PROJETO DE BANCO DE DADOS - CLÍNICA MÉDICA

SALVADOR - BA
2024

Deivide Maciel Sales Costa - 202308426028
Fernando Santana de Morais - 202304504164
Igor Natan dos Santos Ferreira - 202309158949
Iury Conceição França - 202302230253

Projeto de Banco de Dados para sistema computacional apresentado como requisito parcial de avaliação da disciplina de Banco de Dados ARA0040 na instiituição CENTRO UNIVERSITÁRIO FAVIP WYDEN.

Professor: Heleno Cardoso da Silva Filho

SUMÁRIO

INTRODUÇÃO

A área da saúde está em constante evolução, buscando sempre aprimorar a experiência dos pacientes e otimizar o trabalho dos profissionais. Nesse contexto, o desenvolvimento de uma aplicação para agendamento de consultas médicas se torna uma ferramenta crucial para a modernização e agilidade no atendimento. Desenhamos, portanto, todo o processo de desenvolvimento de um banco de dados, partindo do princípio no Levantamento de Requisitos, Modelagem Conceitual, Modelagem Lógica e por fim a Modelagem Física, onde é estabelecido o DDL (“*Data Definition Language”* Linguagem de Definição de Dados).

LEVANTAMENTO DE REQUISITOS

Objetivos da Aplicação

1. **Facilitar o agendamento de consultas**

Permitindo que os pacientes marquem consultas online, 24 horas por dia, 7 dias por semana, sem precisar entrar em contato com a recepção da clínica.

2. **Otimizar o tempo dos médicos**

Reduzindo o tempo gasto com tarefas administrativas, como agendamento de consultas, permitindo que se concentrem no atendimento aos pacientes.

3. **Diminuir as filas de espera**

Agilizando o processo de agendamento e reduzindo o tempo de espera dos pacientes na recepção da clínica.

4. **Melhorar a experiência do paciente**

Oferecendo um atendimento mais prático, rápido e personalizado.

Funcionalidades da Aplicação

1. **Agendamento de Consultas**

Seleção de médico, especialidade, data e horário da consulta. Visualização da disponibilidade dos médicos em tempo real. Confirmação da consulta por e-mail e SMS.

2. **Gerenciamento de Consultas**

Visualização do histórico de consultas do paciente. Cancelamento ou reagendamento de consultas. Envio de lembretes de consultas aos pacientes.

3. **Cadastro e Autenticação de Usuários**

Criação de perfil para pacientes e médicos. Inserção de informações pessoais e profissionais. Login com e-mail e senha. Recuperação de senha esquecida.

Gerenciamento de Agenda

1. **Visualização da Agenda**

Os médicos podem visualizar sua agenda de consultas agendadas.

2. **Registro de Consultas**

Os médicos podem registrar novas consultas em seus horários disponíveis.

3. **Confirmação de Consultas**

O sistema envia confirmações de consultas agendadas aos pacientes.

Histórico de Consultas

Pacientes: Podem visualizar o histórico de suas consultas, com informações sobre data, hora, médico e especialidade.

Médicos: Podem visualizar o histórico de consultas de seus pacientes, com informações relevantes para o atendimento.

Filtros: O histórico de consultas pode ser filtrado por data, médico e especialidade.

Perfil do Usuário

Editar Informações: Pacientes e médicos podem editar suas informações pessoais e profissionais.

Alterar Senha: Usuários podem alterar sua senha de acesso à aplicação.

Segurança: O sistema valida as informações inseridas para garantir a segurança dos dados.

Notificações

Agendamento de Consulta: Pacientes e médicos recebem notificações por e-mail e SMS sobre consultas agendadas.

Cancelamento e Reagendamento: Pacientes e médicos recebem notificações sobre consultas canceladas ou reagendadas.

Lembretes de Consulta: Pacientes recebem lembretes por e-mail e SMS no dia da consulta.

