

# Apresentação do projeto final de Base de Dados

P5G5

**D8 Imobiliárias**



Diogo Cruz  
Artur Romão

nMec: 98595  
nMec: 98470

# Introdução

Este projeto baseou-se na criação de um sistema de dados de uma imobiliária.

Algumas entidades existentes:

- **Pessoa**
- **Marcação**
- **Departamento**
- **Proposta**
- **Imóvel**
- **Vendido**
- **Negócio**
- **Comercial**
- **Habitacional**

# Funcionalidades



**Listar Imóveis**



**Acesso a informação  
por parte dos agentes  
e dos proprietários**



**Comprar/Vender  
Imóveis**

# Interface Venda

Vender



Imobiliárias

Nome - Atalanta Binning

Email - abinning3@cnn.com

NIF - 876915089

Tipo de Negócio

Tipo de Imóvel

Localização

Área total

Área útil

Preço

Ano de construção

Quartos

WCs

Churrasqueira ☐ Sim ☐ Não

Piscina ☐ Sim ☐ Não

Painéis Solares ☐ Sim ☐ Não

Jardim ☐ Sim ☐ Não

Jacuzzi ☐ Sim ☐ Não

Garagem ☐ Sim ☐ Não

Quantidade

Quantidade

Quantidade

Quantidade

Quantidade


Quantidade

Anunciar

Voltar

# Interface Compra

Comprar



Imobiliárias

Nome - Atalanta Binning

Email - abinning3@cnn.com

NIF - 876915089

Tipo de negócio

Tipo de imóvel

Localização

Preço

Área

Quartos

WCs

Construída a partir de

Min

Max

Quinta - 420 Steensland Place, Lisboa | 39710€  
Apartamento - 02 Village Hill, Ovar | 89411€  
Loja - 35993 Hanson Hill, Lisboa | 24938€  
Quarto - 48 Del Sol Circle, Santarém | 122622€  
Quinta - 7 Amoth Crossing, Lagoa | 67280€  
Armazém - 91 Amoth Avenue, Faro | 140323€  
Quinta - 72 Ludington Lane, Setúbal | 7470€  
Quarto - 77324 Montana Circle, Lisboa | 20845€  
Armazém - 40 Northview Court, Setúbal | 122474€  
Terreno - 17 Maryland Junction, Braga | 65929€  
Apartamento - 8 Utah Park, Porto | 141186€  
Escritório - 7 Bayside Street, Espinho | 12648€  
Escritório - 48 Golden Leaf Plaza, Aveiro | 119848€  
Apartamento - 925 Glacier Hill Trail, Leiria | 28939€  
Loja - 857 Scofield Center, Canedo | 142447€  
Moradia - 5470 Helena Lane, Aveiro | 14345€

