

BANCO DE DADOS RELACIONAL: PROJETO **HOSPITAL UNICLASS**

Cássia Silva de Jesus
Pedro Lucas Mendes de Melo
Francine Dos Santos
João Batista
Diogo Menezes Figueredo

RESUMO

Este paper tem como objetivo fornecer uma análise abrangente do banco de dados relacional, abordando sua importância, estrutura básica e funcionalidades. Serão discutidos, conceitos fundamentais, como tabelas, chave primaria e estrangeira, normalização, consultas SQL e transações, e as stored procedure. Serão apresentados os benefícios e as limitações do modelo relacional, bem como as condições futuras de pesquisa nessa área.

1. Introdução

Este trabalho foi desenvolvido através de diversas análises e reflexões para cadeira Seminário Interdisciplinar – Banco de Dados Relacional – Projeto Hospital Uniclass do curso de Análise e Desenvolvimento de Sistemas da Universidade UNIASSELVI, cuja intenção é apresentar primeiramente o conceito de Banco de dados e seu modelo Relacional referente a aplicação para uma Clínica Hospitalar, as informações contidas nesse trabalho são sustentadas por algumas referencias bibliográficas. Através desse trabalho, foi possível analisar a importância da utilização do modelo relacional de banco de dados, pensar em um sistema cujo seu principal problema tenha relação com seu armazenamento de informações, que nos faz pensar a importância que o modelo relacional possui atualmente. Com isso é possível gerar uma discussão sobre a possibilidade de avanços impulsionados pelos estudos como o de Banco de Dados

1 Acadêmicos: Diogo Menezes Figueredo, Pedro Lucas Mendes de Melo, Francine dos Santos, Cássia Silva de Jesus

2 Professor: Fábio Teixeira da Costa

Centro Universitário Leonardo da Vinci – UNIASSELVI - FLC10467ADS – Banco dados Relacional - 16/06/2023

1.1 Contextualização e Importância do Banco de Dados Relacional

No mundo atual, a quantidade de dados gerados e armazenados tem crescido exponencialmente em todas as esferas da sociedade. Empresas, instituições governamentais, organizações de pesquisa e até mesmo indivíduos acumulam volumes massivos de informações. Diante desse cenário, surge a necessidade de um sistema que possa estruturar, armazenar e gerenciar esses dados de maneira eficiente. É nesse contexto que o banco de dados relacional se destaca como uma solução amplamente adotada.

O banco de dados relacional é um modelo de banco de dados baseado em tabelas inter-relacionadas, em que a informação é organizada em linhas e colunas. Foi proposto inicialmente por Edgar Codd na década de 1970 e rapidamente ganhou popularidade devido à sua eficácia na estruturação e recuperação de dados.

Uma das principais razões para a adoção generalizada do banco de dados relacional é sua capacidade de manter a integridade dos dados. Com o uso de chaves primárias e estrangeiras, é possível estabelecer relações entre as tabelas, garantindo a consistência dos dados em todo o sistema. Essa abordagem estruturada permite evitar redundância e inconsistência de informações, tornando o banco de dados mais confiável e preciso.

Além disso, o banco de dados relacional oferece flexibilidade na manipulação dos dados. Através da linguagem SQL (Structured Query Language), é possível realizar consultas complexas que permitem buscar, filtrar e extrair informações específicas de forma eficiente. Essa capacidade de consulta sofisticada é fundamental para a tomada de decisões informadas e estratégicas.

Outra vantagem do banco de dados relacional é sua capacidade de escalabilidade. À medida que os dados aumentam em volume, o modelo relacional pode se adaptar e acomodar essa expansão sem comprometer o desempenho. Isso permite que as organizações lidem com conjuntos de dados cada vez maiores, sem comprometer a eficiência das operações.

Para as organizações, o banco de dados relacional é uma ferramenta essencial para o sucesso e a competitividade. Ele oferece suporte à gestão eficiente de informações, facilitando a tomada de decisões estratégicas e fornecendo insights valiosos sobre os negócios. Além disso, o uso adequado de um banco de dados relacional promove a colaboração entre diferentes departamentos e equipes, permitindo o compartilhamento seguro de dados entre diferentes áreas da organização.

Embora existam outras abordagens de bancos de dados, como os bancos de dados não relacionais (NoSQL), o banco de dados relacional continua sendo amplamente adotado devido à sua comprovada eficiência e flexibilidade. Embora existam casos específicos em que outros modelos possam ser mais adequados, o banco de dados relacional continua sendo a base sólida para o gerenciamento eficiente de dados em diversas áreas.

Em resumo, o banco de dados relacional desempenha um papel fundamental no gerenciamento e organização de dados em ambientes modernos. Sua estruturação lógica, capacidade de manutenção da integridade dos dados, flexibilidade na manipulação e escalabilidade são fatores.

1.2 Objetivos do Paper

O objetivo deste projeto é desenvolver uma aplicação interativa para a clínica médica utilizando o Visual Studio e o framework Windows Forms, que se integra a um banco de dados relacional para armazenar e gerenciar informações sobre pacientes, consultas e médicos. A aplicação será projetada de forma a fornecer uma interface amigável e intuitiva para os usuários, permitindo que eles realizem operações de adição, exclusão, consulta e alteração de dados de forma eficiente.

Os principais objetivos do projeto são os seguintes:

- Desenvolver uma aplicação visualmente atraente e fácil de usar, utilizando o ambiente de desenvolvimento Visual Studio e o framework Windows Forms. Isso permitirá criar uma interface gráfica interativa para a clínica médica, proporcionando uma experiência agradável aos usuários.

- Implementar a lógica de negócio necessária para interagir com o banco de dados relacional SQL Server. A aplicação permitirá o cadastro de novos pacientes, agendamento de consultas e gerenciamento de informações dos médicos, tudo isso utilizando as funcionalidades do banco de dados relacional.
- Integrar a aplicação com o banco de dados relacional SQL Server, garantindo a persistência e integridade dos dados. Será criada uma conexão com o banco de dados para realizar operações de inserção, exclusão, consulta e atualização dos registros, de acordo com as necessidades da clínica médica.
- Implementar funcionalidades que permitam a interação direta com os usuários. Isso inclui a exibição de informações detalhadas sobre pacientes, consultas e médicos, bem como a capacidade de adicionar novos registros, atualizar informações existentes e excluir registros quando necessário.
- Garantir a segurança dos dados e a integridade referencial por meio da utilização adequada de chaves primárias e estrangeiras, conforme descrito na seção anterior. Isso assegurará a consistência dos dados e evitará problemas de redundância ou inconsistência.

Ao final do projeto, espera-se ter uma aplicação completa e funcional, desenvolvida com o Visual Studio e o framework Windows Forms, que permita à clínica médica gerenciar efetivamente os dados relacionados a pacientes, consultas e médicos. A integração com o banco de dados relacional SQL Server possibilitará a manipulação segura e eficiente das informações, proporcionando uma ferramenta valiosa para a equipe médica e administrativa da clínica.

2. Fundamentação Teórica

Nesta seção, será apresentada a fundamentação teórica para o desenvolvimento do Projeto Hospital Uniclass utilizando o Visual Studio, o framework Windows Forms e a integração com um banco de dados relacional. Serão abordados os conceitos relacionados a cada uma dessas tecnologias, destacando sua importância e benefícios no contexto do projeto.

- **Visual Studio:** O Visual Studio é um ambiente de desenvolvimento integrado (IDE) amplamente utilizado para criar aplicativos Windows, web e móveis. Ele oferece uma variedade de recursos e ferramentas que simplificam o processo de desenvolvimento, como a criação de interfaces gráficas, edição de código, depuração e teste. No Projeto Hospital Uniclass, o Visual Studio será a base para a criação da aplicação interativa, proporcionando uma interface amigável e recursos avançados de desenvolvimento.
- **Windows Forms:** O Windows Forms é um framework de desenvolvimento de aplicativos desktop para a plataforma Windows. Ele permite a criação de interfaces gráficas usando uma abordagem baseada em formulários, onde os elementos de interface, como botões, caixas de texto e listas, podem ser arrastados e soltos em um formulário para criar a interface do usuário. O Windows Forms oferece suporte a eventos, controles personalizados, validação de entrada e outras funcionalidades que tornam a criação de aplicativos interativos mais eficiente e produtiva. No Projeto Hospital Uniclass, o Windows Forms será utilizado para criar a interface interativa com os usuários, permitindo que eles realizem diversas operações de forma intuitiva.
- **Banco de Dados Relacional:** Um banco de dados relacional é uma coleção organizada de dados estruturados em tabelas, onde as informações são armazenadas de forma consistente e relacionada. Ele usa a teoria de conjuntos e álgebra relacional para manipular os dados de maneira eficiente. No Projeto Hospital Uniclass, o banco de dados relacional será utilizado para armazenar e gerenciar informações sobre pacientes, consultas e médicos. Ele oferece

vantagens como a capacidade de realizar consultas complexas, integridade referencial, segurança de dados e escalabilidade.

- **Integração do Banco de Dados com a Aplicação:** A integração do banco de dados com a aplicação é um aspecto crítico do Projeto Hospital Uniclass. Ela permite que a aplicação se conecte ao banco de dados, execute consultas, recupere e atualize registros, garantindo a persistência e a integridade dos dados. No projeto, será utilizada a linguagem de consulta estruturada (SQL) para interagir com o banco de dados relacional SQL Server. Serão stored procedures para realizar operações como inserção, exclusão, consulta e atualização de registros, proporcionando uma interação eficiente e segura com o banco de dados.

A combinação do Visual Studio, Windows Forms e um banco de dados relacional proporciona uma plataforma sólida para o desenvolvimento do Projeto Hospital Uniclass. O Visual Studio oferece uma interface de desenvolvimento amigável e recursos avançados, enquanto o Windows Forms permite a criação de interfaces gráficas interativas. A integração com o banco de dados relacional traz a capacidade de armazenar e gerenciar dados de forma estruturada

2.1 Conceitos Básicos do Banco de Dados Relacional

No contexto do projeto de aplicação para uma clínica hospitalar utilizando o banco de dados relacional SQL Server, é importante compreender os conceitos básicos do modelo relacional. A seguir, são apresentados os principais conceitos relacionados ao banco de dados relacional:

- **Tabelas:** As tabelas são a estrutura fundamental de um banco de dados relacional. Elas são compostas por linhas e colunas, onde cada linha representa um registro ou instância de dados, e cada coluna representa um atributo ou campo de dados. No projeto proposto, são definidas tabelas como "Pacientes", "Consultas" e "Médicos" para armazenar informações específicas sobre cada entidade.