Integração com Prontuário Eletrônico

Acesso a Informações do Paciente: O médico pode acessar informações relevantes do prontuário eletrônico do paciente durante o atendimento.

Segurança e Privacidade: A integração com o prontuário eletrônico deve ser opcional e seguir rigorosos padrões de segurança e privacidade.

Melhoria no Atendimento: O acesso a informações do prontuário eletrônico pode melhorar a qualidade do atendimento médico.

MODELAGEM CONCEITUAL

A ideia de Modelagem Conceitual vem para definir uma a primeira visualização de como os dados vão se organizar em um banco de dados, sendo uma forma que guia os passos da modelagem do banco.

Decidimos montar o Modelo Conceitual com base nas informações do levantamento de requisitos do sistema. Identificamos **8 Entidades**, **5 Relacionamentos**, **2 Entidades Associativas**. Além disso, contamos com **32 Atributos** (**9 chaves primárias, 7 chaves estrangeiras, 4 chaves únicas e 12 atributos comuns**).

A entidade de **ENDERECO** foi separada da entidade **PACIENTE**, com a finalidade de permitir que um endereço possa pertencer a mais de um registro da entidade **PACIENTE**.

A entidade de **TELEFONE** também foi separada, para permitir que o paciente possa dispor de telefones distintos durante seu período cadastrado no banco de dados.

A relação entre o Médico que atende e o Paciente atendido foi estabelecida na entidade **CONSULTA**, que por fim, também possui relacionamento com a entidade de **HORARIO**. O objetivo neste caso é permitir o agendamento de consultas e verificação da disponibilidade do horário de médicos pelos clientes.

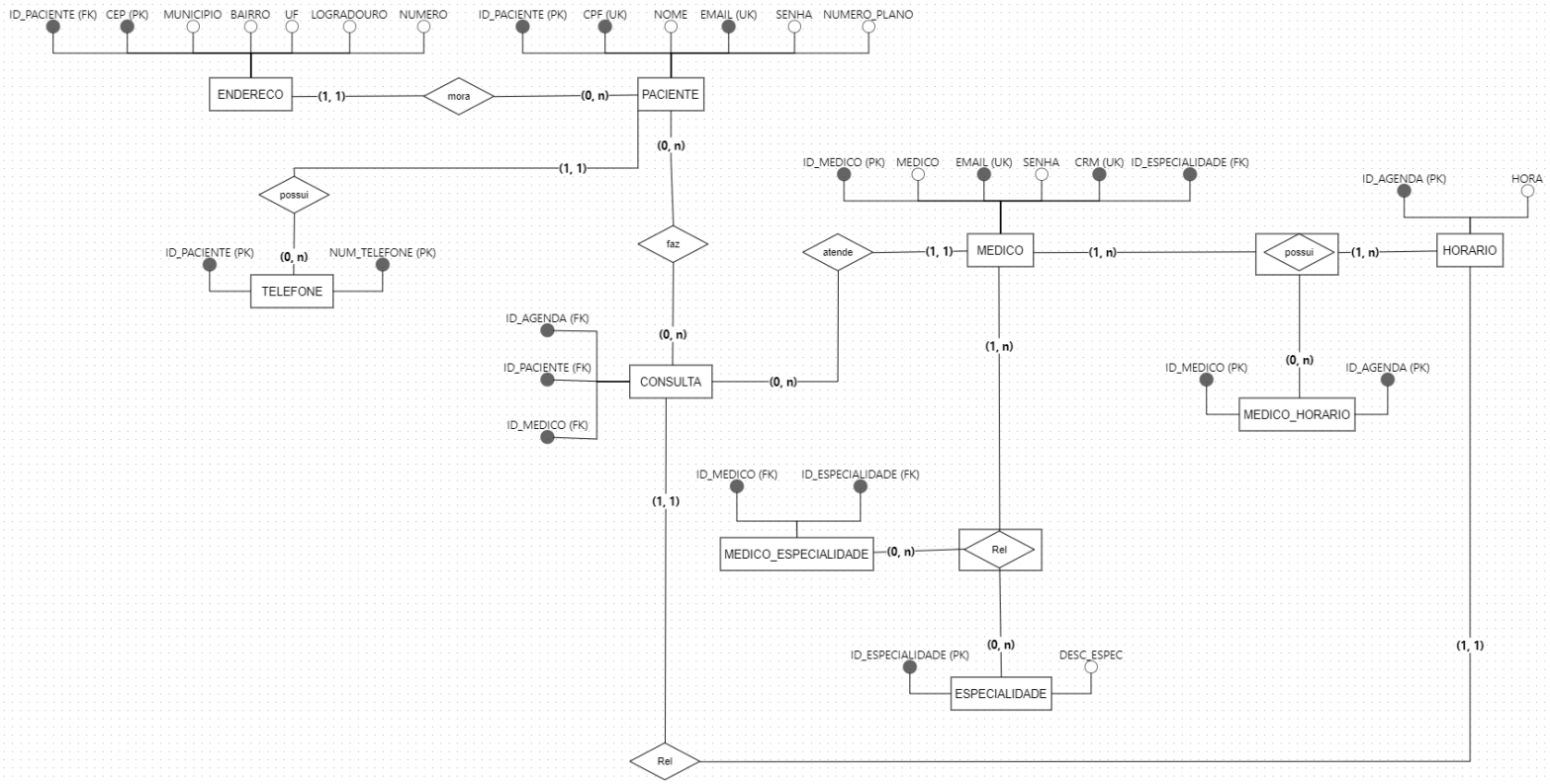
Disponibilizamos a entidade **HORARIO** vinculada a **MEDICO**, para poder apontar quais horários o médico pode ter disponível no sistema. Criamos uma Entidade Associativa **MEDICO_HORARIO** para relacionar os horários disponíveis para os médicos, visto que mais de um médico pode atender ao mesmo tempo.

