Universidade Estadual de Campinas
Bacharelado em Sistemas de Informação

Marcelo Expedito Costa de Oliveira - 248007
Miguel Gomes Fecchio - 187871

Projeto de Banco de Dados II - ST767

Projeto de banco de dados para gerenciamento de uma rede de farmácias do SUS

Limeira

2024

Descrição do sistema

O sistema a ser desenvolvido se destina às farmácias de atendimento básico do Sistema Único de Saúde (SUS) de um município.

Seu principal objetivo é o atendimento e monitoramento de pacientes e o gerenciamento de estoque. Uma farmácia é procurada por pacientes que requisitam medicamentos. Para isso, dados do cliente são inscritos no sistema, e dados de retirada dos medicamentos são gerados. Com a chegada de medicamentos, dados de estoque são inseridos e atualizados. Além disso, o banco de dados do sistema deve ser sincronizado entre as unidades.

O sistema deve permitir o cadastro, busca e atualização de usuários, pacientes e produtos de acordo com os dados relevantes. Além disso, deve gerar datas de próximas retiradas, de acordo com o tempo de tratamento de cada paciente (caso o tratamento seja contínuo), ou notificar os usuários caso um paciente esteja com uma retirada atrasada. Caso um lote de medicamentos esteja próximo da validade, um alerta também deverá ser exibido.

Essas funcionalidades visam a eficiência no monitoramento de pacientes e de estoque e uma maior usabilidade do sistema para o usuário.

Cada farmácia possui um responsável técnico, sendo ele um farmacêutico com CRF ativo. Além disso, existem auxiliares, que podem ser realocados entre unidades para demandas específicas (como por exemplo, auxiliar outra unidade durante faltas ou férias de outro funcionário).

Todas as farmácias são de responsabilidade de uma mesma prefeitura, que também se responsabiliza pelo abastecimento de estoque.

Requisitos de dados

Dois dos principais processos realizados pela farmácia são: O processo de dispensa de medicamentos - quando um paciente procura a farmácia, em posse de uma receita médica, para retirar um medicamento - e o abastecimento de estoque, cuja relação de medicamentos a serem abastecidos é encaminhada para a prefeitura.

Processo de retirada de medicamentos

Uma pessoa deve portar uma ou mais receitas médicas e é atendido por um funcionário. A receita deve estar associada a algum paciente e possuir dados sobre o tratamento.

Cada dispensa é vinculada ao nome do paciente, mas é permitido que um terceiro efetue a retirada pelo paciente, caso esteja em posse da receita e dos documentos do paciente. Contudo, caso a receita seja do tipo “controlado”, também é necessário registrar dados da pessoa que está retirando - o responsável.

Caso o tratamento seja contínuo, a data de uma próxima retirada é gerada dividindo a quantidade entregue pela posologia. Nesse caso, não é permitida uma nova retirada dos mesmos medicamentos antes da data estipulada pelo sistema.

