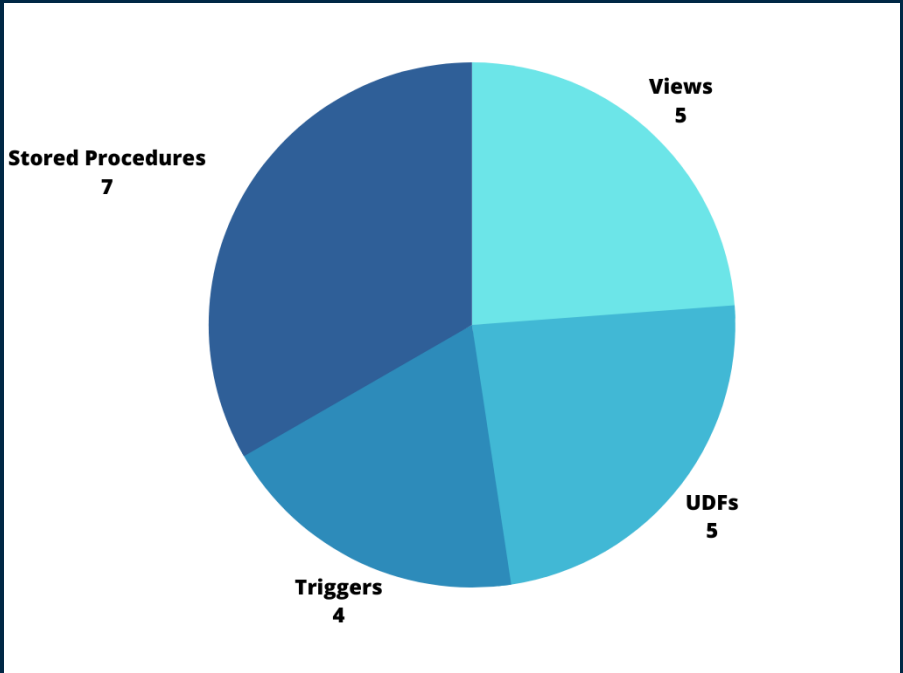
- **Triggers**: Intuito de gerar obrigatoriedades para que o sistema siga as regras de negócio, realizar a declaração de atributos que seguem uma ordem de incremento e, além disso, prevenir delete em pontos e momentos específicos.

- **Functions**: Objetivo principal de buscar informações complexas na base de dados

- **Views**: Utilizadas para fornecer uma representação virtual de dados contidos em diferentes tabelas

- **Cursor**: Implementado para percorrer tabelas que podem ter um grande volume de dados

Sql Proggraming Utilizado



Exemplos: Stored Procedures

```
-- procedure to add new manager, also adding it to the user and person table

CREATE PROC Project.addManager
@nif CHAR(9), @addr VARCHAR(30), @name VARCHAR(50), @phone CHAR(9), @email VARCHAR(30), @bdate DATE,
@password VARCHAR(20), @username VARCHAR(30), @salary INT,
@url VARCHAR(50)
AS
    BEGIN TRAN
        BEGIN TRY
            IF NOT EXISTS(SELECT NIF FROM p5g4.Project.PERSON WHERE NIF=@nif)
            BEGIN
                EXEC p5g4.Project.addUser @nif, @addr, @name, @phone, @email, @bdate, @password, @username, @salary

                DECLARE @userID INT;
                SET @userID = (SELECT ID FROM p5g4.Project.[USER] WHERE UserNIF=@nif);

                INSERT INTO p5g4.Project.Manager VALUES (@nif, @userID, @url)
            END
            COMMIT TRAN
        END TRY
        BEGIN CATCH
            IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
            RAISERROR('Not able to insert Manager (NIF already in the DB)', 16, 1)
            -- SELECT ERROR_MESSAGE() AS ErrorMessage;
        END CATCH
```

Ao inserir um novo manager é inserido também em “user” e “person”

Exemplos: Trigger

```
CREATE TRIGGER [Project].[CheckStaffSalary]
ON [Project].[STAFF]
AFTER INSERT, UPDATE
AS
BEGIN

    IF (EXISTS(SELECT 1 FROM inserted))
    BEGIN

        IF EXISTS(
            SELECT 1
            FROM inserted i
            INNER JOIN Project.MANAGER m ON i.StoreURL = m.StoreURL
            INNER JOIN Project.[USER] us ON i.StaffNIF = us.UserNIF AND i.ID = us.ID
            WHERE us.Salary > (SELECT Salary FROM Project.[USER] WHERE UserNIF = m.ManagerNIF AND ID = m.ID)
        )
        BEGIN

            RAISERROR ('O salário do usuário "Staff" não pode ser maior do que o salário do usuário "Manager" na mesma loja.', 16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END
    END
END;
```