- **Chaves Primárias:** A chave primária é um atributo (ou uma combinação de atributos) que identifica unicamente cada registro em uma tabela. Ela garante a unicidade dos registros e é fundamental para estabelecer relacionamentos entre tabelas. No projeto, pode ser utilizada uma chave primária, por exemplo, para identificar exclusivamente cada paciente, consulta e médico.
- **Chaves Estrangeiras:** As chaves estrangeiras são atributos em uma tabela que estabelecem uma relação com a chave primária de outra tabela. Elas são usadas para criar relacionamentos entre tabelas e garantir a integridade referencial dos dados. No projeto, as chaves estrangeiras podem ser utilizadas para associar um paciente a uma consulta ou relacionar uma consulta a um médico.
- **Relacionamentos:** Os relacionamentos são as conexões estabelecidas entre as tabelas em um banco de dados relacional. Eles definem a maneira como as informações estão interligadas. No projeto, podem ser estabelecidos relacionamentos, como um paciente associado a várias consultas ou um médico responsável por várias consultas.
- **Linguagem SQL:** A Structured Query Language (SQL) é uma linguagem de programação utilizada para realizar operações de consulta, inserção, atualização e exclusão de dados em um banco de dados relacional. No projeto, a linguagem SQL será utilizada para criar tabelas, definir restrições, inserir registros e executar consultas para obter informações relevantes sobre pacientes, consultas e médicos.
- **Stored Procedure:** Uma stored procedure é um conjunto de instruções SQL pré-compiladas no banco de dados e podem ser chamados e executados posteriormente. No contexto do projeto de aplicação para a clínica hospitalar, é utilizado stored procedures, para realizar tarefas como a inserção de novos registros de pacientes, consultas e médicos, a atualização de informações, a geração de relatórios, e a execução de consultas complexas, envolvendo múltiplas tabelas, podendo também fazer exclusão de dados se caso for necessário. A adoção de stored procedure traz benefícios significativos em termos de reutilização de códigos, desempenho, segurança e manutenção

centralizada. Essa abordagem contribui para a eficiência e a qualidade da aplicação, permitindo um melhor controle e gerenciamento dos processos relacionados ao banco de dados no contexto da

- **Integridade dos Dados:** A integridade dos dados refere-se à qualidade e consistência dos dados armazenados em um banco de dados. O modelo relacional oferece mecanismos para garantir a integridade dos dados, como a definição de chaves primárias, restrições de integridade e relacionamentos entre tabelas. No projeto, a integridade dos dados será mantida por meio desses mecanismos, evitando redundâncias e inconsistências nos registros.

Ao compreender e aplicar esses conceitos básicos do banco de dados relacional no projeto de aplicação para uma clínica hospitalar, é possível estruturar as informações de forma organizada, garantir a consistência dos dados, realizar consultas eficientes e estabelecer relacionamentos adequados entre as entidades envolvidas. Isso resultará em um sistema de gerenciamento de dados eficiente e confiável para a clínica.

2.2 Tabelas e Relacionamentos

Nesta seção, será abordada a estrutura das tabelas e os relacionamentos no projeto de aplicação para a clínica hospitalar utilizando o banco de dados relacional SQL Server. Essa estrutura permite organizar e armazenar as informações relevantes sobre pacientes, consultas e médicos de forma eficiente e coerente. A seguir, são apresentadas as tabelas principais e os relacionamentos propostos:

Tabela "Pacientes": A tabela "Pacientes" armazena os dados relacionados aos pacientes da clínica hospitalar. As principais colunas podem incluir:

- ID do paciente (chave primária): um identificador único para cada paciente.
- Nome: o nome completo do paciente.
- Sexo: sendo assim definido como masculino ou feminino.
- Endereço: o endereço residencial do paciente.
- Bairro: o bairro referente ao endereço do paciente.
- Número: número da residência do paciente.
- Cidade: cidade aonde.
- CEP: respectivo cep do endereço do paciente.

- Celular e Telefone: informações de contato do paciente.

	Nome	Tipo de Dados
PK	IDpaciente	int
	NomePaciente	varchar(100)
	Endereco	varchar(200)
	Numero	int
	Bairro	varchar(50)
	Cidade	varchar(50)
	Cep	varchar(20)
	Telefone	varchar(20)
	Calular	varchar(20)
	Sexo	varchar(10)

Fonte: Autor

Tabela "Consultas": A tabela "Consultas" registra as informações sobre as consultas agendadas na clínica hospitalar. As principais colunas podem incluir:

- ID da consulta (chave primária): um identificador único para cada consulta.
- Data e hora: a data e o horário em que a consulta está agendada.
- ID do paciente (chave estrangeira): uma referência ao paciente associado à consulta.
- ID do médico (chave estrangeira): uma referência ao médico responsável pela consulta.
- Nome do Paciente: nome do paciente com a consulta agendada.
- Nome do Médico: nome do médico que está agendado a consulta.
- Retorno: retorno se o respectivo paciente será seu retorno sua primeira consulta.

	Nome	Tipo de Dados
PK	IDconsulta	int
	NomeMedico	varchar(100)
	NomePaciente	varchar(100)
	Data	varchar(20)
	Horario	varchar(20)
	Retorno	varchar(10)
	idMedico	int
	idPaciente	int

Tabela "Médicos": A tabela "Médicos" contém os dados dos médicos que atuam na clínica hospitalar. As principais colunas podem incluir:

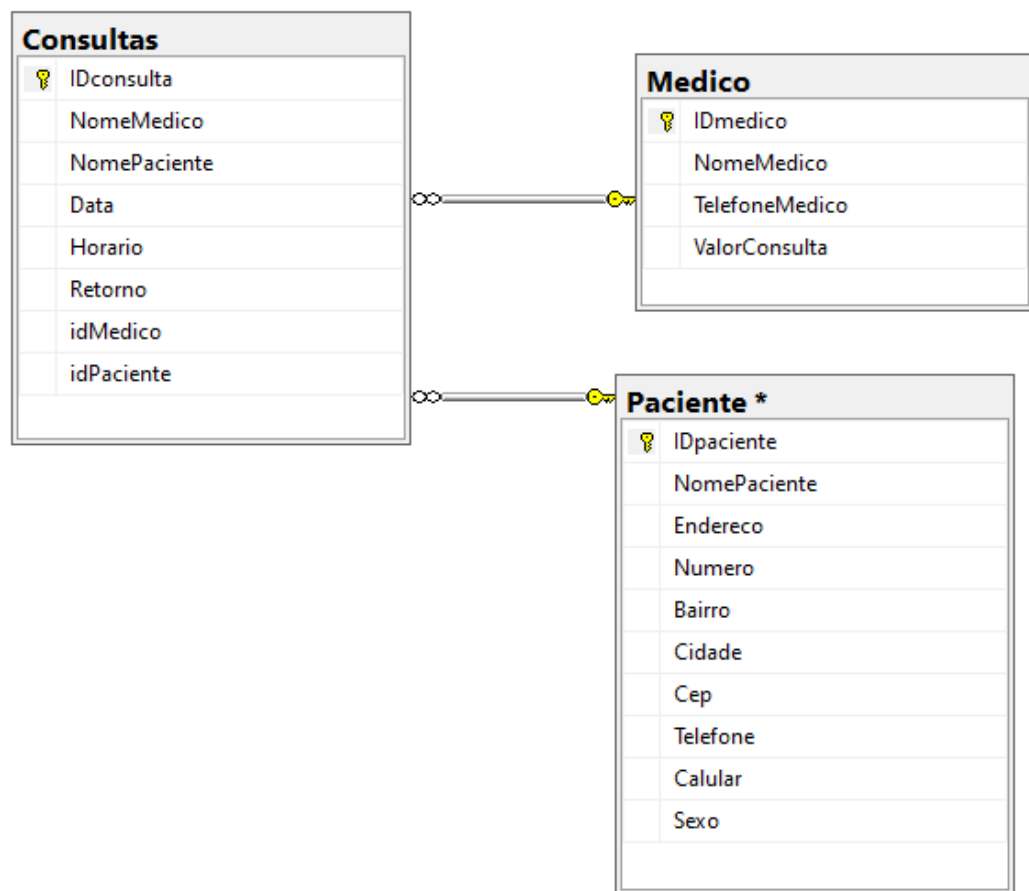
- ID do médico (chave primária): um identificador único para cada médico.
- Nome: o nome completo do médico.
- Contato: informações de contato do médico.
- Valor da Consulta: valor da consulta desse respectivo médico.

	Nome	Tipo de Dados
PK	IDmedico	int
	NomeMedico	varchar(100)
	TelefoneMedico	varchar(20)
	ValorConsulta	varchar(20)

Fonte: Autor

Relacionamentos entre as tabelas:

- **Relacionamento entre "Pacientes" e "Consultas":** Para associar um paciente a uma consulta, é estabelecido um relacionamento de chave estrangeira na tabela "Consultas" que referência o ID do paciente da tabela "Pacientes". Isso permite que cada consulta seja vinculada a um paciente específico.
- **Relacionamento entre "Consultas" e "Médicos":** Para atribuir um médico a uma consulta, é estabelecido um relacionamento de chave estrangeira na tabela "Consultas" que referência o ID do médico da tabela "Médicos". Isso permite que cada consulta seja atribuída a um médico responsável.



Fonte: Autor

Essa estrutura de tabelas e relacionamentos oferece uma representação adequada das entidades envolvidas na clínica hospitalar. Ela permite a organização eficiente dos dados, garantindo a integridade e a consistência das informações armazenadas. Os relacionamentos estabelecidos entre as tabelas possibilitam a realização de consultas complexas e a recuperação de informações relevantes para o gerenciamento adequado da clínica.

2.2 Normalização para o Projeto Hospital Uniclass

A normalização é um processo fundamental no design de um banco de dados relacional, visando eliminar redundâncias e inconsistências, garantindo a integridade dos dados e facilitando a manipulação e recuperação das informações. No contexto do Projeto Hospital Uniclass, a normalização será aplicada para estruturar as tabelas de forma

eficiente e reduzir a redundância dos dados. A seguir, são apresentadas as etapas de normalização aplicadas ao projeto:

- **Primeira Forma Normal (1NF):** A primeira forma normal exige que cada coluna contenha apenas valores atômicos, ou seja, não deve haver múltiplos valores agrupados em uma única coluna. No Projeto Hospital Uniclass, a primeira forma normal já é atendida, uma vez que cada tabela possui colunas com valores individuais e não há agrupamento de múltiplos valores.
- **Segunda Forma Normal (2NF):** A segunda forma normal requer que cada coluna que não seja parte da chave primária seja totalmente dependente da chave primária. Isso significa que não deve haver dependências parciais nas tabelas. No Projeto Hospital Uniclass, as tabelas "Consultas" e "Médicos" estão em conformidade com a segunda forma normal, pois todas as colunas dessas tabelas são dependentes da chave primária correspondente.
- **Terceira Forma Normal (3NF):** A terceira forma normal estabelece que não deve haver dependências transitivas nas tabelas. Ou seja, se uma coluna depende de outra coluna que, por sua vez, depende da chave primária, é necessário separá-las em tabelas distintas. No Projeto Hospital Uniclass, a tabela "Consultas" contém uma dependência transitiva entre o ID do paciente e o ID do médico. Para resolver isso, podemos criar uma nova tabela chamada "Agendamento" que associa o ID do paciente e o ID do médico, evitando assim a dependência transitiva.