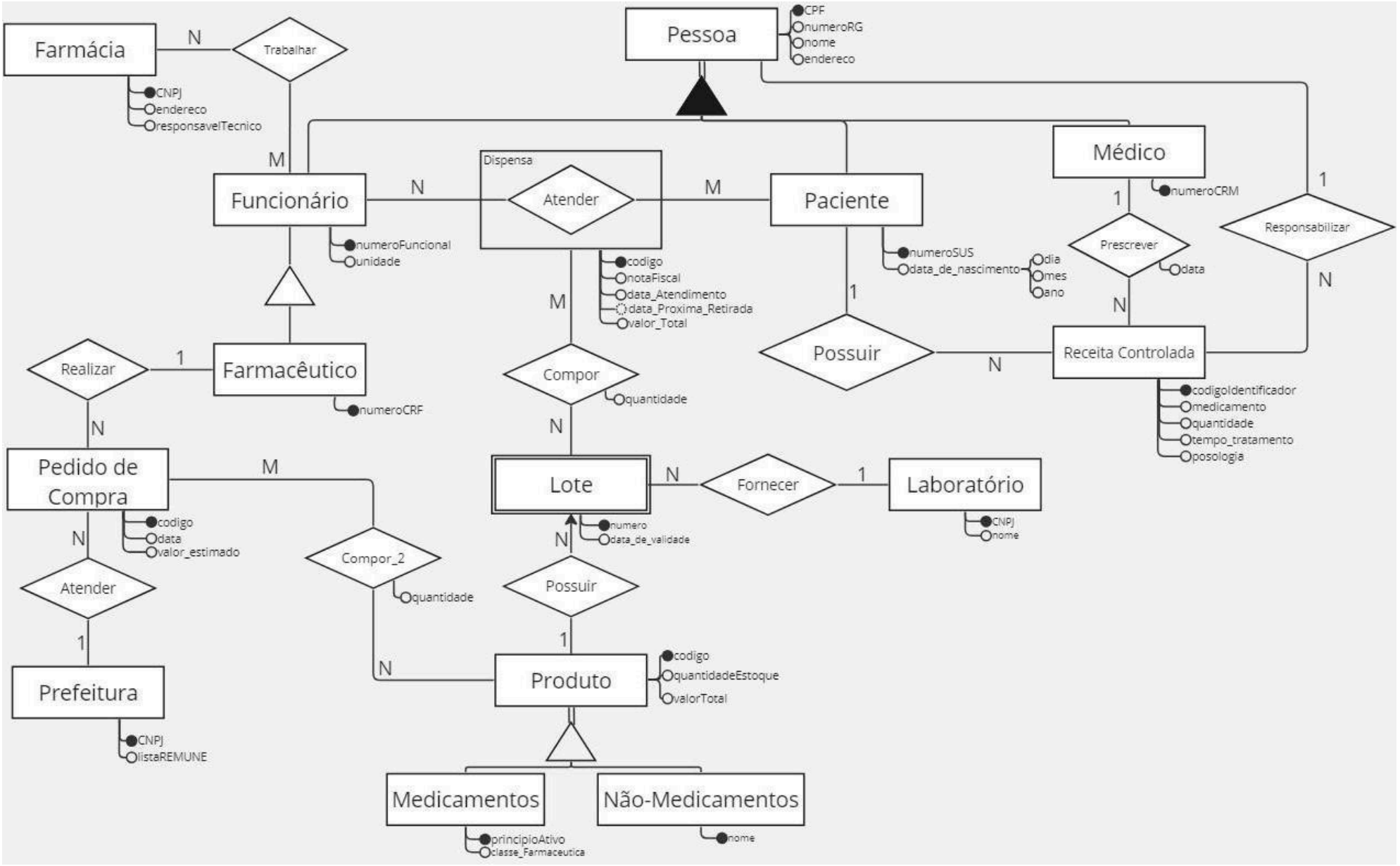
Após todos os dados serem devidamente confirmados, o pedido é gerado, possuindo um código que o identifica, e os produtos serão entregues ao cliente.

Processo de abastecimento de estoque

O farmacêutico(a) deve realizar um pedido de compra e encaminhá-lo para a Prefeitura, indicando quais medicamentos necessita e em quais quantidades. A Prefeitura atende o pedido, desde que os medicamentos façam parte de sua lista REMUME.

O pedido possui um código único que o identifica, bem como uma data na qual foi realizado e uma data na qual foi atendido.

Projeto Conceitual



Projeto Lógico

Farmácia = {CNPJ, endereco, responsavelTecnico}

Prefeitura = {CNPJ, listaRemume}

Laboratório = {CNPJ, nome}
Pessoa = {CPF, numeroRG, nome, endereco}

Funcionário = {CPF,$numeroFuncional$, unidade, tipo}

tipo: A-Farmacêutico, B-Auxiliar
CPF: chave estrangeira referenciando Pessoa

Farmacêutico = {CPF,$numeroCRF$}
CPF: chave estrangeira referenciando Funcionário

Paciente={CPF,$numeroSUS$, data_nascimento}
CPF: chave estrangeira referenciando Pessoa

Médico = {CPF,$numeroCRM$}
CPF: chave estrangeira referenciando Pessoa

Trabalhar = {CNPJ, CPF}
CNPJ: chave estrangeira referenciando Prefeitura
CPF: chave estrangeira referenciando Funcionário

Produto = {codigo, quantidadeEstoque, valorTotal, tipo}
tipo: A-Medicamento, B-Não-Medicamentos