Garante que um membro da staff não ganhe mais que o seu manager

Exemplos: Trigger (2)

```sql
-- this table has a attribute ContactID that is autoincremented
-- so we need to get the last inserted ContactID
-- and then insert it into the trigger table

CREATE TRIGGER addContactTransportTrigger
ON Project.CONTACT_TRANSPORT
INSTEAD OF INSERT
AS
    BEGIN
        DECLARE @ContactID INT;
        DECLARE @ManagerNIF CHAR(9);
        DECLARE @ManagerID INT;
        DECLARE @TransportNumber INT;
        DECLARE @ContactDate DATETIME;
        DECLARE @ContactDescription VARCHAR(200);

        SELECT @ContactID = MAX(ContactID) FROM Project.CONTACT_TRANSPORT;
        -- if the table is empty
        IF @ContactID IS NULL
            SET @ContactID = 1;
        ELSE
            SET @ContactID = @ContactID + 1;

        SELECT @ManagerID = ManagerID, @ManagerNIF = ManagerNIF, @TransportNumber = TransportNumber, @ContactDate = ContactDate, @ContactDescription
= ContactDescription
        FROM inserted;

        INSERT INTO Project.CONTACT_TRANSPORT (ContactID, ManagerNIF, ManagerID, TransportNumber, ContactDate, ContactDescription)
        VALUES (@ContactID, @ManagerNIF, @ManagerID, @TransportNumber, @ContactDate, @ContactDescription);
    END;
```

Gera o incremento do ID para os contatos Manager-Transport

Exemplos: Functions

Buscar informações de uma loja

```
CREATE FUNCTION Project.getStoreRevenue(@StoreURL VARCHAR(50)) RETURNS TABLE
AS
RETURN
(
    SELECT s.StoreName, SUM(o.Price) AS TotalSales
    FROM Project.STORE s
    LEFT JOIN Project.[ORDER] o ON s.StoreURL = o.StoreURL
    WHERE s.StoreURL=@StoreURL
    GROUP BY s.StoreName
);
```

```
CREATE FUNCTION Project.GetItemDescriptionBySupplierName(@SupName VARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT i.ItemDescription
    FROM Project.ITEM i
    JOIN Project.SUPPLIES s ON i.ID = s.ItemID
    JOIN Project.SUPPLIER sup ON s.SupplierID = sup.ID
    WHERE sup.SupName = @SupName
);
```

Items fornecidos por um supplier especifico

Exemplos: Views

```
CREATE VIEW Project.ItemsInOrderView
AS
    SELECT OrderNumber, ItemID, ITEM_ORDER.Quantity, ItemDescription, Price
    FROM Project.ITEM_ORDER JOIN Project.ITEM ON
Project.ITEM_ORDER.ItemID=Project.ITEM.ID
```

Itens presentes, atualmente, em pedidos

```
CREATE VIEW Project.StaffTotalStatsView
AS
    SELECT StaffNIF, StaffID, COUNT(StaffNIF) AS TotalOrders, SUM(Price) AS TotalMoney
FROM (Project.MANAGES JOIN Project.[ORDER] ON Project.Manages.OrderNumber=Project.
[ORDER].OrderNumber)
    GROUP BY StaffNIF, StaffID
```

Informação relacionada a uma staff

Exemplos: Cursor

```sql
CREATE PROCEDURE Project.deleteCanceledOrders @NIF INT
AS
BEGIN
    DECLARE @StoreURL VARCHAR(50);
    DECLARE @OrderNumber INT;
    DECLARE @CostumerNIF CHAR(9);

    SET @StoreURL = (SELECT StoreURL FROM Project.MANAGER WHERE ManagerNIF = @NIF);

    IF @StoreURL IS NULL
    BEGIN
        RAISERROR('Manager with NIF %d does not exist', 16, 1, @NIF);
        RETURN;
    END

    DECLARE CanceledOrders CURSOR FAST_FORWARD
        FOR SELECT OrderNumber, CostumerNIF
        FROM Project.[ORDER]
        WHERE StoreURL = @StoreURL AND OrderState = 'Canceled';

    OPEN CanceledOrders;

    FETCH CanceledOrders INTO @OrderNumber, @CostumerNIF;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        DELETE FROM Project.ITEM_ORDER WHERE OrderNumber = @OrderNumber AND CostumerNIF = @CostumerNIF;
        DELETE FROM Project.MANAGES WHERE OrderNumber = @OrderNumber AND CostumerNIF = @CostumerNIF;
        DELETE FROM Project.[ORDER] WHERE OrderNumber = @OrderNumber AND CostumerNIF = @CostumerNIF;

        FETCH CanceledOrders INTO @OrderNumber, @CostumerNIF;
    END

    CLOSE CanceledOrders;

    DEALLOCATE CanceledOrders;
END
```

Excluir pedidos cancelados

DEMO