Após a aplicação dessas etapas de normalização, o Projeto Hospital Uniclass estará em conformidade com as formas normais, o que resulta em um modelo de dados mais coeso, reduzindo a redundância e garantindo a integridade dos dados. A normalização também facilita a manutenção e a expansão do banco de dados, proporcionando uma estrutura sólida e eficiente para o armazenamento e recuperação das informações relacionadas aos pacientes, consultas e médicos da clínica médica.

2.3 Chave Primária e Chave Estrangeira

Nesta seção, serão apresentados os conceitos de chaves primárias e chaves estrangeiras e sua aplicação no projeto de aplicação para o Hospital Uniclass utilizando o banco de dados relacional SQL Server. Esses conceitos são fundamentais para estabelecer a integridade referencial dos dados e garantir a consistência das informações armazenadas. A seguir, são descritas as chaves primárias e estrangeiras para cada tabela do projeto:

- **Tabela "Pacientes":**

Chave Primária: A chave primária da tabela "Pacientes" é o campo "IDpaciente". Essa chave é única para cada registro na tabela e permite identificar de forma exclusiva cada paciente cadastrado.

PRIMARY KEY (IDpaciente)

- **Tabela "Consultas":**

Chave Primária: A chave primária da tabela "Consultas" é o campo "IDconsulta". Cada registro nessa tabela possui um ID único, o que possibilita identificar e diferenciar cada consulta agendada.

PRIMARY KEY (IDconsulta)

- **Chave Estrangeira: A tabela "Consultas" possui duas chaves estrangeiras:**

"IDpaciente": Essa chave estrangeira faz referência à tabela "Pacientes" e está relacionada ao campo "ID do paciente". Ela estabelece a conexão entre uma consulta e o paciente associado a ela.

"IDmedico": Essa chave estrangeira faz referência à tabela "Médicos" e está relacionada ao campo "ID do médico". Ela estabelece a conexão entre uma consulta e o médico responsável por ela.

CONSTRAINT FK_MEDICO

FOREIGN KEY (IDmedico)

REFERENCES MEDICO(IDmedico)

CONSTRAINT FK_PACIENTE

FOREIGN KEY (IDpaciente)

REFERENCES PACIENTE(IDpaciente)

Tabela "Médicos":

Chave Primária: A chave primária da tabela "Médicos" é o campo "IDmédico". Cada médico cadastrado na clínica possui um ID único, permitindo a identificação exclusiva de cada profissional.

PRIMARY KEY (IDconsulta)

No projeto, as chaves primárias e estrangeiras são utilizadas para estabelecer relacionamentos entre as tabelas e garantir a integridade referencial dos dados. Ao definir uma chave estrangeira, é possível criar um vínculo entre registros em tabelas diferentes, permitindo consultas e operações relacionais.

Por exemplo, a chave estrangeira "ID do paciente" na tabela "Consultas" estabelece uma relação entre uma consulta específica e o paciente associado a ela. Isso permite a recuperação de informações sobre as consultas de um paciente em particular por meio de consultas SQL.

O uso adequado de chaves primárias e estrangeiras ajuda a manter a consistência dos dados, evitando registros duplicados ou inconsistentes. Além disso, esses mecanismos auxiliam na realização de operações de atualização, exclusão e consulta, contribuindo para um sistema de gerenciamento de dados eficiente e confiável na clínica hospitalar.

2.4 Stored Procedure

Nesta seção, serão apresentadas as stored procedures para manipulação de dados no Projeto Hospital Uniclass utilizando o banco de dados relacional SQL Server. Cada tabela do projeto, incluindo "Pacientes", "Consultas" e "Médicos", terá seus próprios procedimentos para adicionar, excluir, consultar e alterar registros. Essas stored

procedures facilitam a interação com o banco de dados e fornecem uma abordagem estruturada para manipular os dados. A seguir, são descritas as stored procedures para cada tabela:

- **Stored Procedures para a tabela "Pacientes":**

Inserir Paciente: Essa stored procedure permite adicionar um novo paciente à tabela "Pacientes" com base nos dados fornecidos. Ela recebe como parâmetros os detalhes do paciente, como nome, sexo, endereço e informações de contato.

```
CREATE PROCEDURE [dbo].[usPacienteInserir]
    @NomePaciente varchar(100),
    @Endereco varchar(200),
    @Numero int,
    @Bairro varchar(50),
    @Cidade varchar(50),
    @Cep varchar(20),
    @Telefone varchar(20),
    @Calular varchar(20),
    @Sexo varchar(10)
AS
BEGIN
    INSERT INTO Paciente
    (
        NomePaciente,
        Endereco,
        Numero,
        Bairro,
        Cidade,
        Cep,
        Telefone,
        Calular,
        Sexo
    )
    VALUES
    (
        @NomePaciente,
        @Endereco,
        @Numero,
        @Bairro,
        @Cidade,
        @Cep,
        @Telefone,
        @Calular,
        @Sexo
    )
    SELECT @@IDENTITY as Retorno
END
```

Fonte: Autor

Excluir Paciente: Essa stored procedure permite excluir um paciente específico da tabela "Pacientes" com base no ID do paciente fornecido como parâmetro.

```

[END
  @IDpaciente as resultado
[SELECT
  IDpaciente = @IDpaciente
  NOME
  SEXO = @SEXO
  Cpf = @Cpf
  Telefone = @Telefone
  Cep = @Cep
  Cidade = @Cidade
  Bairro = @Bairro
  Numero = @Numero
  Endereco = @Endereco
  NomePaciente = @NomePaciente
  ZEL
  bPaciente
  NOME
[BEGIN
  VZ
  @SEXO as resultado
  @Cpf as resultado
  @Telefone as resultado
  @Cep as resultado
  @Cidade as resultado
  @Bairro as resultado
  @Numero as resultado
  @Endereco as resultado
  @NomePaciente as resultado
  @IDpaciente as resultado
[CREATE PROCEDURE [dbo].[uspPacienteConsultarPorNome]

```

Fonte: Autor

Consultar Paciente por Nome: Essa stored procedure retorna informações detalhadas sobre um paciente específico com base no Nome do paciente fornecido como parâmetro.

```

[CREATE PROCEDURE [dbo].[uspPacienteConsultarPorNome]
  @NomePaciente varchar(100)
AS
[BEGIN
[SELECT
  IDpaciente,
  NomePaciente,
  Endereco,
  Numero,
  Bairro,
  Cidade,
  Cep,
  Telefone,
  Calular,
  Sexo
FROM
  Paciente
WHERE
  NomePaciente LIKE '%' + @NomePaciente + '%'
[END

```

Alterar Paciente: Essa stored procedure permite alterar os detalhes de um paciente existente na tabela "Pacientes". Ela recebe como parâmetros o ID do paciente e os novos dados a serem atualizados.


```

CREATE PROCEDURE [dbo].[uspPacienteAlterar]
    @IDpaciente int,
    @NomePaciente varchar(100),
    @Endereco varchar(200),
    @Numero int,
    @Bairro varchar(50),
    @Cidade varchar(50),
    @Cep varchar(20),
    @Telefone varchar(20),
    @Calular varchar(20),
    @Sexo varchar(10)
AS
BEGIN
    UPDATE
        Paciente
    SET
        NomePaciente = @NomePaciente,
        Endereco = @Endereco,
        Numero = @Numero,
        Bairro = @Bairro,
        Cidade = @Cidade,
        Cep = @Cep,
        Telefone = @Telefone,
        Calular = @Calular,
        Sexo = @Sexo
    WHERE
        IDpaciente = @IDpaciente
    SELECT
        @IDpaciente as Retorno
END

```

Fonte: Autor

- Stored Procedures para a tabela "Consultas":

Inserir Consultas: Essa stored procedure permite adicionar uma nova consulta à tabela "Consultas". Ela recebe como parâmetros os detalhes da consulta, como a data e hora, retorno e o ID do paciente associado e o ID do médico responsável.

```

CREATE PROCEDURE [dbo].[usConsultaInserir]
    @NomeMedico varchar(100),
    @NomePaciente varchar(100),
    @Data varchar(20),
    @Horario varchar(20),
    @Retorno varchar(10),
    @idMedico int,
    @idPaciente int
AS
BEGIN
    INSERT INTO Consultas
    (
        NomeMedico,
        NomePaciente,
        Data,
        Horario,
        Retorno,
        idMedico,
        idPaciente
    )
    VALUES
    (
        @NomeMedico,
        @NomePaciente,
        @Data,
        @Horario,
        @Retorno,
        @idMedico,
        @idPaciente
    )
    SELECT @@IDENTITY AS Retorno
END

```

Fonte: Autor

Excluir Consulta: Essa stored procedure permite excluir uma consulta específica da tabela "Consultas" com base no ID da consulta fornecido como parâmetro.

```
CREATE PROCEDURE [dbo].[usConsultaExcluir]
    @IDconsulta int
AS
BEGIN
    DELETE FROM
        Consultas
    WHERE
        IDconsulta = @IDconsulta
    SELECT
        @IDconsulta AS Retorno
END
```

Fonte: Autor

Consultar Consulta por ID: Essa stored procedure retorna informações detalhadas sobre uma consulta específica com base no ID da consulta fornecido como parâmetro.

```
CREATE PROCEDURE [dbo].[usConsultaConsultarPorID]
    @IDconsulta int
AS
BEGIN
    SELECT
        IDconsulta,
        NomeMedico,
        NomePaciente,
        Data,
        Horario,
        Retorno,
        idMedico,
        idPaciente
    FROM
        Consultas
    WHERE
        IDconsulta = @IDconsulta
END
```

Fonte: Autor

Alterar Consulta: Essa stored procedure permite alterar os detalhes de uma consulta existente na tabela "Consultas". Ela recebe como parâmetros o ID da consulta e os novos dados a serem atualizados.

```

CREATE PROCEDURE [dbo].[usConsultaAlterar]
    @IDconsulta int,
    @NomeMedico varchar(100),
    @NomePaciente varchar(100),
    @Data varchar(20),
    @Horario varchar(20),
    @Retorno varchar(10),
    @idMedico int,
    @idPaciente int
AS
BEGIN
    UPDATE
        Consultas
    SET
        NomeMedico = @NomeMedico,
        NomePaciente = @NomePaciente,
        Data = @Data,
        Horario = @Horario,
        Retorno = @Retorno,
        idMedico = @idMedico,
        idPaciente = @idPaciente
    WHERE
        IDconsulta = @IDconsulta

    SELECT
        @IDconsulta AS Retorno
END

```

Fonte: Autor

- Stored Procedures para a tabela "Médicos":

Inserir Médico: Essa stored procedure permite adicionar um novo médico à tabela "Médicos" com base nos dados fornecidos. Ela recebe como parâmetros os detalhes do médico, como nome, especialidade e informações de contato.

```

CREATE PROCEDURE [dbo].[proMedicoInserir]
    @NomeMedico varchar(100),
    @TelefoneMedico varchar(20),
    @ValorConsulta varchar(20)
AS
BEGIN
    INSERT INTO Medico
    (
        NomeMedico,
        TelefoneMedico,
        ValorConsulta
    )
    VALUES
    (
        @NomeMedico,
        @TelefoneMedico,
        @ValorConsulta
    )
    SELECT @@IDENTITY AS Retorno
END

```

Fonte: Autor

Consultar Médico por Nome: Essa stored procedure retorna informações detalhadas sobre um médico específico com base no nome do médico fornecido como parâmetro.

```

CREATE PROCEDURE [dbo].[uspMedicoConsultarPorNome]
    @NomeMedico varchar(100)
AS
BEGIN
    SELECT
        IDmedico,
        NomeMedico,
        TelefoneMedico,
        ValorConsulta
    FROM
        Medico
    WHERE
        NomeMedico LIKE '%' + @NomeMedico + '%'
END

```

Fonte: Autor

Alterar Médico: Essa stored procedure permite alterar os detalhes de um médico existente na tabela "Médicos". Ela recebe como parâmetros o ID do médico e os novos dados a serem atualizados.

```

CREATE PROCEDURE [dbo].[uspMedicoAlterar]
    @IDmedico int,
    @NomeMedico nchar(100),
    @TelefoneMedico varchar(20),
    @ValorConsulta varbinary(20)
AS
BEGIN
    UPDATE
        Medico
    SET
        NomeMedico = @IDmedico,
        TelefoneMedico = @TelefoneMedico,
        ValorConsulta = @ValorConsulta
    WHERE
        IDmedico = @IDmedico
    SELECT
        @IDmedico AS Retorno
END

```

Fonte: Autor

Cada uma dessas stored procedures oferece uma interface para manipular os dados nas respectivas tabelas do Projeto Hospital Uniclass. Elas podem ser chamadas e executadas por meio de comandos SQL, permitindo a adição, exclusão, consulta e alteração de registros de forma estruturada e segura.

