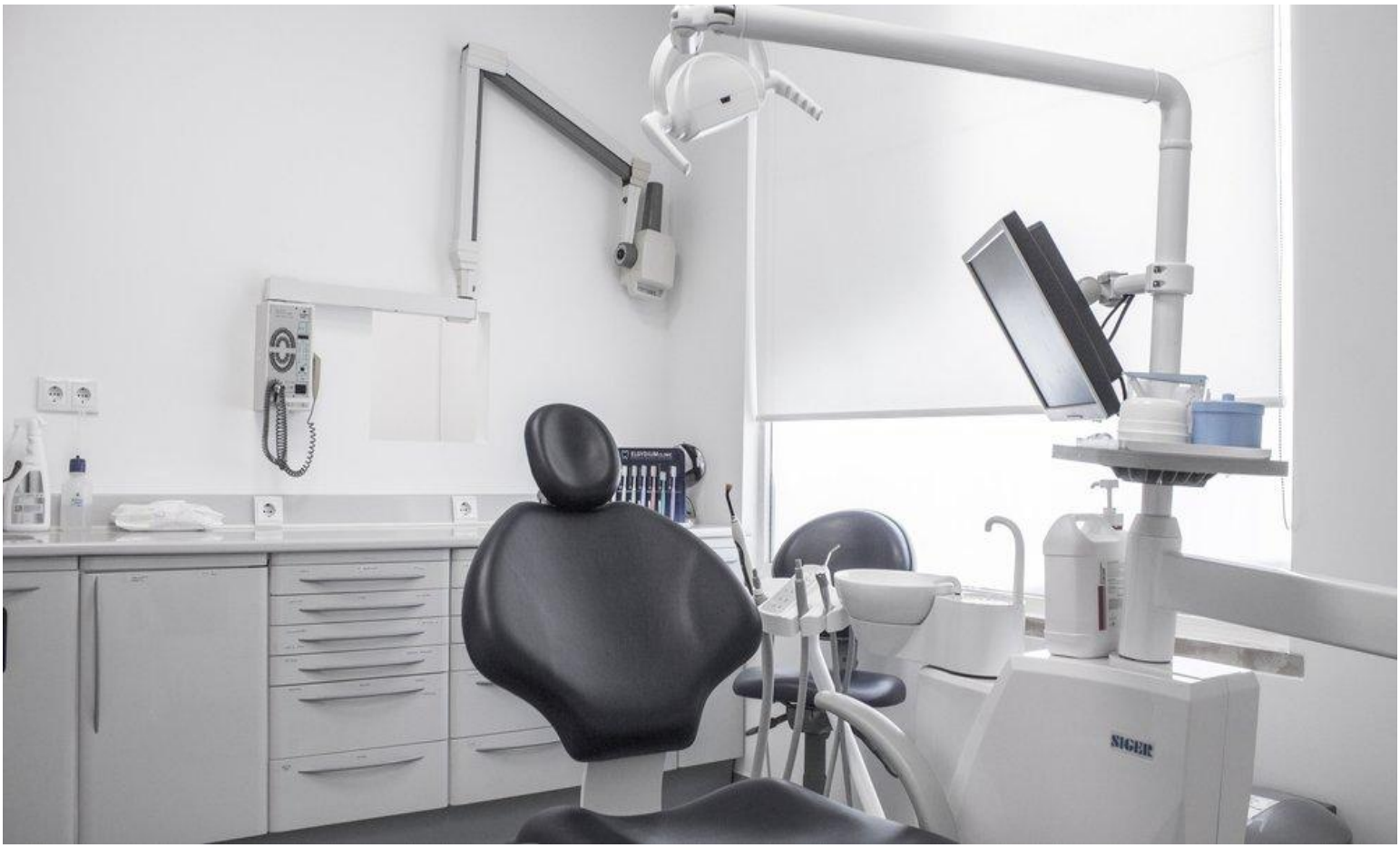
Clinica Dentaria

Base de Dados 24/25



Guilherme Carreira 120159
Rafael Silva 120020
P3-G6

universidade de aveiro

Índice

Introdução

O presente projeto, desenvolvido no âmbito da unidade curricular de Bases de Dados, tem como objetivo a conceção e implementação de uma base de dados para apoiar a gestão de uma clínica dentária. A base de dados foi projetada para organizar e centralizar a informação relevante de forma eficiente, permitindo à clínica melhorar os seus processos administrativos e operacionais.

A aplicação prática desta base de dados visa dar suporte a diversas atividades essenciais da clínica, como o registo de pacientes, marcação de consultas, registo de tratamentos realizados, gestão de faturas e recibos, bem como o armazenamento de informação médica e dentária de cada paciente. Além disso, o sistema contempla ainda o registo dos profissionais envolvidos no funcionamento da clínica, incluindo médicos dentistas, enfermeiros e rececionistas.