Medicamentos = {codigo,$\underline{\underline{principioAtivo}}$, classe_farmaceutica}

codigo: chave estrangeira referenciando Produto

Não-Medicamentos = {codigo,$\underline{\underline{nome}}$}

codigo: chave estrangeira referenciando Produto

Lote = {codigo, numero, CNPJ, data_de_validade}

codigo: chave estrangeira referenciando Produto

CNPJ: chave estrangeira referenciando Laboratório

Pedido de compra = {codigo, CNPJ, CPF, data, valor_estimado}

CNPJ = chave estrangeira referenciando Prefeitura

CPF = chave estrangeira referenciando Farmacêutico

Compor_2 = {codigo_pedido, codigo_produto, quantidade}

codigo_pedido: chave estrangeira referenciando Pedido de Compra

codigo_produto: chave estrangeira referenciando Produto

Dispensa = {codigo, CPF_func, CPF_pac, notaFiscal, data_atendimento, data_proxima_retirada, valor_total}

CPF_func: Chave estrangeira referenciando funcionário

CPF_pac: Chave estrangeira referenciando paciente

Compor = {cod_dis, cod_lote, numero, quantidade}

cod_dis = chave estrangeira referenciando Dispensa

cod_lote = chave estrangeira referenciando Lote

numero = chave estrangeira referenciando Lote

Receita Controlada = {codigoIdentificador, medicamento, quantidade, tempo_tratamento, posologia, CPF_Medico, CPF_Pessoa, CPF_Paciente, data}

CPF_Medico = chave estrangeira referenciando Médico

CPF_Pessoa = chave estrangeira referenciando Pessoa

CPF_Paciente = chave estrangeira referenciando Paciente

Projeto físico

Unset

```
create database projetobdiifinal
go

use projetobdiifinal
go

create table farmacia (
cnpj char(18) not null,
endereco char(50) not null,
responsaveltecnico char(30) not null,
primary key (cnpj)
)
go

create table prefeitura (
cnpj char(18) not null,
listaremume char(19) not null,
primary key (cnpj)
)
go

create table laboratorio (
cnpj char(18) not null,
nome char(30) not null,
primary key (cnpj)
)
go

create table pessoa (
cpf char(14) not null,
rg char(12) not null,
nome char(30) not null,
endereco char(50) not null,
primary key (cpf)
)
go

create table funcionario (
cpf char(14) not null,
numerofuncional int not null unique,
unidade char(30) not null,
area char(12) not null,
primary key (cpf),
foreign key (cpf) references pessoa
)
go

create table farmaceutico (
cpf char(14) not null,
numerocrf char(5) not null unique,
primary key (cpf),
foreign key (cpf) references pessoa
)
```

```
go

create table paciente (
cpf char(14) not null,
numerosus char(15) not null unique,
data_nasc date not null,
primary key (cpf),
foreign key (cpf) references pessoa
)
go

create table medico (
cpf char(14) not null,
numerocrm char(8) not null unique,
primary key (cpf),
foreign key (cpf) references pessoa
)
go

create table produto (
codigo int not null,
qtdestoque int not null,
valortotal float not null,
tipo char(16) not null,
primary key (codigo)
)
go

create table medicamento (
codigo int not null,
principioativo char(20) not null unique,
classefarmaceutica char(20) not null,
primary key (codigo),
foreign key (codigo) references produto
)
go

create table naomedicamento (
codigo int not null,
nome char(30) not null unique,
primary key (codigo),
foreign key (codigo) references produto
)
go

create table lote (
codigo int not null,
numero int not null,
cnpj char(18) not null,
data_val date not null,
primary key (codigo, numero),
foreign key (codigo) references produto,
foreign key (cnpj) references laboratorio
)
go
```

```
create index ix_cnpjlote
on lote (cnpj)
go

create table pedidodecompra (
codigo int not null,
cnpj char(18) not null,
cpf char(14) not null,
data date not null,
valorestimado float not null,
primary key (codigo),
foreign key (cnpj) references prefeitura,
foreign key (cpf) references farmaceutico
)
go

create index ix_cnpjpedido
on pedidodecompra(cnpj)
go

create index ix_cpfpedido
on pedidodecompra(cpf)
go

create table compra (
codigo_pedido int not null,
codigo_produto int not null,
quantidade int not null,
primary key (codigo_pedido, codigo_produto),
foreign key (codigo_pedido) references pedidodecompra,
foreign key (codigo_produto) references produto
)
go

create table dispensa (
codigo int not null,
cpf_func char(14) not null,
cpf_pac char(14) not null,
notafiscal char(44) not null,
data_atendimento date not null,
data_prox_retirada date not null,
valortotal float not null,
primary key (codigo),
foreign key (cpf_func) references funcionario,
foreign key (cpf_pac) references paciente
)
go

create index ix_cpffuncdispensa
on dispensa(cpf_func)
go

create index ix_cpffuncpaciente
on dispensa(cpf_pac)
go
```

```
create table distribuicao (
cod_dis int not null,
cod_lote int not null,
numero int not null,
quantidade int not null,
primary key (cod_dis, cod_lote, numero),
foreign key (cod_dis) references dispensa,
foreign key (cod_lote, numero) references lote,
)
go

create table receita_controlada(
codigoidentificador int not null,
medicamento char(30) not null,
quantidade int not null,
tempo_tratamento char(20) not null,
posologia char(20) not null,
cpf_medico char(14) not null,
cpf_pessoa char(14) not null,
cpf_paciente char(14) not null,
data date not null
primary key (codigoidentificador),
foreign key (cpf_medico) references medico,
foreign key (cpf_pessoa) references pessoa,
foreign key (cpf_paciente) references paciente
)
go

create index ix_cpfmed_receitacont
on receita_controlada(cpf_medico)
go

create index ix_cpfpes_receitacont
on receita_controlada(cpf_pessoa)
go

create index ix_cpfpac_receitacont
on receita_controlada(cpf_paciente)
go
```