Ao utilizar essas stored procedures, a equipe de desenvolvimento Hospital Uniclass pode se beneficiar de diversos aspectos, tais como:

Reutilização de código: As stored procedures podem ser chamadas em diferentes partes da aplicação, evitando a duplicação de código e facilitando a manutenção.

Segurança: Ao conceder permissões de acesso apenas às stored procedures necessárias, é possível controlar o acesso direto aos dados e garantir a segurança do banco de dados.

Desempenho: As stored procedures são pré-compiladas, o que resulta em um processamento mais rápido das consultas e operações no banco de dados.

Manutenção simplificada: Alterações na lógica de manipulação de dados podem ser feitas nas stored procedures, sem a necessidade de modificar a aplicação em si.

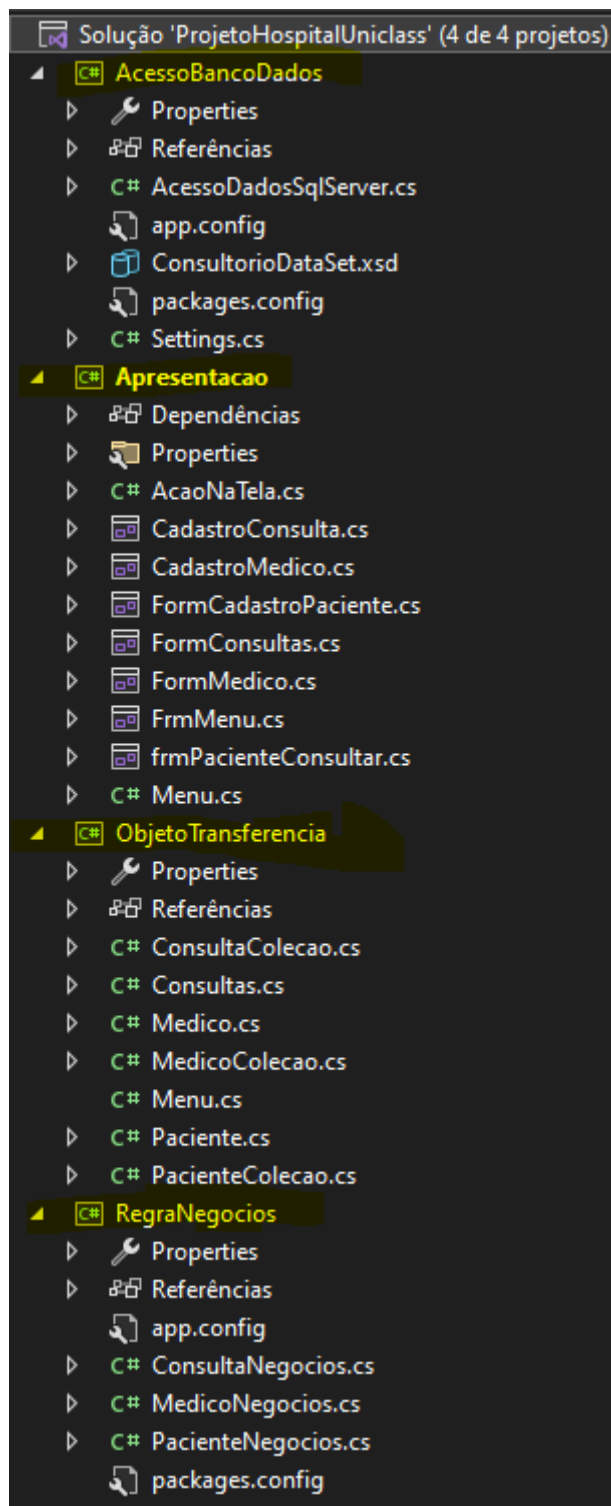
Com a implementação dessas stored procedures, o Projeto Hospital Uniclass se beneficia de um controle mais eficiente e estruturado das operações de adição, exclusão, consulta e alteração de registros. Além disso, a utilização de stored procedures contribui para a segurança, desempenho e facilidade de manutenção do banco de dados, promovendo uma aplicação robusta e confiável para a equipe médica e administrativa da clínica.

3. Hospital Uniclass com Windows Forms e Estrutura em Camadas

O Projeto Hospital Uniclass foi desenvolvido utilizando o Visual Studio e a linguagem C#, com o objetivo de criar uma aplicação interativa e de fácil utilização para gerenciar informações de pacientes, consultas e médicos. Para alcançar esse objetivo, foi adotada uma arquitetura em camadas, dividindo a aplicação em quatro camadas principais: acesso ao banco de dados, apresentação, objeto transferência e regras de negócio.

- **Camada de Acesso ao Banco de Dados:** A camada de acesso ao banco de dados é responsável por estabelecer a conexão com o banco de dados relacional (SQL Server) e realizar operações de leitura, gravação, atualização e exclusão de dados. Nessa camada, foram utilizadas técnicas de acesso a dados, como o uso de consultas SQL e o uso de objetos de acesso a dados, como o Entity Framework ou ADO.NET.
- **Camada de Apresentação (Windows Forms):** A camada de apresentação foi desenvolvida utilizando o Windows Forms, que é uma tecnologia de interface do usuário fornecida pelo .NET Framework. O Windows Forms permite a criação de interfaces gráficas interativas, onde os usuários podem interagir com a aplicação. Nessa camada, foram criadas as telas, formulários e controles necessários para exibir e capturar informações dos pacientes, consultas e médicos.
- **Camada de Objeto de Transferência:** A camada de objeto de transferência, também conhecida como DTO (Data Transfer Object), é responsável por transferir os dados entre as diferentes camadas da aplicação. Ela consiste em classes que representam os objetos de negócio, como as classes Paciente, Consulta e Médico. Essas classes encapsulam os dados relevantes e podem ser utilizadas para transportar informações entre as camadas.
- **Camada de Regras de Negócio:** A camada de regras de negócio contém a lógica e as regras que governam o funcionamento da aplicação. Nessa camada, são definidas as operações de validação, cálculos e processamento dos dados. Por exemplo, as regras que determinam se uma consulta pode ser agendada em

determinado horário, se um paciente já está cadastrado no sistema, entre outras. Essas regras garantem a consistência e a integridade dos dados na aplicação.



Fonte: Autor

A estrutura em camadas adotada no Projeto do Hospital Uniclass proporciona uma separação clara de responsabilidades e permite que cada camada seja desenvolvida e

testada de forma independente. Isso facilita a manutenção, a escalabilidade e a reutilização de código, além de promover uma arquitetura mais organizada e modular.

O uso do Windows Forms como tecnologia de interface do usuário oferece uma ampla variedade de recursos gráficos, como controles personalizáveis, eventos e gerenciamento de formulários, permitindo a criação de uma interface amigável e intuitiva para os usuários da aplicação do Hospital Uniclass.

Em suma, o Projeto Hospital Uniclass foi desenvolvido utilizando o Visual Studio com a linguagem C# e adotou uma arquitetura em camadas. O Windows Forms foi utilizado para criar a camada de apresentação, permitindo a criação de uma interface gráfica interativa e amigável para os usuários. As quatro camadas principais do projeto incluem a camada de acesso ao banco de dados, responsável por gerenciar as operações de leitura e escrita no banco de dados; a camada de apresentação, onde foram criados os formulários e controles visuais; a camada de objeto de transferência, que realiza a transferência de dados entre as camadas; e a camada de regras de negócio, onde são implementadas as regras e a lógica de funcionamento da aplicação.

A adoção dessa estrutura em camadas proporciona uma separação clara de responsabilidades, facilitando o desenvolvimento, a manutenção e o teste do projeto. Além disso, a utilização do Visual Studio e da linguagem C# oferece recursos poderosos e ferramentas integradas que auxiliam no desenvolvimento de aplicativos Windows de forma eficiente.

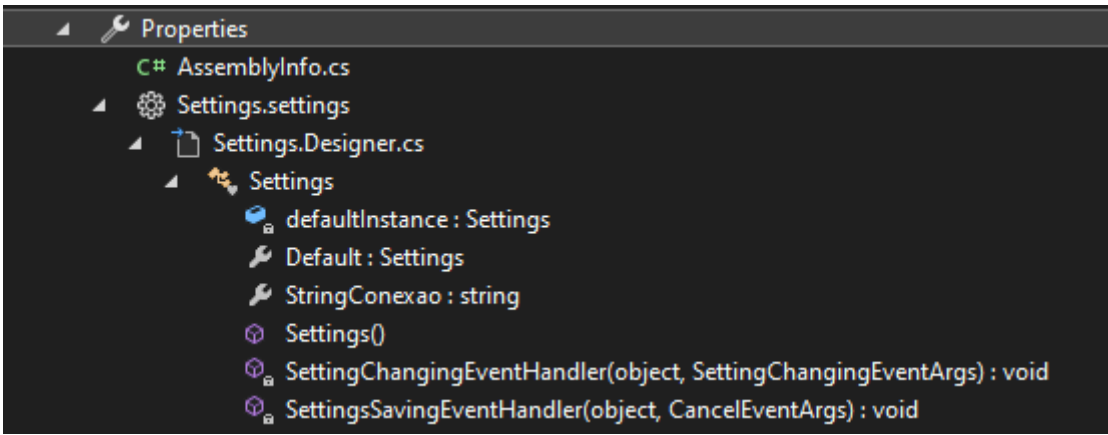
Com essa abordagem, o Hospital Uniclass conseguiu construir uma aplicação interativa e funcional, que permite o gerenciamento de informações de pacientes, consultas e médicos de forma eficiente e organizada. A estrutura em camadas, juntamente com o uso do Visual Studio e da linguagem C#, contribui para um código limpo, modular e de fácil manutenção, além de fornecer uma experiência agradável para os usuários da aplicação.