A entidade **MEDICO** também se relaciona com a entidade de **ESPECIALIDADE**, seguindo o mesmo ponto que aplicamos no horário, em que mais de um médico pode possuir a mesma especialidade, assim criamos a **MEDICO_ESPECIALIDADE**.

Utilizamos estas Entidades Associativas para reduzir redundâncias na modelagem e aplicar corretamente a normalização.

Por fim, acredito que seja ideal destacar a existência de alguns atributos como chaves únicas, (**UK**) como **EMAIL**, **CRM**, e **CPF**. Visto que não se trata de uma chave primária que define a identificação do registro, mas é um tipo de chave que vai receber dados exclusivos de um registro e que não podem ser replicados aos demais.

Abaixo ilustramos a visualização da Modelagem Conceitual:

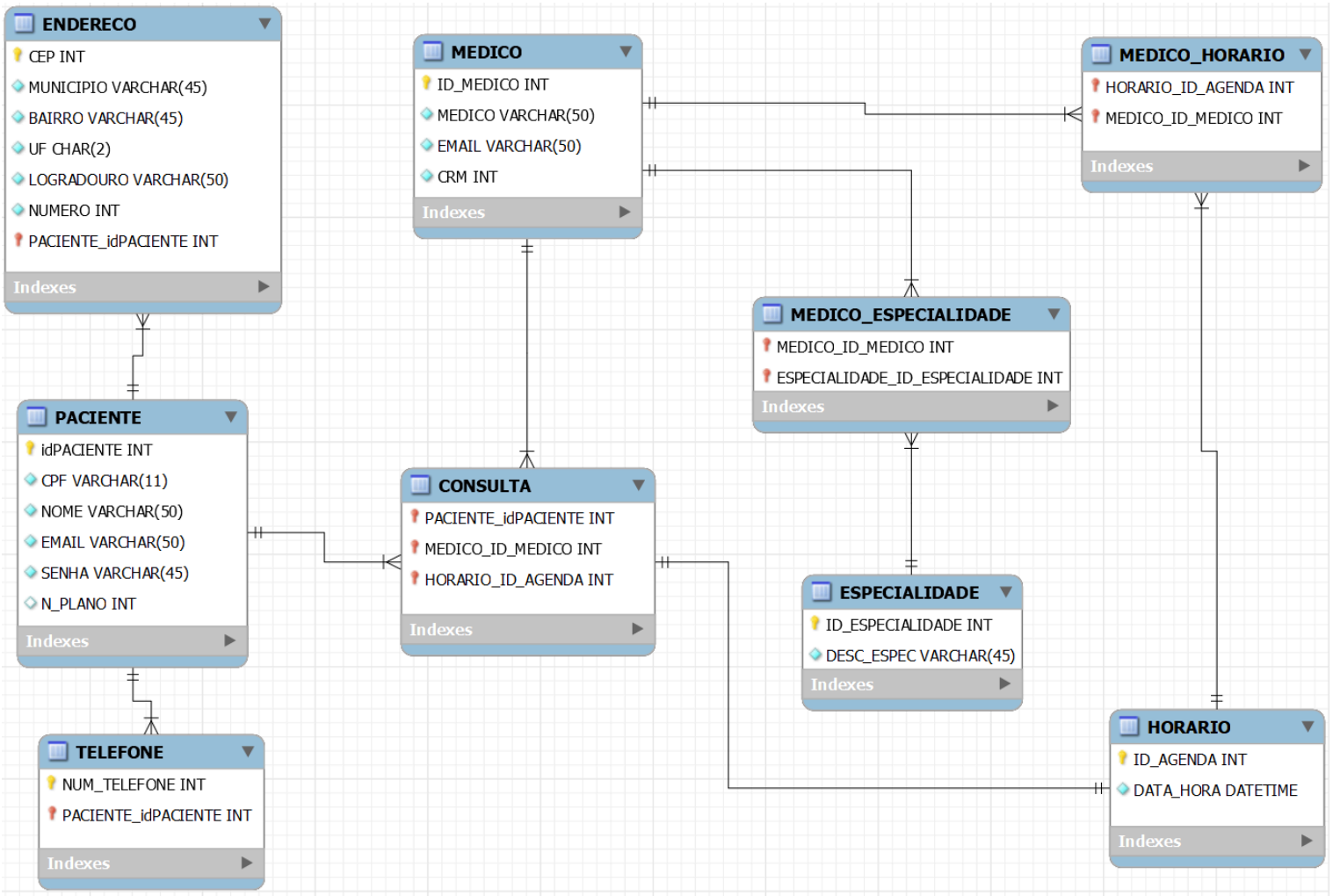


MODELAGEM LÓGICA

O modelo lógico é uma representação de como os dados serão organizados e estruturados dentro do banco de dados. Ele define as tabelas, as relações e cardinalidade, os atributos das entidades e tipos de dados desses atributos.

Tal modelo é crucial por servir como ponte entre o modelo conceitual e o modelo físico do banco de dados e seu principal objetivo é detalhar a estrutura dos dados para garantir a integridade, a consistência e a precisão dos dados

Abaixo ilustramos a visualização da Modelagem Lógica:



MODELAGEM FÍSICA

Após definido os parâmetros da modelagem lógica, partimos para criação do DDL. acessamos a ferramenta, que no caso foi o Microsoft Workbench e iniciamos a criação dos scripts.