Requisitos Fundamentais

Com base em pesquisa documental e entrevistas realizadas a profissionais da área da saúde dentária, foi possível identificar os seguintes requisitos funcionais para o sistema de base de dados:

- Gestão de pacientes: Registo e atualização de dados pessoais e clínicos dos pacientes;
- Marcação de consultas: Planeamento e calendarização de consultas com os respetivos profissionais;
- Registo de tratamentos: Associação de tratamentos realizados às consultas de cada paciente;
- Informação dentária: Armazenamento de dados clínicos relevantes, como histórico de tratamentos e observações médicas;
- Faturação e recibos: Emissão e gestão de faturas relativas aos serviços prestados;
- Gestão de funcionários: Registo dos dados dos profissionais da clínica, incluindo médicos dentistas, enfermeiros e rececionistas.

Entidades e Relações

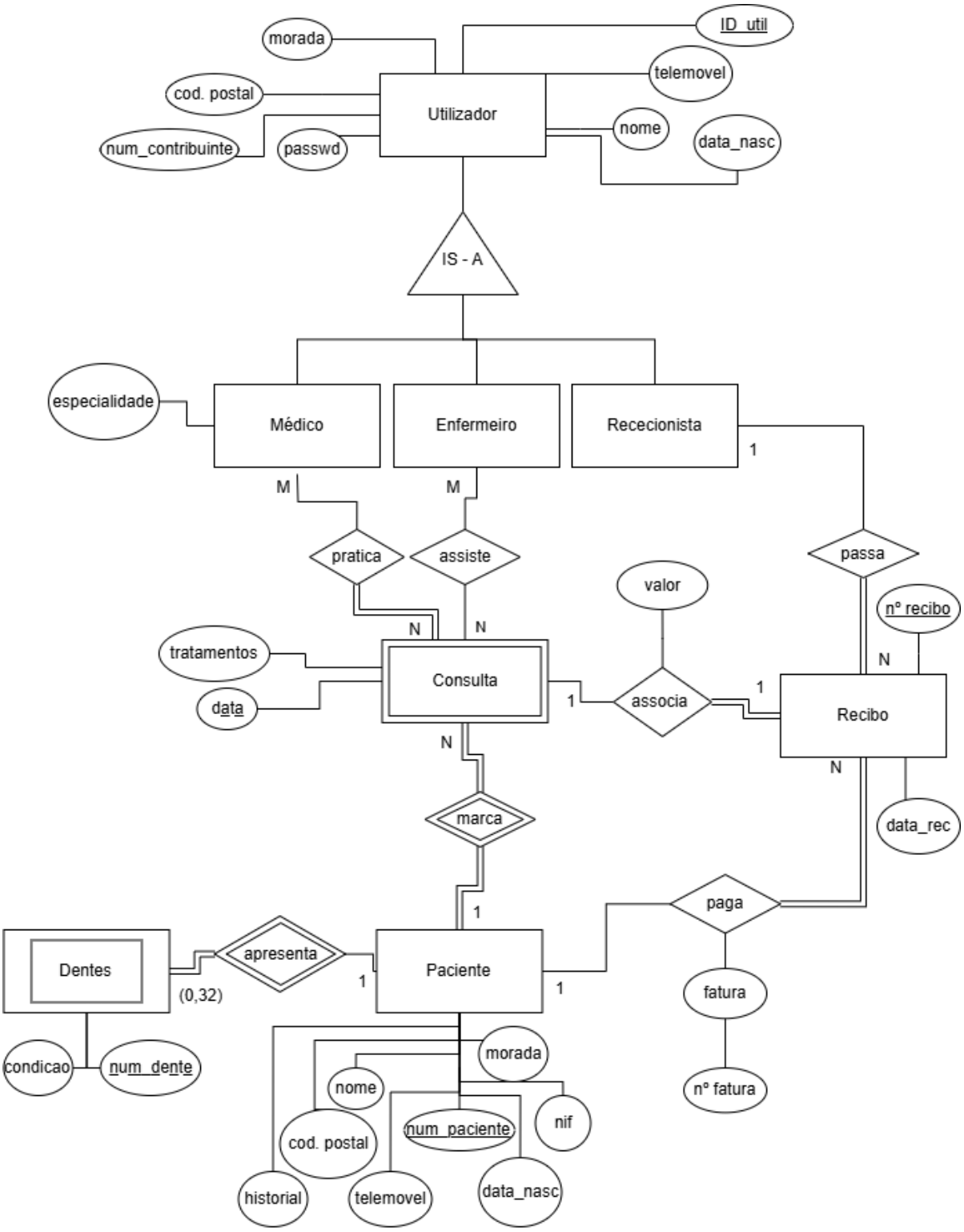
Com base nos requisitos previamente definidos e na análise funcional do sistema, foram identificadas as seguintes entidades principais:

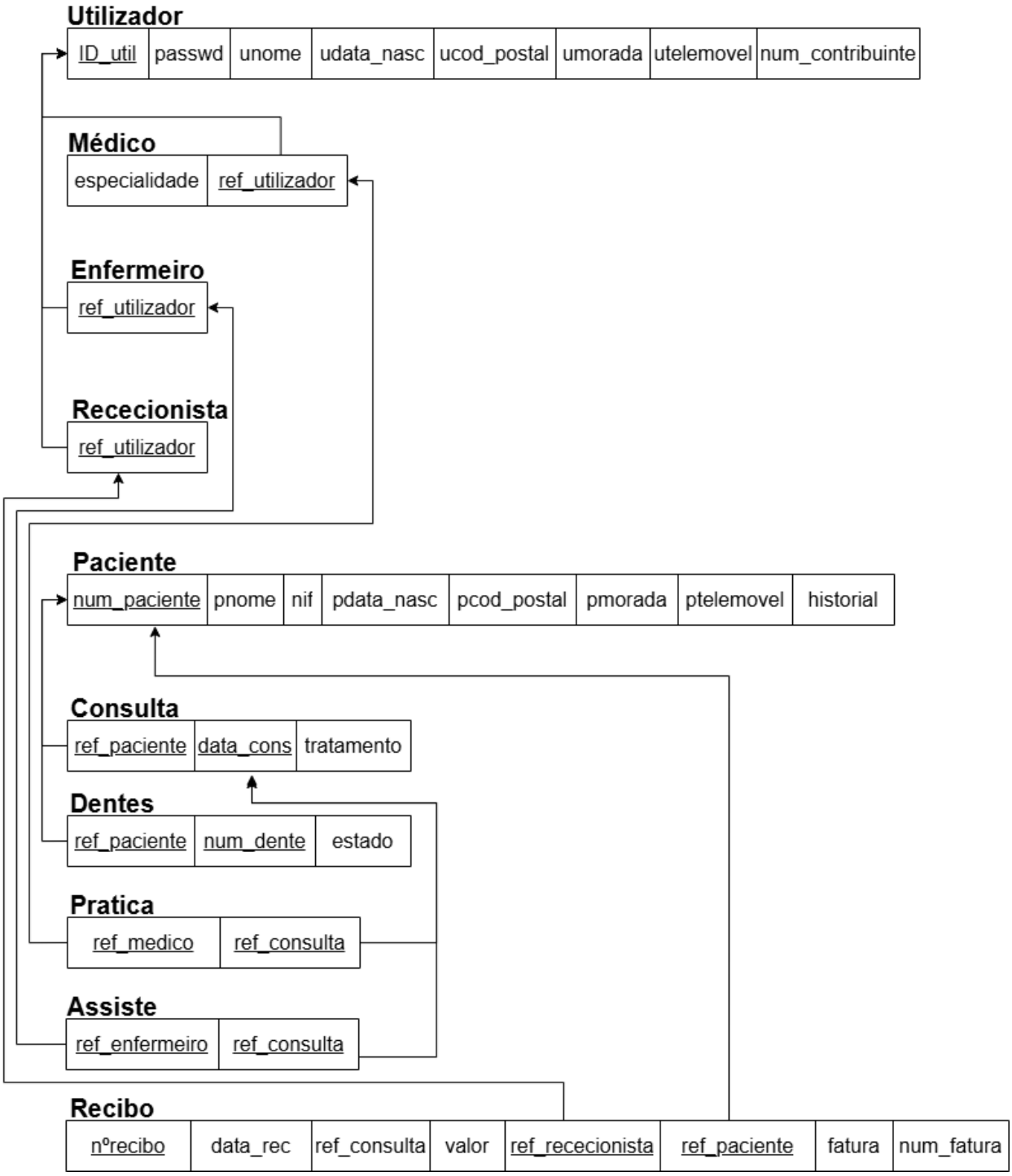
- Utilizador: representa os trabalhadores da clínica, nomeadamente médicos, enfermeiros e rececionistas. Contém dados pessoais como nome, data de nascimento, morada, contacto telefónico e número de contribuinte. Cada tipo de utilizador especializa a entidade principal através da relação "IS-A", sendo os médicos ainda associados a uma especialidade.
- Paciente: armazena a informação pessoal dos pacientes, como nome, data de nascimento, NIF, morada, contacto telefónico e histórico clínico. Esta entidade está relacionada com outras que representam os tratamentos e consultas realizadas.
- Dentes: representa os dentes de cada paciente, sendo possível armazenar o estado de cada um. Cada paciente pode apresentar até 32 dentes, com diferentes condições clínicas.
- Consulta: regista as consultas realizadas na clínica, associando a data, o paciente, os tratamentos efetuados e os profissionais envolvidos (médico e/ou enfermeiro). É também associada à faturação.
- Recibo: está relacionado com as consultas e representa os documentos de pagamento. Armazena o número do recibo, a data de emissão, o valor cobrado, o número da fatura e os dados da rececionista responsável.
- Relacionamentos:
    - Um paciente pode marcar várias consultas.