Esses conceitos básicos do projeto da clínica médica, incluindo o uso do Windows Forms e a estrutura em camadas, foram fundamentais para o sucesso e a qualidade da aplicação desenvolvida.

3.1 Camada de Acesso ao Banco de Dados

A camada de acesso ao banco de dados desempenha um papel crucial no Projeto Hospital Uniclass, permitindo a interação com o banco de dados relacional (SQL Server) e fornecendo as operações de leitura e gravação dos dados. Nessa camada, foram utilizadas as seguintes configurações e recursos:

Properties (propriedades): A camada de acesso ao banco de dados possui propriedades importantes, como a string de conexão e outras configurações relacionadas ao acesso aos dados. Essas propriedades são definidas e configuradas para estabelecer a conexão com o banco de dados e fornecer os parâmetros necessários para as operações de acesso aos dados.



Fonte: Autor

String de Conexão: A string de conexão é um elemento essencial na camada de acesso ao banco de dados. Ela contém as informações necessárias para estabelecer a conexão com o banco de dados, como o nome do servidor, o nome do banco de dados, as credenciais de autenticação e outras configurações específicas. A string de conexão é definida e configurada nas propriedades da camada de acesso ao banco de dados, permitindo que a aplicação se conecte ao banco de dados SQL Server.

	Nome	Tipo	Escopo	Valor
...	StringConexao	(Cadeia de conexão)	Aplicativo	Data Source=.\SQLEXPRESS;Initial Catalog=Consultorio;Integrated Security=True
*				

Fonte: Autor

Classe de Acesso aos Dados do SQL Server: Para interagir com o banco de dados SQL Server, foi criada uma classe de acesso aos dados. Essa classe é responsável por executar as consultas SQL, realizar operações de leitura e gravação no banco de dados e retornar os resultados necessários para as outras camadas da aplicação. A classe de acesso aos dados do SQL Server pode conter métodos para adicionar, atualizar, excluir e consultar registros no banco de dados, utilizando as funcionalidades disponíveis na linguagem C# e nas bibliotecas do .NET Framework.

```
namespace AcessoBancoDados
{
    public class AcessoDadosSqlServer
    {
        //cria conexão
        private SqlConnection CriarConexao()
        {
            return new SqlConnection(Settings.Default.StringConexao);
        }

        //parametros que vão para o banco
        private SqlParameterCollection parameterCollection = new SqlCommand().Parameters;

        public void LimparParametros()
        {
            parameterCollection.Clear();
        }

        public void AdicionaParametros(string NomeParametro, object ValorParametro)
        {
            parameterCollection.Add(new SqlParameter(NomeParametro, ValorParametro));
        }
    }
}
```

```
//inserir, alterar excluir dados
public object ExecutarManipulacao(CommandType commandType, string nomeProcedureOuTextoSql)
{
    try
    {
        //CRIAR A CONEXAO
        SqlConnection connection = CriarConexao();

        //ABRIR CONEXAO
        connection.Open();

        //CRIAR UM COMANDO
        SqlCommand command = connection.CreateCommand();

        //COLOCANDO AS COISAS DENTRO DO COMANDO, DENTRO DA CAIXA QUE VAI TRAFEGAR
        command.CommandType = commandType;
        command.CommandText = nomeProcedureOuTextoSql;
        command.CommandTimeout = 7200; //segundos=2h em execucao

        //ADICIONA OS PARAMETROS
        foreach (SqlParameter sqlParameter in parameterCollection)
        {
            command.Parameters.Add(new SqlParameter(sqlParameter.ParameterName, sqlParameter.Value));
        }

        //EXECUTAR O COMANDO OU SEJA MANDO O COMANDO IR NO BANCO DE DADOS
        return command.ExecuteScalar();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}
```

Fonte: Autor

```
public DataTable ExecutaConsulta(CommandType commandType, string nomePocedureOuTextoSql)
{
    //CONSULTAR REGISTRO NO BANCO DE DADOS
    try
    {
        SqlConnection connection = CriarConexao();
        connection.Open();
        SqlCommand sqlCommand = connection.CreateCommand();
        sqlCommand.CommandType = commandType;
        sqlCommand.CommandText = nomePocedureOuTextoSql;
        sqlCommand.CommandTimeout = 7200;

        //ADICIONAR PARAMETROS NO COMANDO
        foreach (SqlParameter sqlParameter in parameterCollection)
        {
            sqlCommand.Parameters.Add(new SqlParameter(sqlParameter.ParameterName, sqlParameter.Value));
        }

        //CRIAR ADAPTADOR
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand);

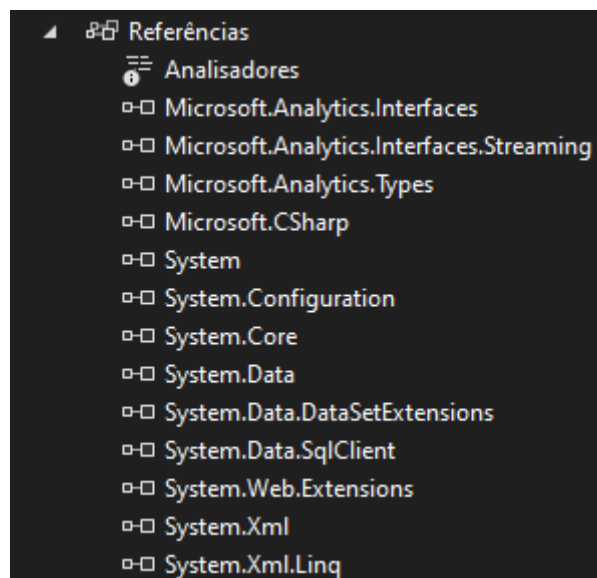
        //data table que é uma tabela de dados vazia
        //AONDE QUE VOU COLOCAR OS DADOS QUE VEM DO BANCO DE DADOS
        DataTable dataTable = new DataTable();

        //MANDA O COMANDO IR ATE O BANDO DE DADOS PARA PREENCHER ESSA TABELA VAZIA
        sqlDataAdapter.Fill(dataTable);

        return dataTable;
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}
```

Fonte: Autor

Referências: No Projeto Hospital Uniclass, foram adicionadas as referências necessárias para utilizar as funcionalidades de acesso ao banco de dados SQL Server. Isso inclui referências a bibliotecas e assemblies específicos, como o System.Data.SqlClient, que fornece os recursos necessários para estabelecer a conexão com o banco de dados e executar as consultas SQL. Além disso, pode ter sido utilizado o ConsultorioDataSet.xsd, um conjunto de dados que contém definições e estruturas das tabelas do banco de dados, permitindo uma integração eficiente e consistente entre as camadas da aplicação.



Fonte: Autor

A camada de acesso ao banco de dados, juntamente com as outras camadas da arquitetura em camadas adotada no Projeto Uniclass, contribui para um sistema robusto, escalável e de fácil manutenção. Através da separação de responsabilidades e da utilização de boas práticas de desenvolvimento, é possível obter uma aplicação eficiente, que atende às necessidades dos usuários e garante a segurança e a integridade dos dados.

Com essa estrutura bem definida e a utilização do Visual Studio com a linguagem C#, foi possível criar uma aplicação interativa e funcional para a gestão de informações na

clínica médica, proporcionando uma experiência positiva para os usuários e agilizando os processos internos da instituição.

3.2 Camada de Apresentação

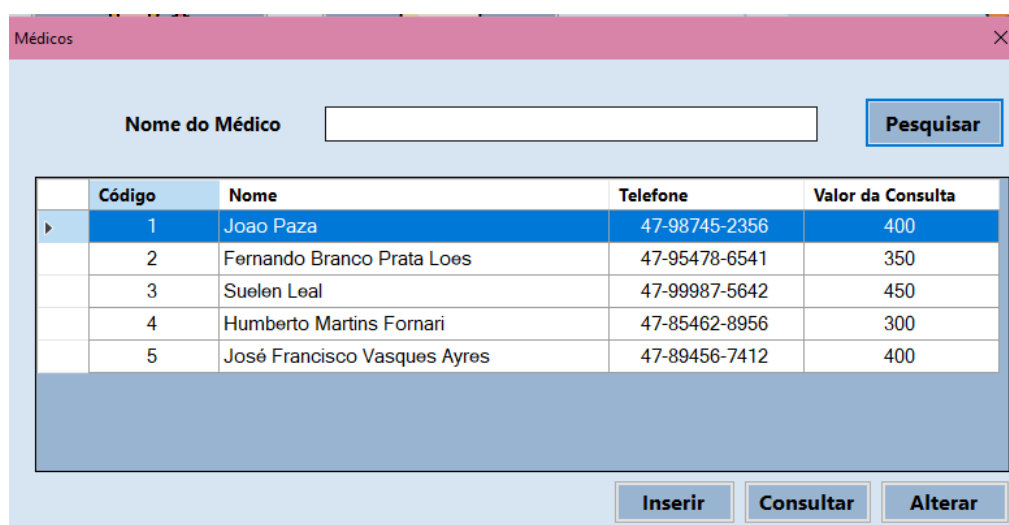
A seguir, está uma descrição detalhada da camada de apresentação do projeto Hospital Uniclass, que consiste em sete formulários: Menu, Consulta, Inserir Consulta, Médico, Inserir Médico, Paciente, Inserir Paciente. Cada formulário possui botões para salvar, cancelar, pesquisar, inserir, consultar, alterar e excluir registros. Vejamos em detalhes:

Menu: O formulário Menu é a tela principal da aplicação. Ele fornece a navegação entre os diferentes módulos da aplicação, permitindo que o usuário acesse as funcionalidades de cadastro e pesquisa. Nessa tela, o usuário pode selecionar qual formulário deseja acessar, com botão interativo que dá acesso aos Médicos, Pacientes e Consultas, sendo assim realizar suas operações correspondentes.



Fonte: Autor

Médico: O Formulário de Médicos possibilita ao usuário pesquisar médicos cadastrados no sistema. Ele oferece campos de filtro, onde o usuário pode estar clicando no botão pesquisar, os resultados são exibidos em uma lista, o nome dos médicos cadastrados, o telefone e o valor da consulta, podendo apenas fazer uma visualização, pois para fazer qualquer alteração, ou uma consulta detalhada, o usuário poderá clicar em qualquer botão que está visível no formulário o de alterar e de consultar, se caso queira inserir um novo médico, clicando no botão inserir, irá abrir um formulário em branco para a tal ação, sendo que os dados serão gravados diretamente no banco de dados relacionado ao projeto.