Esse script é o de criação inicial da tabela de **PACIENTE**, onde serão armazenados os registros definidos anteriormente. aqui também já são definidas as chaves primárias e únicas.

```
CREATE TABLE IF NOT EXISTS `Clinica`.`PACIENTE` (
  `idPACIENTE` INT NOT NULL AUTO_INCREMENT,
  `CPF` VARCHAR(11) NOT NULL,
  `NOME` VARCHAR(50) NOT NULL,
  `EMAIL` VARCHAR(50) NOT NULL,
  `SENHA` VARCHAR(45) NOT NULL,
  `N_PLANO` INT NULL,
  PRIMARY KEY (`idPACIENTE`),
  UNIQUE INDEX `CPF_UNIQUE` (`CPF` ASC) VISIBLE,
  UNIQUE INDEX `EMAIL_UNIQUE` (`EMAIL` ASC) VISIBLE,
  UNIQUE INDEX `ID_PLANO_UNIQUE` (`N_PLANO` ASC) VISIBLE)
ENGINE = InnoDB
```

Tabela chamada **ENDERECO** do esquema de nossa Clínica. A tabela armazena informações de endereço associadas aos pacientes. Um pouco das principais características do modelo das colunas são:

CEP: Código postal, tipo inteiro, não nulo.
MUNICIPIO: Nome do município, tipo varchar com até 45 caracteres, não nulo.
BAIRRO: Nome do bairro, tipo varchar com até 45 caracteres, não nulo.
UF: Unidade Federativa (estado), tipo char com 2 caracteres, não nulo.
LOGRADOURO: Nome da rua/logradouro, tipo varchar com até 50 caracteres, não nulo.
NUMERO: Número do endereço, tipo inteiro, não nulo.
PACIENTE_idPACIENTE: Identificador do paciente, tipo inteiro, não nulo.

Este modelo físico é crucial para garantir a integridade referencial e a organização eficiente dos dados de endereço associados aos pacientes em uma aplicação de gerenciamento de clínicas.

Este script SQL cria uma tabela chamada **TELEFONE** dentro do banco de dados Clínica.
A tabela contém duas colunas: **NUM_TELEFONE** e **PACIENTE_idPACIENTE**. A chave primária é composta por ambas as colunas, garantindo que cada número de telefone associado a um paciente seja único. A coluna **PACIENTE_idPACIENTE** é uma chave estrangeira que referencia a coluna **idPACIENTE** da tabela **PACIENTE**, assegurando a integridade referencial.

A Tabela **MEDICO** armazena informações sobre médicos, garantindo que cada médico tenha um identificador único (**ID_MEDICO**), email (**EMAIL**) e número de CRM (**CRM**). A estrutura da tabela impõe essas restrições usando índices primários e únicos.

Tabela **ESPECIALIDADE**

Finalidade: Armazenar informações sobre as especialidades médicas disponíveis na clínica. Colunas:
ID_ESPECIALIDADE: Identificador único da especialidade (chave primária, inteiro não nulo).
DESC_ESPEC: Descrição da especialidade (caractere variável de até 45 caracteres, não nulo).

Tabela **MEDICO_ESPECIALIDADE**

Finalidade: Relacionar médicos com suas especialidades, representando quais especialidades cada médico possui. Colunas:
MEDICO_ID_MEDICO: Identificador do médico (inteiro não nulo), chave estrangeira que referencia **ID_MEDICO** na tabela **MEDICO**.
ESPECIALIDADE_ID_ESPECIALIDADE: Identificador da especialidade (inteiro não nulo), chave estrangeira que referencia **ID_ESPECIALIDADE** na tabela **ESPECIALIDADE**.

Índices e Restrições:
Índices sobre as colunas **MEDICO_ID_MEDICO** e **ESPECIALIDADE_ID_ESPECIALIDADE** para otimizar as consultas.
Chaves estrangeiras que garantem a integridade referencial com as tabelas **MEDICO e ESPECIALIDADE**.

Tabela **HORARIO**

Finalidade: Armazenar os horários de agendamento dos médicos na clínica.
Colunas:
ID_AGENDA: Identificador único do agendamento (chave primária, inteiro não nulo, autoincremento).
DATA_HORA: Data e hora do agendamento (tipo *DATETIME*, não nulo).