- Cada consulta pode ser realizada por um ou mais profissionais (médico e/ou enfermeiro).
- As consultas podem gerar recibos, os quais são emitidos por rececionistas.
- A entidade Dentes está associada aos pacientes, permitindo armazenar o estado de cada dente.
- A relação entre Utilizador e os seus papéis (médico, enfermeiro, recepcionista) é feita através de especializações.

Diagramas

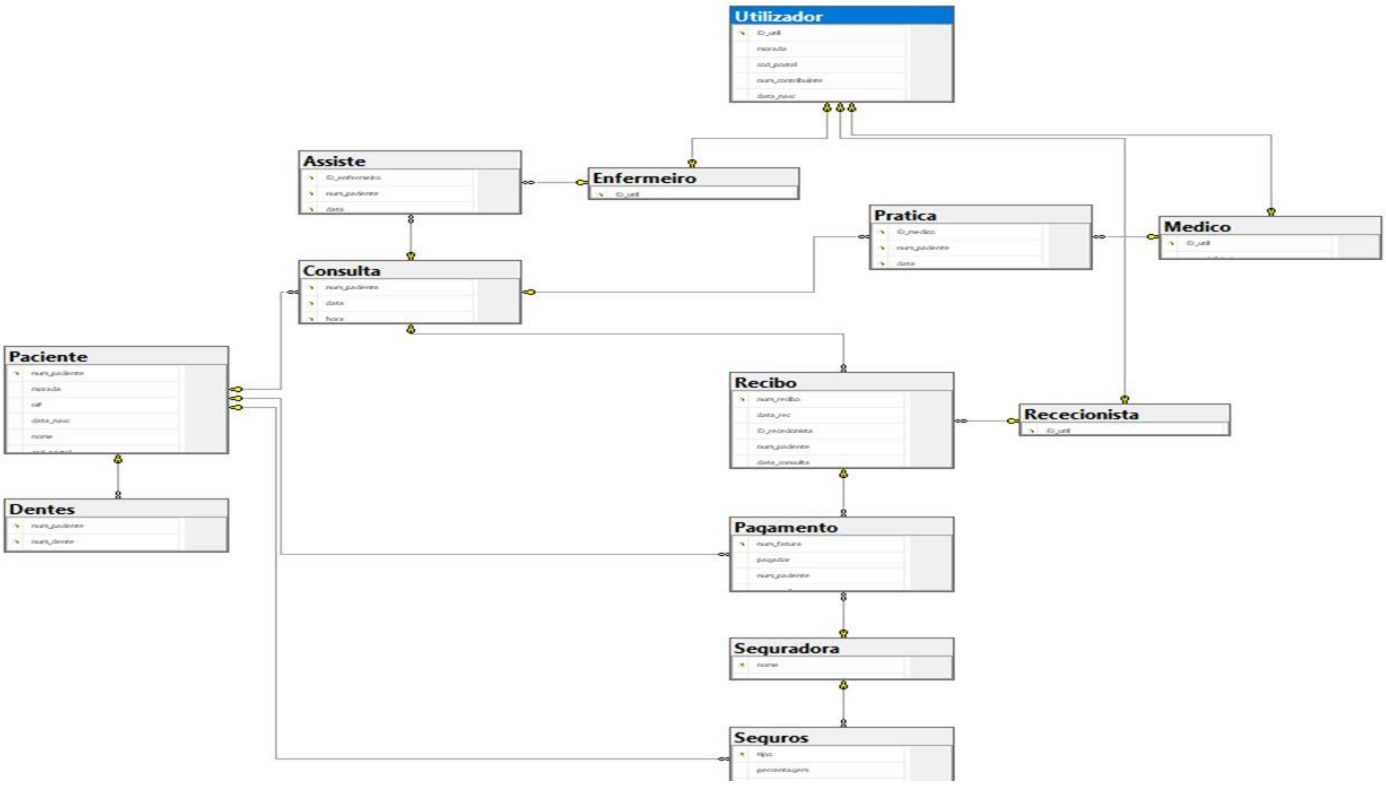
DER-Diagrama Entidade-Relacionamento





ER-Esquema Relacional

ER-Esquema Relacional(SGDB)



DDL-Data Definition Language

A DDL (Data Definition Language) é uma componente essencial da linguagem SQL (Structured Query Language) e tem como principal objetivo a definição da estrutura dos dados, através da criação, modificação e eliminação de objetos de base de dados, como tabelas, chaves primárias, estrangeiras, e restrições de integridade.