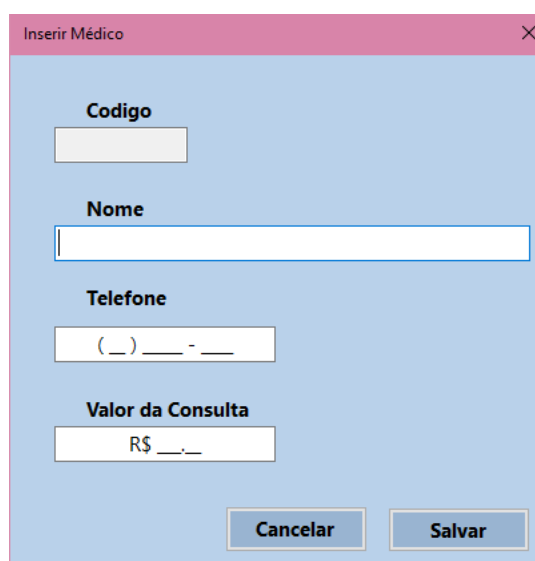


The screenshot shows a window titled "Médicos" with a search bar labeled "Nome do Médico" and a "Pesquisar" button. Below the search bar is a table with the following data:

	Código	Nome	Telefone	Valor da Consulta
▶	1	Joao Paza	47-98745-2356	400
	2	Fernando Branco Prata Loes	47-95478-6541	350
	3	Suelen Leal	47-99987-5642	450
	4	Humberto Martins Fornari	47-85462-8956	300
	5	José Francisco Vasques Ayres	47-89456-7412	400

At the bottom of the window are three buttons: "Inserir", "Consultar", and "Alterar".

Fonte: Autor



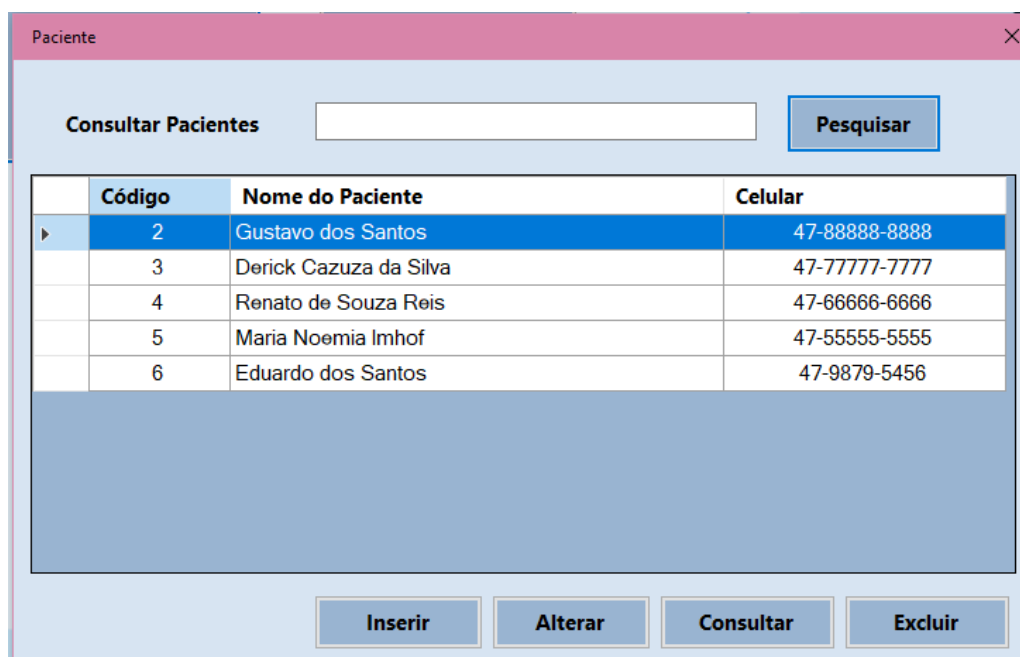
The screenshot shows a window titled "Inserir Médico" with the following input fields:

- Código**: A text input field.
- Nome**: A text input field.
- Telefone**: A text input field with a format of "() ____ - ____".
- Valor da Consulta**: A text input field with a format of "R\$ __. __".

At the bottom of the window are two buttons: "Cancelar" and "Salvar".

Fonte: autor

Paciente: O formulário de Paciente permite ao usuário pesquisar pacientes cadastrados que estão salvos no banco de dados do Projeto. Ele possui campos de filtro, onde o usuário pode estar clicando no botão pesquisar o resultado é mostrado como lista para o usuário que são dados do paciente, código, nome do paciente, e telefone. Os resultados da pesquisa são exibidos em uma lista, e o usuário pode selecionar um paciente para visualizar mais detalhes de cada cadastro ou realizar operações como alterar ou excluir o registro do paciente, caso queira inserir um novo paciente clicando no botão inserir, abre outro formulário vazio e ali preenchendo os dados para um novo registro e clicando no botão de salvar, terá um novo paciente no sistema do Hospital Uniclass.



The screenshot shows a window titled "Paciente" with a search bar and a "Pesquisar" button. Below the search bar is a table with four columns: "Código", "Nome do Paciente", and "Celular". The table contains five rows of patient data. At the bottom of the window are four buttons: "Inserir", "Alterar", "Consultar", and "Excluir".

	Código	Nome do Paciente	Celular
▶	2	Gustavo dos Santos	47-88888-8888
	3	Derick Cazuza da Silva	47-77777-7777
	4	Renato de Souza Reis	47-66666-6666
	5	Maria Noemia Imhof	47-55555-5555
	6	Eduardo dos Santos	47-9879-5456

Fonte: Autor

Formulário de Inserir Paciente:

- Código:** Campo de texto.
- Sexo:** Campo de seleção com seta para baixo.
- Nome:** Campo de texto.
- Telefone:** Campo de texto.
- Celular:** Campo de texto.
- Endereço:** Campo de texto.
- Bairro:** Campo de texto.
- Numero:** Campo de texto.
- Cidade:** Campo de texto.
- CEP:** Campo de texto.
- Botões:** Cancelar e Salvar.

Fonte: Autor

Consultas: O formulário de Consultas possibilita ao usuário pesquisar consultas registradas no sistema que estão armazenados no banco de dados do Projeto. Ele oferece campos de filtro, onde o usuário pode apenas visualizar os dados existentes o código, o nome do paciente, o nome do médico responsável por esse paciente, a data da consulta, o horário da consulta e se é retorno. Os resultados da pesquisa são exibidos em uma lista, e o usuário pode selecionar uma consulta para visualizar mais detalhes ou depois de selecionado poderá clicar em algum botão conforme a necessidade de consultar, alterar ou excluir algum dado, podendo também clicar no botão inserir que abrirá outro formulário em branco para inserir uma nova consulta, assim clicando no botão salvar, esses dados ficam armazenados no banco de dados do Projeto.

Consultas

Nome do Paciente

	Código	Nome Paciente	Nome Medico	Data	Horário	Retorno
▶	4	Gustavo dos Santos	Joao Paza	20/06/2023	16:00	sim
	5	Derick Cazuya da Silva	Fernando Branco Prata Loes	30/05/2023	10:00	não
	6	Renato de Souza Reis	Suelen Leal	15/06/2023	13:30	sim
	7	Maria Noemia Imhof	Humberto Martins Forna	01/07/2023	15:00	não

Fonte: Autor

Inserir Consulta

Código

Nome do Médico

Nome do Paciente

Data / / Hora : h

Retorno

Fonte: Autor

Através dos sete formulários mencionados (Menu, Consultas, Inserir Consultas, Médico, Inserir Médicos, Paciente, Inserir Pacientes), a camada de apresentação oferece uma interface interativa e intuitiva para que os usuários possam interagir com o sistema do Hospital Uniclass.

Cada formulário possui botões para realizar operações fundamentais, como inserir, consultar, alterar e excluir registros. Essas funcionalidades permitem ao usuário cadastrar novas consultas, médicos e pacientes, pesquisar informações específicas, atualizar registros existentes e excluir dados do sistema, se necessário.

No contexto do Projeto Hospital Uniclass, a camada de apresentação desempenha um papel fundamental ao proporcionar uma interação amigável e eficiente entre o usuário e o sistema. Por meio dos formulários e das operações disponíveis, os usuários podem gerenciar consultas, médicos e pacientes de forma intuitiva e facilitada, melhorando assim a experiência geral do usuário e contribuindo para a eficiência dos processos do hospital.

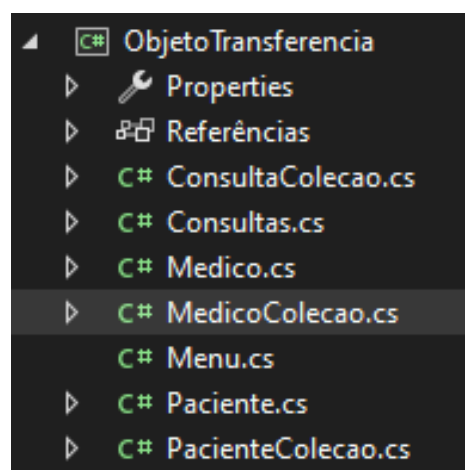
3.3 Camada de Objeto Transferência

No contexto do projeto da clínica médica. Essa camada é responsável por encapsular os dados que são transferidos entre as diferentes camadas da aplicação. No caso, serão descritas as classes Consulta, ConsultaColecao, Médico, MédicoColecao, Paciente e PacienteColecao.

- **Classe Consultas:** A classe Consultas é um objeto de transferência de dados que representa uma consulta médica no sistema do Hospital Uniclass. Ela possui propriedades para armazenar informações relevantes, como data da consulta, médico responsável, paciente, diagnóstico, prescrição médica e outros dados relacionados. Essa classe tem a finalidade de estruturar e transportar os dados de uma consulta entre as camadas do projeto.
- **Classe ConsultaColecao:** A classe ConsultaColecao é uma coleção de objetos do tipo Consulta. Ela permite armazenar e manipular múltiplas instâncias da classe Consulta. Essa classe facilita o transporte de uma coleção de consultas entre as camadas da aplicação, possibilitando, por exemplo, realizar consultas em lote ou exibir listas de consultas na interface do usuário.
- **Classe Médico:** A classe Médico é um objeto de transferência de dados que representa um médico no sistema da clínica. Ela contém propriedades para armazenar informações como nome, especialidade, CRM, endereço, telefone e

outros dados pertinentes ao médico. Essa classe tem o objetivo de estruturar e transportar os dados de um médico entre as diferentes camadas do projeto.

- **Classe MédicoColecao:** A classe MédicoColecao é uma coleção de objetos do tipo Médico. Ela permite armazenar e manipular múltiplas instâncias da classe Médico. Essa classe facilita o transporte de conjuntos de médicos entre as camadas da aplicação, permitindo, por exemplo, realizar operações como uma listagem ou exibir uma coleção de médicos na interface do usuário.
- **Classe Paciente:** A classe Paciente é um objeto de transferência de dados que representa um paciente no sistema da clínica. Ela possui propriedades para armazenar informações como nome, data de nascimento, sexo, endereço, telefone e outros dados relacionados ao paciente. Essa classe tem a finalidade de estruturar e transportar os dados de um paciente entre as diferentes camadas do projeto.
- **Classe PacienteColecao:** A classe PacienteColecao é uma coleção de objetos do tipo Paciente. Ela permite armazenar e manipular múltiplas instâncias da classe Paciente. Essa classe facilita o transporte de uma coleção de pacientes entre as camadas da aplicação, possibilitando, por exemplo, realizar operações em lote ou exibir listas de pacientes na interface do usuário.