Aplicar

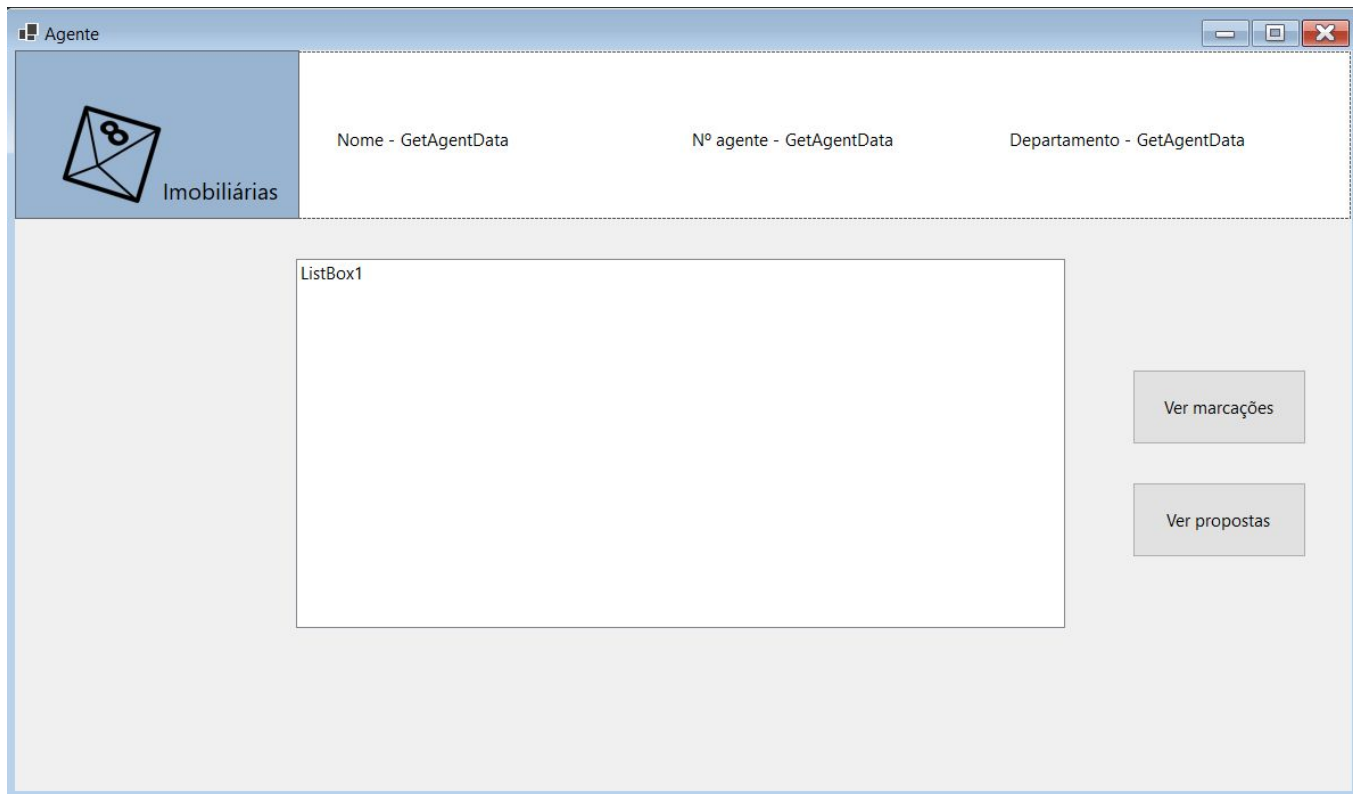
Fazer marcação

Fazer proposta

Mais informação

Voltar

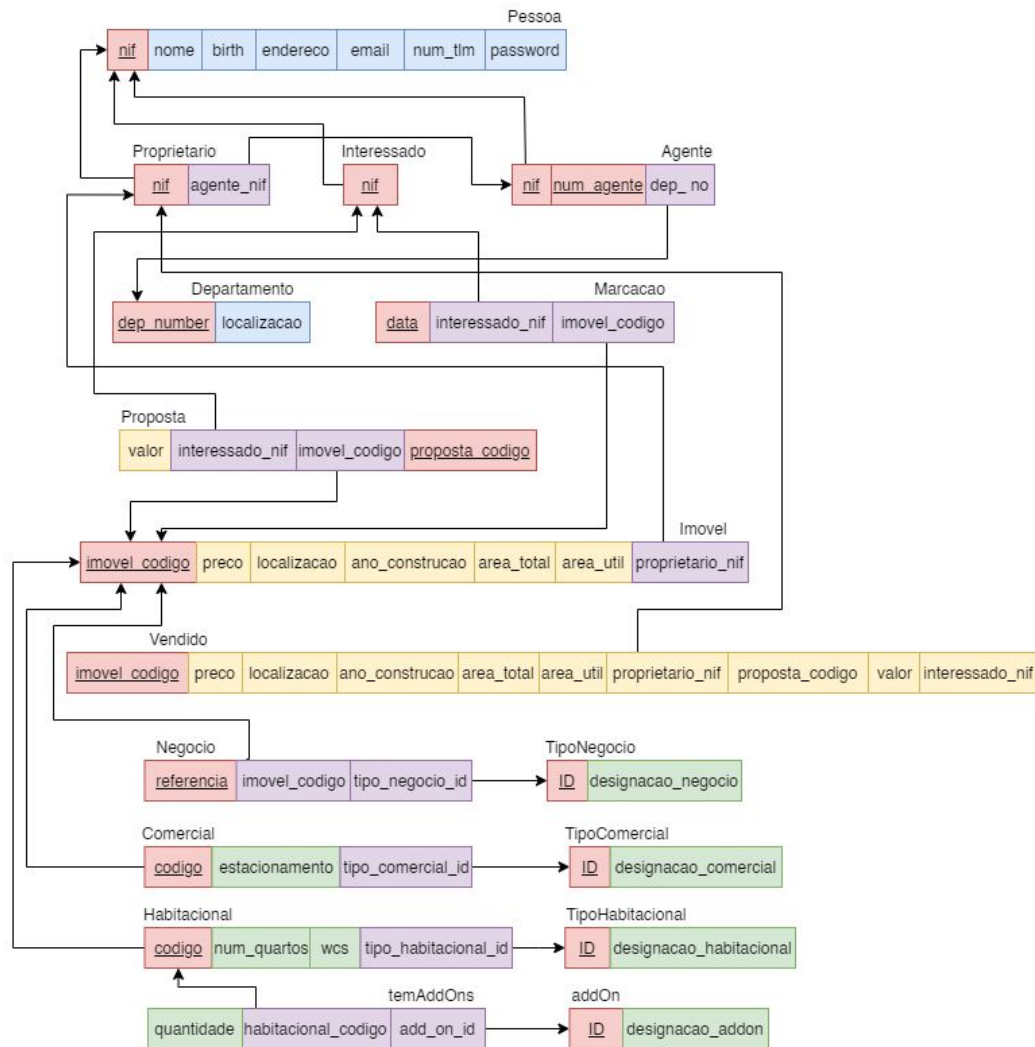
# Interface Agente



# Implementação das funcionalidades

- **Triggers:** Fazer verificações aos dados do utilizador (ex: verificar se a data inserida é maior que hoje)
- **Views:** Em vez de usar vários *join* são usadas Views (ex: ver todos os imóveis no mercado)
- **Stored Procedures:** Maioritariamente para inserir dados novos (ex: adicionar um imóvel)