No âmbito deste projeto, foi elaborada uma sequência de instruções DDL que permite a criação das tabelas necessárias à gestão da clínica dentária. Estas instruções foram implementadas num sistema de gestão de base de dados (SQL Server), tendo em conta os requisitos funcionais e o modelo lógico previamente definidos.

Destaques da Estrutura:

- A tabela Utilizador serve como base para as entidades especializadas Médico, Enfermeiro e Rececionista, usando uma abordagem de herança com chaves primárias partilhadas.
- A entidade Paciente regista todos os dados pessoais e clínicos dos pacientes, incluindo um campo de historial e dados de contacto.
- A tabela Consulta liga-se ao paciente e permite o registo de tratamentos realizados, sendo identificada por uma chave composta (paciente, data, hora).
- A entidade Dentes regista o estado de cada dente de um paciente, permitindo um acompanhamento detalhado da saúde dentária.
- A entidade Recibo armazena dados de faturação associados a consultas e inclui relações com pacientes e rececionistas.
- As entidades Seguradora, Seguros e Pagamento introduzem um mecanismo para gerir faturas e seguros de saúde associados às consultas.

- As tabelas Pratica e Assiste representam relações muitos-para-muitos entre consultas e os profissionais que nelas participam (médicos e enfermeiros, respetivamente).

Segue em anexo (dentistaDDL.sql) o Código SQL para a criação dos DDL.

DML-Data Manipulation Language

A Data Manipulation Language (DML) é uma parte essencial da linguagem SQL, responsável pelas operações que manipulam os dados armazenados na base de dados. Estas operações incluem inserção (`INSERT`), atualização (`UPDATE`), remoção (`DELETE`) e consulta (`SELECT`) de registos.

No contexto deste projeto, utilizámos a DML para inserir dados realistas nas tabelas criadas previamente com a DDL. Estes dados foram estruturados de modo a simular um cenário funcional completo de um consultório dentário, com utilizadores (médicos, enfermeiros e rececionistas), pacientes, consultas, tratamentos, recibos, pagamentos e seguros.

Foram realizados os seguintes passos:

- Inserção de dados nas tabelas de utilizadores e respetivas especializações (médico, enfermeiro, rececionista);
- Inserção de pacientes com históricos clínicos;
- Inserção de seguradoras e seguros associados aos pacientes;
- Registo do estado dentário dos pacientes;
- Inserção de consultas, com a respetiva associação a médicos (Pratica) e enfermeiros (Assiste);
- Emissão de recibos para cada consulta;

- Registo de pagamentos, com ou sem apoio de seguradora.

Aqui esta alguns dos dados que inserimos segue a totalidade em anexo (dentistaDATA.sql).

```
-- Utilizadores
INSERT INTO Utilizador VALUES
(1, 'Rua das Flores 12', '1200-100', 123456789, '1975-04-22', '912345678', 'Dr. António Lopes'),
(2, 'Av. República 45', '1050-197', 234567891, '1980-02-10', '913456789', 'Dr. Beatriz Silva'),
...

-- Médicos
INSERT INTO Medico VALUES
(1, 'Ortodontia'),
(2, 'Odontopediatria'),
...

-- Pacientes
INSERT INTO Paciente VALUES
(101, 'Rua Nova 1', 111111111, '2010-06-12', 'Joana Marques', '1500-001', '921111111', 'Usa aparelho desde 2023'),
...

-- Consultas
INSERT INTO Consulta VALUES
(101, '2025-06-01', '09:00', 'Ajuste de aparelho'),
...

-- Recibos
INSERT INTO Recibo VALUES
(1, '2025-06-01', 6, 101, '2025-06-01', '09:00', 60.00),
...

-- Pagamentos
INSERT INTO Pagamento VALUES
(1, 'Medis', 101, 1, 'Medis'),
...
```

Esta fase é fundamental para testar o modelo de dados implementado e garantir que todas as relações entre tabelas funcionam corretamente com dados reais. Além disso, permite validar as restrições de integridade e simular cenários típicos de utilização no contexto clínico.