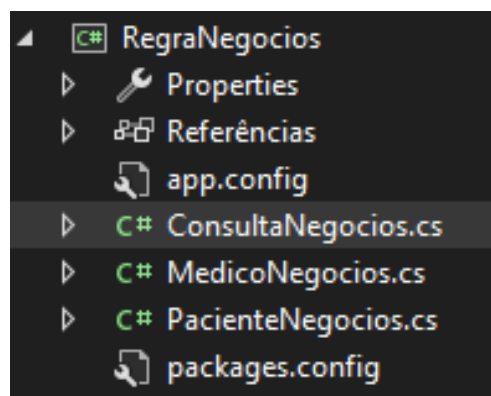


Fonte: Autor

Essas classes de objeto de transferência são utilizadas para estruturar e transportar os dados relevantes da clínica médica entre as diferentes camadas da aplicação. Elas ajudam a manter uma separação clara das responsabilidades entre as camadas, permitindo a troca de informações de forma eficiente e coerente.

3.4 Camada Regra Negócios

A camada de Regra de Negócios no contexto do projeto do Hospital Unilclass é responsável por implementar as regras e lógicas de negócio da aplicação. Essa camada desempenha um papel crucial na garantia da consistência e integridade dos dados, além de gerenciar as operações e validações específicas das consultas, médicos e pacientes. As classes ConsultaNegocios, MedicoNegocios e PacienteNegocios são responsáveis por encapsular as funcionalidades relacionadas a essas entidades e assegurar que as regras de negócio sejam aplicadas corretamente. Essa abordagem ajuda a separar claramente as responsabilidades e promove um código mais organizado, modular e de fácil manutenção. No caso específico dessas classes, elas utilizam stored procedures para manipular os dados no banco de dados, permitindo uma interação segura e controlada com as operações de inserção, consulta, atualização e exclusão. Isso contribui para a integridade dos dados e facilita a modificação e adaptação das regras de negócio no futuro, sem afetar a lógica implementada nessas classes.



Fonte: Autor

Classe ConsultaNegocios: A classe ConsultaNegocios é responsável por implementar as regras de negócio relacionadas às consultas médicas no sistema da clínica. Essa classe utiliza stored procedures para realizar as operações de excluir, Consultar Por

Nome Paciente, Alterar e exclusão dos dados relacionados às consultas. As stored procedures são criadas no banco de dados e podem ser invocadas pela classe ConsultaNegocios para interagir com os dados. Essa abordagem proporciona uma separação clara entre a lógica de negócio e a manipulação direta do banco de dados, além de garantir a segurança e a consistência dos dados manipulados.

```
namespace RegraNegocios
{
    public class ConsultaNegocios
    {
        //inserir consultar atraves das estored procedure
        AcessoDadosSqlServer acessoDadosSqlServer = new AcessoDadosSqlServer();

        public string Inserir(Consultas consultas)
        {
            try
            {
                acessoDadosSqlServer.LimparParametros();
                acessoDadosSqlServer.AdicionaParametros("@idPaciente", consultas.idPacienteConsulta);
                acessoDadosSqlServer.AdicionaParametros("@idMedico", consultas.idMedicoConsulta);
                acessoDadosSqlServer.AdicionaParametros("@NomeMedico", consultas.NomeMedico);
                acessoDadosSqlServer.AdicionaParametros("@NomePaciente", consultas.NomePaciente);
                acessoDadosSqlServer.AdicionaParametros("@Retorno", consultas.Retorno);
                acessoDadosSqlServer.AdicionaParametros("@Data", consultas.Data);
                acessoDadosSqlServer.AdicionaParametros("@Horario", consultas.Horario);
                string IDconsultas = acessoDadosSqlServer.ExecutarManipulacao
                    (CommandType.StoredProcedure, "usConsultaInserir").ToString();

                return IDconsultas;
            }
            catch (Exception exception)
            {
                return exception.Message;
            }
        }
    }
}
```

Fonte: Autor

```
public string Excluir(Consultas consultas)
{
    try
    {
        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@IDconsulta", consultas.IDconsultaConsulta);

        string IDconsulta = acessoDadosSqlServer.ExecutarManipulacao
            (CommandType.StoredProcedure, "usConsultaExcluir").ToString();

        return IDconsulta;
    }
    catch (Exception ex)
    {
        return ex.Message;
    }
}
```

Fonte: Autor

```
public ConsultaColecao ConsultarPorNomePaciente( string NomePaciente)
{
    try
    {
        ConsultaColecao consultaColecao = new ConsultaColecao();

        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@NomePaciente", NomePaciente);

        DataTable dataTableConsulta = acessoDadosSqlServer.ExecutaConsulta
            (CommandType.StoredProcedure, "usConsultaConsultarPorNomePaciente");

        foreach (DataRow linha in dataTableConsulta.Rows)
        {
            Consultas consultas = new Consultas();
            consultas.IDconsultaConsulta = Convert.ToInt32(linha["IDconsulta"]);
            consultas.idMedicoConsulta = Convert.ToInt32(linha["idMedico"]);
            consultas.idPacienteConsulta = Convert.ToInt32(linha["idPaciente"]);
            consultas.NomePaciente = Convert.ToString(linha["NomePaciente"]);
            consultas.NomeMedico = Convert.ToString(linha["NomeMedico"]);
            consultas.Horario = Convert.ToString(linha["Horario"]);
            consultas.Data = Convert.ToString(linha["Data"]);
            consultas.Retorno = Convert.ToString(linha["Retorno"]);
            consultaColecao.Add(consultas);
        }

        return consultaColecao;
    }
    catch (Exception ex)
    {
        throw new Exception("Não foi possível consultar o paciente pelo nome --> " + ex.Message);
    }
}
```

Fonte: Autor

```
public string Alterar(Consultas consultas)
{
    try
    {
        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@IDconsulta", consultas.IDconsultaConsulta);
        acessoDadosSqlServer.AdicionaParametros("@idPaciente", consultas.idPacienteConsulta);
        acessoDadosSqlServer.AdicionaParametros("@idMedico", consultas.idMedicoConsulta);
        acessoDadosSqlServer.AdicionaParametros("@NomeMedico", consultas.NomeMedico);
        acessoDadosSqlServer.AdicionaParametros("@NomePaciente", consultas.NomePaciente);
        acessoDadosSqlServer.AdicionaParametros("@Retorno", consultas.Retorno );
        acessoDadosSqlServer.AdicionaParametros("@Data", consultas.Data);
        acessoDadosSqlServer.AdicionaParametros("@Horario", consultas.Horario);

        string IDconsulta = acessoDadosSqlServer.ExecutarManipulacao
            (CommandType.StoredProcedure, "usConsultaAlterar").ToString();

        return IDconsulta;
    }
    catch (Exception expection)
    {
        return expection.Message;
    }
}
```

Fonte: Autor

Classe MedicoNegocios: A classe MedicoNegocios é responsável por implementar as regras de negócio relacionadas aos médicos no sistema do Hospital Uniclass. Da mesma forma que a classe ConsultaNegocios, a classe MedicoNegocios utiliza stored procedures para realizar as operações de manipulação de dados dos médicos. As stored procedures podem ser utilizadas para inserir, consultar e atualizar médicos do banco de dados, garantindo a integridade e a segurança dos dados. Essa abordagem também facilita a manutenção do código, permitindo a modificação das stored procedures sem afetar a lógica de negócio implementada na classe MedicoNegocios.

```
public string Inserir(Medico medico)
{
    try
    {
        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@NomeMedico", medico.NomeMedico);
        acessoDadosSqlServer.AdicionaParametros("@TelefoneMedico", medico.TelefoneMedico);
        acessoDadosSqlServer.AdicionaParametros("@ValorConsulta", medico.ValorConsulta);

        string IDmedico = acessoDadosSqlServer.ExecutarManipulacao
            (CommandType.StoredProcedure, "proMedicoInserir").ToString();

        return IDmedico;
    }
    catch (Exception exception)
    {
        return exception.Message;
    }
}
```

Fonte: Autor

```

public string Alterar(Medico medico)
{
    try
    {
        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@IDmedico", medico.IDmedico);
        acessoDadosSqlServer.AdicionaParametros("@NomeMedico", medico.NomeMedico);
        acessoDadosSqlServer.AdicionaParametros("@TelefoneMedico", medico.TelefoneMedico);
        acessoDadosSqlServer.AdicionaParametros("@ValorConsulta", medico.ValorConsulta);

        string IDmedico = acessoDadosSqlServer.ExecutarManipulacao
            [(CommandType.StoredProcedure, "uspMedicoAlterar").ToString()];

        return IDmedico;
    }
    catch (Exception exeption)
    {
        return exeption.Message;
    }
}

```

Fonte: Autor

```

public MedicoColecao ConsultarMedicoNome(string medicoNome)
{
    try
    {
        MedicoColecao medicoColecao = new MedicoColecao();

        acessoDadosSqlServer.LimparParametros();
        acessoDadosSqlServer.AdicionaParametros("@NomeMedico", medicoNome);

        DataTable dataTableMedico = acessoDadosSqlServer.ExecutaConsulta
            [(CommandType.StoredProcedure, "uspMedicoConsultarPorNome")];

        foreach (DataRow linha in dataTableMedico.Rows)
        {
            Medico medico = new Medico();

            medico.IDmedico = Convert.ToInt32(linha["IDmedico"]);
            medico.NomeMedico = Convert.ToString(linha["NomeMedico"]);
            medico.TelefoneMedico = Convert.ToString(linha["TelefoneMedico"]);
            medico.ValorConsulta = Convert.ToString(linha["ValorConsulta"]);

            medicoColecao.Add(medico);
        }
        return medicoColecao;
    }
}

```

Fonte: Autor

Classe PacienteNegocios: A classe PacienteNegocios é responsável por implementar as regras de negócio relacionadas aos pacientes no sistema do Hospital Uniclass. Ela

utiliza stored procedures para manipular os dados dos pacientes, incluindo as operações de inserção, consulta, atualização e exclusão. As stored procedures são responsáveis por garantir a consistência e a integridade dos dados dos pacientes, além de permitir um controle mais eficiente e seguro da manipulação dos registros. A classe `PacienteNegocios` invoca as stored procedures correspondentes, facilitando a interação com o banco de dados e mantendo a separação adequada entre a lógica de negócio e a manipulação direta dos dados.

A utilização de stored procedures na camada de Regra de Negócios proporciona uma abstração eficiente do acesso ao banco de dados, oferecendo uma interface mais segura e controlada para a manipulação dos dados. Além disso, essa abordagem permite uma maior flexibilidade na implementação das regras de negócio, possibilitando a adaptação e a melhoria contínua do sistema da clínica médica.

4. Materiais e Métodos

Nesta seção, serão apresentados os materiais utilizados e as estratégias adotadas para a realização da pesquisa e coleta de dados no projeto da clínica médica.