- **Functions:** Usados maioritariamente para verificações ou buscar informações quando existem parâmetros inseridos (ex: ver imóveis associados a um agente)
- **Transactions:** Usado para eliminar uma linha da tabela e todas as suas dependências.

# ER





# Points scored

Views

18,5%

Triggers

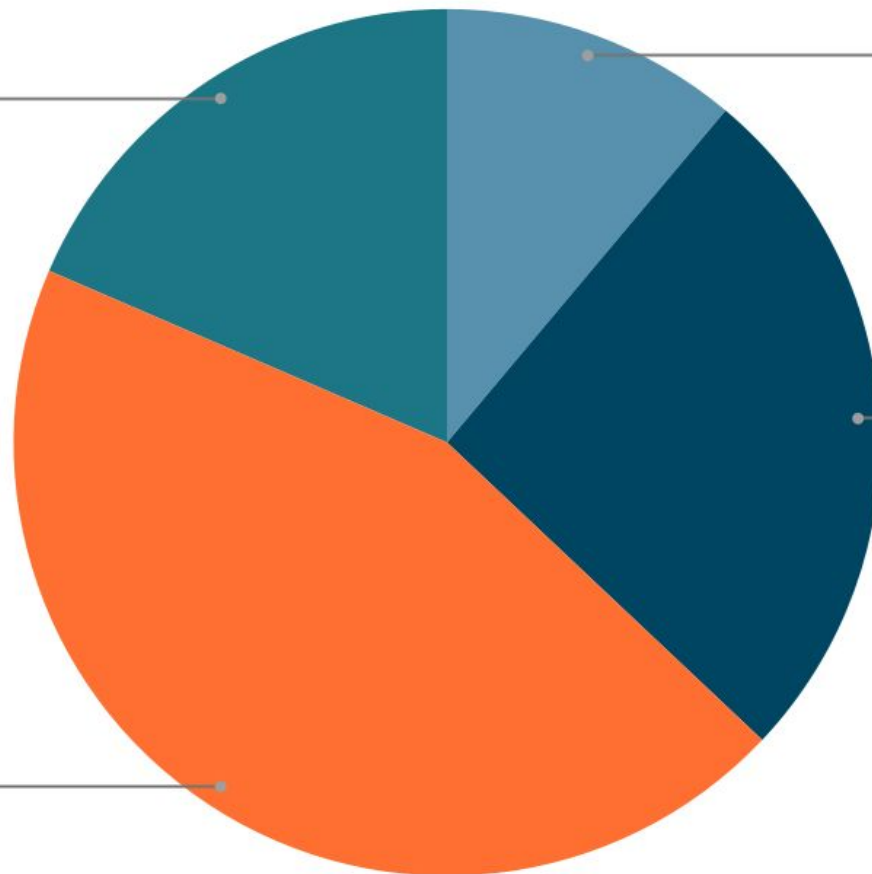
11,1%

SPs

25,9%

UDFs

44,4%



A solid orange vertical bar is positioned on the left side of the slide.