Manipulação de dados

Visões

Unset

```
USE projetobdiifinal
GO


-- view de pacientes e datas de retiradas de medicamentos
CREATE VIEW visao_paciente_retirada_medicamento AS
SELECT
    pac.cpf AS paciente_cpf,
    rc.medicamento,
    d.data_atendimento AS data_retirada
FROM
    dispensa d
INNER JOIN
    paciente pac ON d.cpf_pac = pac.cpf
INNER JOIN
    receita_controlada rc ON rc.cpf_paciente = pac.cpf
WHERE
    rc.medicamento IS NOT NULL;
GO

SELECT * FROM visao_paciente_retirada_medicamento;
GO


-- view de lotes de medicamentos
CREATE VIEW visao_medicamento_lote AS
SELECT
    m.principioativo,
    m.classefarmaceutica,
    l.numero AS lote_numero,
    l.data_val AS data_validade,
    pr.qtdestoque
FROM
    medicamento m
INNER JOIN
    produto pr ON m.codigo = pr.codigo
INNER JOIN
    lote l ON pr.codigo = l.codigo
GO

SELECT * FROM visao_medicamento_lote;
GO


-- view de clientes atendidos por funcionarios
CREATE VIEW visao_funcionario_paciente_atendido AS
SELECT
    f.cpf AS funcionario_cpf,
    pac.cpf AS paciente_cpf,
    d.data_atendimento
FROM
    dispensa d
```

```
INNER JOIN
    funcionario f ON d.cpf_func = f.cpf
INNER JOIN
    paciente pac ON d.cpf_pac = pac.cpf;
GO

SELECT * FROM visao_funcionario_paciente_atendido;
GO


-- view de pedidos de produtos
CREATE VIEW visao_pedido_produtos AS
SELECT
    pdc.codigo AS pedido_codigo,
    pdc.data,
    pdc.valorestimado,
    c.codigo_produto,
    c.quantidade
FROM
    pedidodecompra pdc
INNER JOIN
    compra c ON pdc.codigo = c.codigo_pedido;
GO

SELECT * FROM visao_pedido_produtos;
GO


-- view de pacientes e medicamentos
CREATE VIEW visao_paciente_medicamento AS
SELECT
    pac.cpf AS paciente_cpf,
    rc.medicamento,
    rc.quantidade
FROM
    receita_controlada rc
INNER JOIN
    paciente pac ON rc.cpf_paciente = pac.cpf;
GO

SELECT * FROM visao_paciente_medicamento;
GO
```