VIEWS

As Views (ou vistas) são uma poderosa funcionalidade do SQL que permite encapsular consultas personalizadas como objetos virtuais na base de dados. Estas vistas não armazenam dados fisicamente, mas sim a lógica de consulta, permitindo:

- Maior facilidade no acesso aos dados;
- Melhor organização da informação;
- Reutilização de lógica em Stored Procedures, UDFs (User Defined Functions) e interfaces de aplicações.

No âmbito deste projeto, as views foram amplamente utilizadas para facilitar a apresentação de dados complexos, reduzindo a complexidade das consultas repetitivas e promovendo a separação entre lógica de aplicação e estrutura de dados.

```
USE Dentista;
GO

CREATE VIEW vw_Consulta AS
SELECT
    c.data AS DataConsulta,
    p.num_paciente AS ID_Paciente,
    p.nome AS Nome_Paciente,

    m.ID_util AS ID_Medico,
    u_med.nome AS Nome_Medico,
    m.especialidade AS Especialidade_Medico,

    e.ID_util AS ID_Enfermeiro,
    u_enf.nome AS Nome_Enfermeiro

FROM Consulta c
JOIN Paciente p
    ON c.num_paciente = p.num_paciente

LEFT JOIN Pratica pr
    ON c.num_paciente = pr.num_paciente
    AND c.data = pr.data
    AND c.hora = pr.hora

LEFT JOIN Medico m
```

```
    ON pr.ID_medico = m.ID_util

LEFT JOIN Utilizador u_med
    ON m.ID_util = u_med.ID_util

LEFT JOIN Assiste a
    ON c.num_paciente = a.num_paciente
    AND c.data = a.data
    AND c.hora = a.hora

LEFT JOIN Enfermeiro e
    ON a.ID_enfermeiro = e.ID_util

LEFT JOIN Utilizador u_enf
    ON e.ID_util = u_enf.ID_util;
```

Esta view é extremamente útil para relatórios ou listagens na aplicação, permitindo mostrar, por exemplo:

- Quem foi o médico e o enfermeiro de determinada consulta;
- Quais os pacientes atendidos num dado dia;
- Consultas realizadas por especialidade médica.

SP-Stored Procedures

Os *Stored Procedures* constituem um recurso fundamental na gestão de bases de dados relacionais, permitindo a encapsulação de blocos de código SQL que podem ser armazenados, reutilizados e executados no servidor de forma eficiente e segura.

Entre as principais vantagens da sua utilização destacam-se:

- Melhoria do desempenho, uma vez que os procedimentos são pré-compilados e armazenados em cache;
- Organização e reutilização do código, promovendo uma estrutura modular e de fácil manutenção;

- Garantia de integridade e atomicidade, especialmente relevante em operações que envolvem múltiplas tabelas e transações;
- Redução da duplicação de lógica, ao centralizar a execução de operações críticas.

Neste projeto, os *Stored Procedures* foram utilizados para automatizar operações de inserção, atualização e remoção de dados, bem como para realizar pesquisas por atributos específicos (como o nome). O objetivo principal foi garantir consistência e integridade nas interações com os dados clínicos e administrativos.

A seguir, apresenta-se o conjunto de *Stored Procedures* implementados, cada um acompanhado de um comentário que descreve a sua finalidade no sistema.

```
USE Dentista;
GO

-- Criação de um novo paciente
CREATE OR ALTER PROCEDURE CriarPaciente
   @morada VARCHAR(100),
   @nif INT,
   @data_nasc DATE,
   @nome VARCHAR(100),
   @cod_postal VARCHAR(10),
   @telemovel VARCHAR(9),
   @historial VARCHAR(500)
AS
BEGIN
   DECLARE @num_paciente INT;

   SELECT @num_paciente = ISNULL(MAX(num_paciente), 0) + 1 FROM Paciente;

   INSERT INTO Paciente (num_paciente, morada, nif, data_nasc, nome, cod_postal,
telemovel, historial)
   VALUES (@num_paciente, @morada, @nif, @data_nasc, @nome, @cod_postal,
@telemovel, @historial);
END;
GO

-- Atualização dos dados de um paciente
CREATE OR ALTER PROCEDURE AtualizarPaciente
   @num_paciente INT,
   @morada VARCHAR(100),
   @cod_postal VARCHAR(10),
```

```
    @telemovel VARCHAR(9),
    @historial VARCHAR(500)
AS
BEGIN
    UPDATE Paciente
    SET morada = @morada,
        cod_postal = @cod_postal,
        telemovel = @telemovel,
        historial = @historial
    WHERE num_paciente = @num_paciente;
END;
GO

-- Remoção de um paciente (apenas se não existirem dados associados)
CREATE OR ALTER PROCEDURE RemoverPaciente
    @num_paciente INT
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM Consulta WHERE num_paciente = @num_paciente
        UNION
        SELECT 1 FROM Dentes WHERE num_paciente = @num_paciente
        UNION
        SELECT 1 FROM Recibo WHERE num_paciente = @num_paciente
        UNION
        SELECT 1 FROM Pagamento WHERE num_paciente = @num_paciente
        UNION
        SELECT 1 FROM Seguros WHERE num_paciente = @num_paciente
    )
    BEGIN
        RAISERROR('Paciente já associado a consulta/recibo/pagamento.', 16, 1);
        RETURN;
    END

    DELETE FROM Paciente WHERE num_paciente = @num_paciente;
END;
GO
```