# Exemplos

# Stored Procedures

```
CREATE PROCEDURE Proj.[cp_add_imovel_habitacional] -- criar e adicionar a imovel, habitacional, proprietario e negocio
    @preco INT, @localizacao VARCHAR(50), @ano_construcao INT, @area_total INT, @area_util INT, @proprietario_nif INT,
    @num_quartos INT, @wcs INT, @habitacional_id INT, @negocio_id INT
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        SET NOCOUNT ON
        IF EXISTS(SELECT nif FROM p5g5.Proj.[pessoa] WHERE nif=@proprietario_nif) -- se o nif existe
        BEGIN
            -- se o nif_prop n existe
            IF NOT EXISTS(SELECT proprietario_nif FROM p5g5.Proj.[proprietario] WHERE proprietario_nif=@proprietario_nif)
            BEGIN
                DECLARE @agente_nif INT -- gerar agente aleatorio entre os que existem
                SET @agente_nif = (SELECT TOP 1 agente_nif FROM p5g5.Proj.[agente] ORDER BY NEWID())

                -- se ainda nao está na tabela de proprietarios, adiciona
                INSERT INTO p5g5.Proj.[proprietario] VALUES (@proprietario_nif, @agente_nif)
            END

            DECLARE @imovel_codigo VARCHAR(5) -- geração de um codigo random
            SET @imovel_codigo = (SELECT p5g5.Proj.[udf_createImovelCode]())

            DECLARE @referencia VARCHAR(9) -- gerar referencia para negocio
            SET @referencia = (SELECT p5g5.Proj.[udf_createRefCode]())

            -- se nao for um imovel repetido, adiciona a tabela imovel
            IF NOT EXISTS(SELECT localizacao FROM p5g5.Proj.[imovel] WHERE localizacao=@localizacao)
            INSERT INTO p5g5.Proj.[imovel] (imovel_codigo, preco, localizacao, ano_construcao, area_total, area_util, proprietario_nif)
            VALUES(@imovel_codigo, @preco, @localizacao, @ano_construcao, @area_total, @area_util, @proprietario_nif)

            -- add tabela habitacional
            INSERT INTO p5g5.Proj.[habitacional] (imovel_codigo, num_quartos, wcs, tipo_habitacional_id)
            VALUES(@imovel_codigo, @num_quartos, @wcs, @habitacional_id)

            -- add tabela negocio
            INSERT INTO p5g5.Proj.[negocio] (referencia, imovel_codigo, tipo_negocio_id)
            VALUES(@referencia, @imovel_codigo, @negocio_id)
        END
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        RAISERROR('couldnt add imovel habitacional', 16, 1)
    END CATCH
END
GO
```

# Triggers

```
CREATE TRIGGER Proj.[trigger_register_pessoa] ON Proj.[pessoa] --registar pessoa nova
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nif INT, @nome VARCHAR(50), @birth DATE, @endereco VARCHAR(50), @email VARCHAR(50), @num_tlm INT, @password VARCHAR(50);

    SELECT @nif=nif, @nome=nome, @birth=birth, @endereco=endereco, @email=email,
           @num_tlm=num_tlm, @password=CONVERT(VARCHAR(20), DECRYPTBYPASSPHRASE('*****',[password])) FROM inserted;
    IF ((SELECT p5g5.Proj.[udf_validateEmail](@email)) > 0) -- existe na base de dados pessoa c esse email
        RAISERROR('Email is already registered!', 16, 1)
    ELSE
        INSERT INTO p5g5.Proj.[pessoa](nif, nome, birth, endereco, email, num_tlm, [password])
        VALUES (@nif, @nome, @birth, @endereco, @email, @num_tlm, ENCRYPTBYPASSPHRASE('*****', @password))
END
GO
```

# Functions

```
CREATE FUNCTION Proj.[udf_validateEmail](@email VARCHAR(50)) RETURNS INT
AS
BEGIN
    IF EXISTS(SELECT * FROM Proj.[Pessoa] AS P WHERE P.email = @email)
        RETURN 1;
    RETURN 0;
END
GO
```

```
CREATE FUNCTION Proj.[udf_createImovelCode] () RETURNS VARCHAR(5)
AS
BEGIN
    DECLARE @temp VARCHAR(5)
    SET @temp = (SELECT Value FROM p5g5.Proj.[view_getImobRand])

    WHILE EXISTS (SELECT imovel_codigo FROM p5g5.Proj.[imovel] WHERE imovel_codigo = @temp)
    BEGIN
        SET @temp = (SELECT Value FROM p5g5.Proj.[view_getImobRand])
    END
    RETURN @temp
END
GO
```

# Views

```
CREATE VIEW Proj.[view_getAllImob] -- informacao de todos os imoveis
AS
|   SELECT preco, localizacao, ano_construcao, area_total, area_util
|   FROM Proj.[imovel];
GO
```

```
CREATE VIEW Proj.[view_getDepNumAgent]
AS
|   SELECT D.dep_number, D.localizacao, COUNT(*)
|   FROM Proj.[agente] AS A JOIN Proj.[dept] AS D ON A.dep_no = D.dep_number
GO
```

# Transactions

```
CREATE PROCEDURE Proj.[delete_imovel]
    @imovel_codigo VARCHAR(5)
AS
BEGIN TRANSACTION
    DELETE FROM Proj.temAddOn WHERE habitacional_codigo=@imovel_codigo
    DELETE FROM Proj.habitacional WHERE imovel_codigo=@imovel_codigo
    DELETE FROM Proj.comercial WHERE imovel_codigo=@imovel_codigo
    DELETE FROM Proj.proposta WHERE imovel_codigo=@imovel_codigo
    DELETE FROM Proj.marcacao WHERE imovel_codigo=@imovel_codigo
    DELETE FROM Proj.negocio WHERE imovel_codigo=@imovel_codigo
    DELETE FROM Proj.imovel WHERE imovel_codigo=@imovel_codigo
COMMIT
GO
```

# Demo