Tabela **MEDICO_HORARIO**

Finalidade: Relacionar médicos com seus horários de atendimento, indicando em quais horários cada médico está disponível. Colunas:

HORARIO_ID_AGENDA: Identificador do horário de agendamento (inteiro não nulo). Chave estrangeira que referencia a coluna **ID_AGENDA** na tabela **HORARIO**.
MEDICO_ID_MEDICO: Identificador do médico (inteiro não nulo). Chave estrangeira que referencia a coluna **ID_MEDICO** na tabela **MEDICO**.

Índices e Restrições:
Chave Primária: Composta pelas colunas **HORARIO_ID_AGENDA** e **MEDICO_ID_MEDICO**.

Índices:
fk_HORARIO_has_MEDICO_MEDICO1_idx: Índice sobre a coluna **MEDICO_ID_MEDICO**.
fk_HORARIO_has_MEDICO_HORARIO1_idx: Índice sobre a coluna HORARIO_ID_AGENDA.

Chaves Estrangeiras:
fk_HORARIO_has_MEDICO_HORARIO1: Referencia **ID_AGENDA** na tabela HORARIO, garantindo a integridade referencial.
fk_HORARIO_has_MEDICO_MEDICO1: Referencia **ID_MEDICO** na tabela MEDICO, garantindo a integridade referencial.
Mecanismo de Armazenamento: InnoDB

Tabela **CONSULTA**

Finalidade: Registrar as consultas médicas, incluindo o paciente, médico e o horário da consulta. Colunas:

PACIENTE_idPACIENTE: Identificador do paciente (inteiro não nulo). Chave estrangeira que referencia a coluna **idPACIENTE** na tabela **PACIENTE**.
MEDICO_ID_MEDICO: Identificador do médico (inteiro não nulo). Chave estrangeira que referencia a coluna **ID_MEDICO** na tabela **MEDICO**.
HORARIO_ID_AGENDA: Identificador do horário de agendamento (inteiro não nulo). Chave estrangeira que referencia a coluna **ID_AGENDA** na tabela **HORARIO**.

Índices e Restrições:

Chave Primária: Composta pelas colunas **PACIENTE_idPACIENTE**, **MEDICO_ID_MEDICO** e **HORARIO_ID_AGENDA**.

Índices:
fk_PACIENTE_has_MEDICO_MEDICO1_idx: Índice sobre a coluna **MEDICO_ID_MEDICO**.
fk_PACIENTE_has_MEDICO_PACIENTE1_idx: Índice sobre a coluna **PACIENTE_idPACIENTE**.
fk_CONSULTA_HORARIO1_idx: Índice sobre a coluna **HORARIO_ID_AGENDA**.

Chaves Estrangeiras:

fk_PACIENTE_has_MEDICO_PACIENTE1: Referencia **idPACIENTE** na tabela **PACIENTE**, garantindo a integridade referencial.
fk_PACIENTE_has_MEDICO_MEDICO1: Referencia **ID_MEDICO** na tabela **MEDICO**, garantindo a integridade referencial.
fk_CONSULTA_HORARIO1: Referencia **ID_AGENDA** na tabela **HORARIO**, garantindo a integridade referencial.
Mecanismo de Armazenamento: InnoDB

DDL - LINGUAGEM DE DEFINIÇÃO DE DADOS

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_
DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBS
TITUTION';

-- -----------------------------------------------------
-- Schema Clinica
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `Clinica` DEFAULT CHARACTER SET utf8 ;
USE `Clinica` ;