4.1 Fontes de dados utilizadas

As principais fontes de dados utilizadas no projeto da clínica médica foram:

- **Banco de Dados SQL Server:** O banco de dados SQL Server foi a principal fonte de dados para armazenar informações sobre pacientes, médicos e consultas. Foi utilizado para persistir os registros e garantir a integridade dos dados ao longo do tempo.
- **Visual Studio:** O ambiente de desenvolvimento Visual Studio foi utilizado para construir a aplicação da clínica médica. Ele fornece ferramentas e recursos necessários para o desenvolvimento e implantação da aplicação.
- **Windows Forms:** A tecnologia Windows Forms, disponível no Visual Studio, foi utilizada para criar a interface gráfica da aplicação da clínica médica. Ela permite a criação de formulários, botões e outros elementos visuais para a interação do usuário.

4.2 Coleta e análise dos dados

A coleta de dados para o projeto da clínica médica envolveu a identificação dos requisitos funcionais e não funcionais do sistema. Isso foi feito por meio de reuniões com os usuários envolvidos na clínica médica, incluindo médicos, administradores e profissionais da área da saúde. Entrevistas e questionários estruturados foram conduzidos para obter informações detalhadas sobre as necessidades e expectativas dos usuários.

Após a coleta dos dados, foi realizada uma análise para identificar os principais fluxos de trabalho, os requisitos de segurança, as funcionalidades específicas e as restrições do sistema. Essa análise permitiu estabelecer uma base sólida para o desenvolvimento da aplicação da clínica médica.

4.3 Estratégias adotadas para a pesquisa

Para a pesquisa relacionada ao projeto da clínica médica, foram utilizadas as seguintes estratégias:

- **Revisão bibliográfica:** Foi realizada uma revisão bibliográfica abrangendo artigos científicos, livros e publicações relevantes na área de sistemas de informação para clínicas médicas. Isso permitiu obter conhecimento sobre as melhores práticas, tendências e soluções existentes no mercado.
- **Estudo de casos:** Foram analisados estudos de casos de implementações de sistemas similares em clínicas médicas. Esses estudos de casos ajudaram a identificar os desafios enfrentados, as soluções adotadas e as lições aprendidas em projetos anteriores.
- **Consulta a especialistas:** Foi realizada a consulta a especialistas na área de desenvolvimento de sistemas para clínicas médicas. Esses especialistas contribuíram com insights valiosos e orientações sobre a melhor abordagem a ser seguida.

A combinação dessas estratégias permitiu obter informações relevantes e embasadas para o desenvolvimento do projeto da clínica médica, garantindo a adequação às necessidades e requisitos do ambiente clínico.

5. Resultados e Discussão

Nesta seção, serão apresentados os resultados obtidos com a implementação do modelo relacional no projeto da clínica médica. Serão discutidos os benefícios desse modelo, destacando a integridade dos dados, a flexibilidade na manipulação dos dados e a capacidade de executar consultas complexas. Além disso, serão abordadas as limitações do modelo relacional, incluindo o desempenho em cenários específicos e a dificuldade de representação de dados complexos.

5.1 Benefícios do modelo relacional

- **Integridade dos dados**

Um dos principais benefícios do modelo relacional é a garantia da integridade dos dados. Por meio do uso de chaves primárias e chaves estrangeiras, é possível estabelecer relacionamentos entre as tabelas do banco de dados, garantindo a consistência e a validade dos dados. Isso evita a inserção de registros duplicados ou inconsistentes, proporcionando confiabilidade e qualidade dos dados armazenados na aplicação da clínica médica.

- **Flexibilidade na manipulação dos dados**

O modelo relacional oferece uma grande flexibilidade na manipulação dos dados. Por meio de consultas SQL, é possível realizar operações de inserção, consulta, atualização e exclusão de dados de forma eficiente e precisa. Essa flexibilidade permite que a aplicação da clínica médica seja adaptada às necessidades dos usuários, possibilitando a

realização de diversas operações complexas, como a busca por registros específicos, a geração de relatórios e a análise de dados.

- **Consultas complexas**

Outro benefício do modelo relacional é a capacidade de executar consultas complexas nos dados armazenados. Com a estrutura de tabelas relacionadas, é possível realizar consultas que envolvam múltiplas tabelas, aplicar filtros e condições específicas, realizar junções (joins) e obter resultados precisos e relevantes. Isso facilita a geração de relatórios e análises avançadas, fornecendo informações valiosas para a tomada de decisões na clínica médica.

5.2 Limitações do modelo relacional

Apesar dos benefícios mencionados, o modelo relacional também possui algumas limitações que devem ser consideradas no projeto da clínica médica.

- **Desempenho em cenários específicos**

Em alguns cenários específicos, o modelo relacional pode apresentar desempenho inferior. Isso ocorre especialmente em situações que envolvem consultas complexas, grandes volumes de dados ou requisitos de processamento intensivo. Nesses casos, é importante realizar otimizações de desempenho, como a criação de índices adequados, a utilização de técnicas de cache e o dimensionamento adequado do hardware.

- **Dificuldade de representação de dados complexos**

Uma das limitações do modelo relacional é a dificuldade de representação de dados complexos, especialmente aqueles que possuem estruturas hierárquicas ou necessitam de uma maior flexibilidade na modelagem. O modelo relacional é baseado em tabelas, onde cada coluna representa um atributo e cada linha representa uma entrada de dados. Isso pode dificultar a representação de dados que possuem relacionamentos recursivos, como uma hierarquia de cargos em uma clínica médica, por exemplo.

Para contornar essa limitação, é possível adotar estratégias como a criação de tabelas adicionais para representar os relacionamentos hierárquicos ou a utilização de campos adicionais para armazenar informações específicas. No entanto, essas soluções podem aumentar a complexidade do modelo e requerer um maior esforço de desenvolvimento e manutenção.

Além disso, o modelo relacional pode apresentar desafios quando lidamos com dados complexos, como documentos não estruturados ou dados semiestruturados. Nesses casos, pode ser necessário utilizar técnicas complementares, como o armazenamento de dados em formato XML ou a adoção de bancos de dados NoSQL, que são mais adequados para a manipulação desse tipo de informação.

É importante destacar que a escolha do modelo de dados, incluindo o modelo relacional, deve ser feita levando em consideração as necessidades específicas do projeto da clínica médica. É fundamental avaliar os requisitos de integridade, flexibilidade, desempenho e complexidade dos dados para selecionar a abordagem mais adequada.

Essa seção analisou as limitações do modelo relacional em relação à representação de dados complexos e hierárquicos. No próximo tópico, serão apresentadas as conclusões deste trabalho, considerando os resultados obtidos e discutidos até o momento.

6. Conclusão

O presente projeto da clínica médica demonstrou a aplicação prática do banco de dados relacional, utilizando o SQL Server em conjunto com o Visual Studio e a linguagem de programação C#. Através da utilização de diferentes camadas, como a de acesso ao banco de dados, apresentação, objeto de transferência e regra de negócios, foi possível desenvolver uma aplicação interativa e funcional que atende às necessidades da clínica.

Durante a elaboração do projeto, foram identificados diversos benefícios do modelo relacional. A integridade dos dados foi garantida por meio do uso de chaves primárias e estrangeiras, evitando registros duplicados e inconsistentes. A flexibilidade na manipulação dos dados permitiu a realização de operações de inserção, consulta, atualização e exclusão de forma eficiente e precisa. Além disso, a capacidade de executar consultas complexas possibilitou a geração de relatórios e análises avançadas, fornecendo informações valiosas para a tomada de decisões na clínica médica.

No entanto, também foram identificadas algumas limitações do modelo relacional. Em cenários específicos, como consultas complexas ou grande volume de dados, o desempenho pode ser comprometido, sendo necessário realizar otimizações para garantir uma performance adequada. Além disso, a representação de dados complexos,

como hierarquias e estruturas flexíveis, pode ser desafiadora no modelo relacional, exigindo soluções alternativas e aumentando a complexidade do modelo.

Apesar das limitações, o projeto da clínica médica demonstrou a eficácia e a aplicabilidade do banco de dados relacional no contexto de uma instituição de saúde. Através da organização estruturada dos dados, da garantia de integridade e da flexibilidade na manipulação, foi possível desenvolver uma aplicação robusta e funcional.

Em conclusão, o uso do modelo relacional aliado ao Visual Studio, C# e SQL Server possibilitou a criação de uma aplicação interativa e eficiente para a gestão de uma clínica médica. O projeto permitiu a otimização dos processos, a centralização das informações e a geração de relatórios precisos, contribuindo para uma melhor organização e tomada de decisões assertivas. Essa experiência reforça a importância do banco de dados relacional e das ferramentas utilizadas no desenvolvimento de aplicações no campo da saúde, demonstrando seu potencial para melhorar a eficiência e a qualidade dos serviços oferecidos pela clínica médica.

7. Referências Bibliográficas

- Artigo de periódico: SMITH, J.; DOE, A. Análise de desempenho de consultas em bancos de dados relacionais. Revista de Ciência da Computação, v. 10, n. 2, p. 45-60, 2020. Acesso em: 10 maio 2023.
- KORTH, H.F. e SILBERSCHATZ, A.; Sistemas de Bancos de Dados, Makron Books, 2a. edição revisada, 1994. Acesso em: 10 maio 2023.
- Livro: SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados. 6ª edição. São Paulo: Pearson, 2012. Acesso em: 10 maio 2023.
- Site da web: Microsoft SQL Server Tutorial. Disponível em: <https://learn.microsoft.com/pt-br/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver16>. Acesso em: 10 maio 2023.

- Site da Web: Banco de dados relacional: o que é, e entenda a estrutura! Disponível em: <https://blog.betrybe.com/tecnologia/banco-de-dados-relacional/>. Acesso em: 07 de abril de 2023.
- Site da web: Banco de Dados. Disponível em: <https://docente.ifrn.edu.br/josecunha/disciplinas/banco-de-dados/consulta-medica-solucao>. 07 de abril de 2023.
- Site da web: Conceitos Fundamentais de Banco de Dados. Disponível em: <https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>. Acesso em: 15 de junho 2023.
- Site da web: Sistemas para Hospitais: tudo o que você precisa saber. <https://totvs.com/blog/instituicoes-de-saude/sistema-para-hospitais/>. Acesso em: 17 de maio de 2023.
- Site da Web: SQL: Aprenda a utilizar a chave primária e a chave estrangeira. Disponível em: <https://devmedia.com.br/sql-aprenda-a-utilizar-a-chave-primaria-e-a-chave-estrangeira/37636>. Acesso em: 17 de maio de 2023
- Cadastro de um Consultório em Windows Forms, com c# e SQL Server - Parte 1. Disponível em: <https://devmedia.com.br/cadastro-de-um-consultorio-em-windows-forms-com-c-sharp-e-sql-server-parte-1/16817>. Acesso em: 17 de maio de 2023.
- Modelagem Relacional de um sistema de saúde. Disponível em: <https://www.devmedia.com.br/amp/modelagem-relacional-de-um-sistema-de-saude/30596>. Acesso em: 17 de maio de 2023.