O restante código dos SP seguem em anexo (dentistaSP.sql).

Triggers

Os Triggers (ou gatilhos) são objetos da base de dados que são executados automaticamente em resposta a eventos específicos que ocorrem em determinadas tabelas, como operações de inserção (INSERT), atualização (UPDATE) ou eliminação (DELETE). A sua

principal função é garantir a automatização de processos e a integridade dos dados sem necessidade de intervenção manual adicional.

Entre as vantagens mais relevantes destacam-se:

- Automatização de ações reativas, como o registo automático de eventos ou a propagação de alterações entre tabelas;
- Reforço da integridade lógica, especialmente em validações mais complexas que não podem ser facilmente asseguradas por constraints;
- Redução de erro humano, ao evitar que determinadas operações fiquem dependentes da aplicação cliente;
- Centralização da lógica de negócio, promovendo maior consistência entre diferentes interfaces de acesso à base de dados.

No contexto deste projeto, foram implementadas *triggers* com finalidades específicas: gerar automaticamente recibos e pagamentos após o registo de uma consulta, e impedir a atribuição de seguros infantis a pacientes com idade superior ou igual a 18 anos.

```
-- Trigger para gerar automaticamente um recibo após o registo de uma nova consulta
CREATE OR ALTER TRIGGER trg_gerar_recibo_apos_consulta
ON Consulta
AFTER INSERT
AS
BEGIN
    DECLARE @id_rececionista INT;
    SELECT TOP 1 @id_rececionista = ID_util FROM Rececionista ORDER BY NEWID();

    DECLARE @max_recibo INT;
    SELECT @max_recibo = ISNULL(MAX(num_recibo), 0) FROM Recibo;
```

```
    ;WITH NovasConsultas AS (
        SELECT
            ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum,
            num_paciente, data, hora
        FROM inserted
    )
    INSERT INTO Recibo (num_recibo, data_rec, ID_rececionista, num_paciente, data_consulta,
hora_consulta, valor)
    SELECT
        @max_recibo + RowNum,
        GETDATE(),
        @id_rececionista,
        num_paciente,
        data,
        hora,
        60.00
    FROM NovasConsultas;


END;
```

O restante código dos triggers segue em anexo (dentistaTriggers.sql).

UDF-User Defined Functions

As *User Defined Functions* (UDFs) são elementos fundamentais na extensão da funcionalidade do SQL, permitindo encapsular lógica reutilizável em funções definidas pelo utilizador. Estas funções podem ser invocadas em consultas, *views*, *stored procedures* ou *triggers*, tornando o código mais legível, modular e consistente.

As principais vantagens das UDFs incluem:

- Reutilização de lógica complexa em diferentes pontos da aplicação;
- Melhoria da legibilidade e manutenção do código SQL;
- Separação de responsabilidades, ao isolar cálculos ou transformações de dados.

Neste projeto, as UDFs foram utilizadas exclusivamente para o retorno de valores derivados dos dados existentes, como por exemplo, o cálculo da idade de um paciente com base na sua data de nascimento. Importa referir que, em conformidade com as boas práticas, as UDFs aqui aplicadas não realizam qualquer modificação direta nos dados da base de dados (inserção, atualização ou eliminação).

Abaixo é apresentado o código da função definida:

```
USE Dentista;
GO
-- Função que calcula a idade de um paciente a partir da data de nascimento
CREATE OR ALTER FUNCTION dbo.ObterIdade(@data_nasc DATE)
RETURNS INT
AS
BEGIN
   DECLARE @idade INT;
   SET @idade = DATEDIFF(YEAR, @data_nasc, GETDATE());

   IF DATEADD(YEAR, @idade, @data_nasc) > GETDATE()
      SET @idade = @idade - 1;

   RETURN @idade;
END;
GO
```

Esta função é particularmente útil para validações condicionais, como demonstrado na trigger que impede a atribuição de seguros infantis a pacientes com 18 anos ou mais, garantindo que a lógica de negócio é centralizada e mantida de forma uniforme.