-- Table `Clinica`.`PACIENTE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`PACIENTE` (
  `idPACIENTE` INT NOT NULL AUTO_INCREMENT,
  `CPF` VARCHAR(11) NOT NULL,
  `NOME` VARCHAR(50) NOT NULL,
  `EMAIL` VARCHAR(50) NOT NULL,
  `SENHA` VARCHAR(45) NOT NULL,
  `N_PLANO` INT NULL,
  PRIMARY KEY (`idPACIENTE`),
  UNIQUE INDEX `CPF_UNIQUE` (`CPF` ASC) VISIBLE,
  UNIQUE INDEX `EMAIL_UNIQUE` (`EMAIL` ASC) VISIBLE,
  UNIQUE INDEX `ID_PLANO_UNIQUE` (`N_PLANO` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-- Table `Clinica`.`TELEFONE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`TELEFONE` (
  `NUM_TELEFONE` INT NOT NULL,
  `PACIENTE_idPACIENTE` INT NOT NULL,
  UNIQUE INDEX `NUM_TELEFONE_UNIQUE` (`NUM_TELEFONE` ASC)
VISIBLE,
  PRIMARY KEY (`PACIENTE_idPACIENTE`, `NUM_TELEFONE`),
  CONSTRAINT `fk_TELEFONE_PACIENTE`
    FOREIGN KEY (`PACIENTE_idPACIENTE`)
    REFERENCES `Clinica`.`PACIENTE` (`idPACIENTE`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Clinica`.`ENDERECO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`ENDERECO` (
  `CEP` INT NOT NULL,
  `MUNICIPIO` VARCHAR(45) NOT NULL,
  `BAIRRO` VARCHAR(45) NOT NULL,
  `UF` CHAR(2) NOT NULL,
  `LOGRADOURO` VARCHAR(50) NOT NULL,
  `NUMERO` INT NOT NULL,
  `PACIENTE_idPACIENTE` INT NOT NULL,
  PRIMARY KEY (`CEP`, `PACIENTE_idPACIENTE`),
  INDEX `fk_ENDERECO_PACIENTE1_idx` (`PACIENTE_idPACIENTE` ASC) VISIBLE,
  CONSTRAINT `fk_ENDERECO_PACIENTE1`
    FOREIGN KEY (`PACIENTE_idPACIENTE`)
    REFERENCES `Clinica`.`PACIENTE` (`idPACIENTE`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `Clinica`.`MEDICO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`MEDICO` (
  `ID_MEDICO` INT NOT NULL AUTO_INCREMENT,
  `MEDICO` VARCHAR(50) NOT NULL,
  `EMAIL` VARCHAR(50) NOT NULL,
  `CRM` INT NOT NULL,
  PRIMARY KEY (`ID_MEDICO`),
  UNIQUE INDEX `EMAIL_UNIQUE` (`EMAIL` ASC) VISIBLE,
  UNIQUE INDEX `CRM_UNIQUE` (`CRM` ASC) VISIBLE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Clinica`.`ESPECIALIDADE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`ESPECIALIDADE` (
  `ID_ESPECIALIDADE` INT NOT NULL,
  `DESC_ESPEC` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID_ESPECIALIDADE`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Clinica`.`MEDICO_ESPECIALIDADE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`MEDICO_ESPECIALIDADE` (
  `MEDICO_ID_MEDICO` INT NOT NULL,
  `ESPECIALIDADE_ID_ESPECIALIDADE` INT NOT NULL,
  PRIMARY KEY (`MEDICO_ID_MEDICO`,
`ESPECIALIDADE_ID_ESPECIALIDADE`),
  INDEX `fk_MEDICO_has_ESPECIALIDADE_ESPECIALIDADE1_idx`
(`ESPECIALIDADE_ID_ESPECIALIDADE` ASC) VISIBLE,
  INDEX `fk_MEDICO_has_ESPECIALIDADE_MEDICO1_idx`
(`MEDICO_ID_MEDICO` ASC) VISIBLE,
  CONSTRAINT `fk_MEDICO_has_ESPECIALIDADE_MEDICO1`
    FOREIGN KEY (`MEDICO_ID_MEDICO`)
    REFERENCES `Clinica`.`MEDICO` (`ID_MEDICO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_MEDICO_has_ESPECIALIDADE_ESPECIALIDADE1`
    FOREIGN KEY (`ESPECIALIDADE_ID_ESPECIALIDADE`)
    REFERENCES `Clinica`.`ESPECIALIDADE` (`ID_ESPECIALIDADE`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Clinica`.`HORARIO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`HORARIO` (
  `ID_AGENDA` INT NOT NULL AUTO_INCREMENT,
  `DATA_HORA` DATETIME NOT NULL,
  PRIMARY KEY (`ID_AGENDA`))
ENGINE = InnoDB;
```

```
-- Table `Clinica`.`MEDICO_HORARIO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`MEDICO_HORARIO` (
  `HORARIO_ID_AGENDA` INT NOT NULL,
  `MEDICO_ID_MEDICO` INT NOT NULL,
  PRIMARY KEY (`HORARIO_ID_AGENDA`, `MEDICO_ID_MEDICO`),
  INDEX `fk_HORARIO_has_MEDICO_MEDICO1_idx` (`MEDICO_ID_MEDICO`
ASC) VISIBLE,
  INDEX `fk_HORARIO_has_MEDICO_HORARIO1_idx` (`HORARIO_ID_AGENDA`
ASC) VISIBLE,
  CONSTRAINT `fk_HORARIO_has_MEDICO_HORARIO1`
    FOREIGN KEY (`HORARIO_ID_AGENDA`)
    REFERENCES `Clinica`.`HORARIO` (`ID_AGENDA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_HORARIO_has_MEDICO_MEDICO1`
    FOREIGN KEY (`MEDICO_ID_MEDICO`)
    REFERENCES `Clinica`.`MEDICO` (`ID_MEDICO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `Clinica`.`CONSULTA`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Clinica`.`CONSULTA` (
  `PACIENTE_idPACIENTE` INT NOT NULL,
  `MEDICO_ID_MEDICO` INT NOT NULL,
  `HORARIO_ID_AGENDA` INT NOT NULL,
  PRIMARY KEY (`PACIENTE_idPACIENTE`, `MEDICO_ID_MEDICO`,
`HORARIO_ID_AGENDA`),
  INDEX `fk_PACIENTE_has_MEDICO_MEDICO1_idx` (`MEDICO_ID_MEDICO`
ASC) VISIBLE,
  INDEX `fk_PACIENTE_has_MEDICO_PACIENTE1_idx`
(`PACIENTE_idPACIENTE` ASC) VISIBLE,
  INDEX `fk_CONSULTA_HORARIO1_idx` (`HORARIO_ID_AGENDA` ASC)
VISIBLE,
  CONSTRAINT `fk_PACIENTE_has_MEDICO_PACIENTE1`
    FOREIGN KEY (`PACIENTE_idPACIENTE`)
    REFERENCES `Clinica`.`PACIENTE` (`idPACIENTE`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PACIENTE_has_MEDICO_MEDICO1`
    FOREIGN KEY (`MEDICO_ID_MEDICO`)
    REFERENCES `Clinica`.`MEDICO` (`ID_MEDICO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_CONSULTA_HORARIO1`
    FOREIGN KEY (`HORARIO_ID_AGENDA`)
    REFERENCES `Clinica`.`HORARIO` (`ID_AGENDA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

CONCLUSÃO

Mediante as tarefas encaradas e organizadas, adquirimos o importante conhecimento sobre os processos que norteiam o desenvolvimento efetivo de sistemas de Banco de Dados. Entendemos que todos os passos são essenciais para o pleno funcionamento de um sistema deste nível, desde o primeiro contato com o cliente (público alvo), passando pelas modelagens e a criação e operação do Banco de Dados.

Acreditamos que a realização do projeto nos possibilitou uma nova visão sobre processos de desenvolvimentos de tecnologias da informação, trabalho em equipe e coordenação de atividades.