Interface

Apesar da interface gráfica não constituir o foco principal deste projeto, foi desenvolvida uma aplicação simples em *Windows Forms* com o objetivo de demonstrar, de forma clara e intuitiva, a funcionalidade e

utilidade prática da base de dados implementada. A interface foi desenhada para permitir a gestão visual dos principais dados armazenados, promovendo a interação com o sistema de forma acessível, sobretudo no contexto da administração da clínica.

Estrutura da Interface

A aplicação foi desenvolvida com recurso à framework .NET (Windows Forms), sendo o ponto de entrada o formulário `Forms1.cs`, a partir do qual o utilizador pode aceder e interagir com a base de dados. Foram criados formulários específicos para as seguintes entidades:

- Pacientes
- Utilizadores (abrangendo médicos, enfermeiros e rececionistas)
- Consultas

Os três formulários apresentam uma estrutura semelhante, com listagens em tabelas (*DataGridViews*) e opções de adição, edição e remoção de registos. A comunicação entre a interface e a base de dados é feita através de repositórios próprios, garantindo a separação entre a lógica de apresentação e a lógica de acesso a dados.

Funcionalidades do Formulário de Pacientes

O formulário principal relacionado com os pacientes (`Pacientes.cs`) permite:

- Visualizar uma lista com todos os pacientes existentes;
- Adicionar novos pacientes através do formulário `CreateEditFormPacientes`, com campos como nome, morada, NIF, data de nascimento, contacto, código postal e historial clínico;
- Editar os dados de um paciente selecionado;
- Remover pacientes, com verificação de integridade e confirmação prévia por parte do utilizador.

Todos os dados são atualizados dinamicamente na interface após cada operação, assegurando a consistência visual da aplicação.

Funcionalidades do Formulário de Utilizadores

O formulário `Utilizadores.cs` apresenta uma listagem de todos os utilizadores da clínica (médicos, enfermeiros e rececionistas), permitindo:

- Visualizar todos os registos;
- Adicionar novos utilizadores;
- Editar os dados dos utilizadores existentes;
- Remover utilizadores, com verificação da existência de dependências (por exemplo, se o utilizador está associado a consultas ou recibos).

Este formulário tem também ligação com o formulário `CreateEditFormsUtils`, que segue o mesmo modelo do formulário de pacientes.

Considerações Finais

Embora não tenha sido o foco central do projeto, a criação desta interface serviu para ilustrar a integração entre a base de dados e a aplicação de front-end, bem como validar a eficácia dos procedimentos, *views* e restrições definidos. Caso o sistema fosse expandido, poderiam ser adicionadas novas interfaces específicas para rececionistas, médicos ou pacientes, com diferentes níveis de acesso e permissões.

Conclusão

O desenvolvimento deste projeto permitiu aplicar, de forma prática, os conhecimentos adquiridos ao longo da unidade curricular de Bases de

Dados, através da conceção e implementação de um sistema de gestão de base de dados orientado às necessidades de uma clínica dentária.

Durante o processo, foi possível compreender a importância de uma estrutura de dados bem planeada, capaz de garantir a integridade, coerência e segurança da informação clínica e administrativa. A base de dados desenvolvida suporta funcionalidades cruciais como o registo de pacientes, a marcação e gestão de consultas, o acompanhamento do estado dentário, a faturação automatizada e a associação a seguros de saúde — elementos fundamentais no funcionamento diário de uma clínica.

Além disso, foram implementados mecanismos de validação e automação com recurso a *Stored Procedures*, *Triggers*, *Views* e *User Defined Functions*, que não só simplificaram a manipulação de dados como asseguraram regras de negócio essenciais, como por exemplo, a verificação da idade para atribuição de seguros infantis.

Embora o foco principal do projeto tenha sido a estrutura da base de dados, foi ainda desenvolvida uma interface gráfica funcional, com o objetivo de ilustrar a integração entre o sistema de gestão e a camada de apresentação. Esta interface permitiu validar operações de inserção, edição e remoção de dados, promovendo uma experiência mais próxima de um ambiente real de utilização.

Em suma, os objetivos inicialmente propostos foram atingidos com sucesso. O projeto revelou-se uma oportunidade para consolidar os fundamentos teóricos da disciplina, demonstrando, na prática, o papel central que um Sistema de Gestão de Base de Dados desempenha na organização, eficiência e fiabilidade da informação numa unidade de saúde como uma clínica dentária.