Procedimentos armazenados

Unset

```
use projetobdiifinal
go

--Procedimento para adicionar medicamentos sem erros
create procedure adicionar_medicamento
@codigo int,
@principioativo char(20),
@classefarmaceutica char(20),
@qtdestoque int,
@valortotal float,
@tipo char(16)
as
begin
--Verificando se o produto existe
if not exists (select 1 from produto where codigo = @codigo)
begin
--Inserindo na tabela de produto se não existir
insert into produto(codigo, qtdestoque, valortotal, tipo)
values (@codigo, @qtdestoque, @valortotal, @tipo)
end
--Verifica se princípio ativo é único
if not exists (select 1 from medicamento where principioativo =
@principioativo)
begin
--Inserindo na tabela de medicamento caso o princípio ativo
seja único
insert into medicamento(codigo, principioativo,
classefarmaceutica)
values (@codigo, @principioativo, @classefarmaceutica)
print 'Medicamento adicionado com sucesso.';
return 0;
end;
else
begin
--Erro caso o princípio ativo já exista na tabela
print 'Já existe um medicamento com esse princípio ativo';
return 1;
end
end;
go

--Procedimento para adicionar não medicamentos sem erros
create procedure adicionar_naomedicamento
@codigo int,
@nome char(30),
@qtdestoque int,
@valortotal float,
@tipo char(16)
as
begin
--Verificando se o produto existe
if not exists (select 1 from produto where codigo = @codigo)
begin
--Inserindo na tabela de produto se não existir
```

```
insert into produto(codigo, qtdestoque, valortotal, tipo)
values (@codigo, @qtdestoque, @valortotal, @tipo)
end
--Verifica se já tem produto com o mesmo nome
if not exists (select 1 from naomedicamento where nome = @nome)
begin
--Inserindo na tabela de não medicamento caso seja produto novo
insert into naomedicamento(codigo, nome)
values (@codigo, @nome)
print 'Não medicamento adicionado com sucesso.';
return 0;
end;
else
begin
--Erro caso o produto já exista na tabela
print 'Já existe um não medicamento com esse nome';
return 1;
end
end;
go

--Procedimento para fazer a dispensa do medicamento
create procedure dispensar_medicamento
@codigo_dispensa int,
@cpf_func char(14),
@cpf_pac char(14),
@notafiscal char(44),
@data_atendimento date,
@data_prox_retirada date,
@valortotal float,
@codigo_produto int,
@quantidade int

as
begin
--Verificando o estoque
declare @qtdestoque int
select @qtdestoque = qtdestoque from produto
where @codigo_produto = codigo

if @qtdestoque >= @quantidade
begin

--Insere dispensa na tabela
insert into dispensa (codigo, cpf_func, cpf_pac, notafiscal,
data_atendimento, data_prox_retirada, valortotal)
values (@codigo_dispensa, @cpf_func, @cpf_pac, @notafiscal,
@data_atendimento, @data_prox_retirada, @valortotal)

--Insere distribuicao na tabela
insert into distribuicao (cod_dis, cod_lote, numero,
quantidade)
values (@codigo_dispensa, @codigo_produto, @codigo_dispensa,
@quantidade)
print 'Dispensa concluída'
return 0
```

```
end
else
begin
--Mensagem caso não seja possível efetuar a dispensa
print 'Quantidade indisponível no estoque'
return 1
end
end;
go

--Procedimento para gerar relatório mensal
create procedure gerar_relatoriomensal
@mes int,
@ano int
as
begin
--Seleciona código do produto, valor total vendido e quantas
vezes foi vendido
select p.codigo, sum(d.valortotal) as valorfinal,
count(d.codigo) as numero_vendas
from produto p inner join dispensa d
on p.codigo = d.codigo
where month(d.data_atendimento) = @mes and
year(d.data_atendimento) = @ano
group by p.codigo
end;
go

--Procedimento para verificar lotes que vão vencer
create procedure verificar_validadelotes
@dias int
as
begin
--Seleciona lotes que vencem dentro da quantidade de dias
selecionada
select * from lote
where datediff(day, getdate(), data_val) <= @dias
end;
go
```

Gatilhos

Unset

```
USE projetobdiifinal
GO


-- Trigger para atualizar a quantidade em estoque apos compras
CREATE TRIGGER atualiza_estoque
     ON compra
     AFTER INSERT
     AS
```

```
BEGIN
    DECLARE @codigo_produto INT, @quantidade INT;

    SELECT @codigo_produto = codigo_produto, @quantidade =
quantidade
    FROM inserted;

    -- Confere se o estoque eh suficiente
    IF EXISTS (
        SELECT 1
        FROM produto
        WHERE codigo = @codigo_produto AND qtdestoque <
@quantidade
    )
    BEGIN
        RAISERROR('Estoque insuficiente para o produto.', 16,
1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        -- Atualiza a quantidade em estoque
        UPDATE produto
        SET qtdestoque = qtdestoque - @quantidade
        WHERE codigo = @codigo_produto;
    END
END;
GO

-- Trigger para prevenir a insercao de lotes vencidos
CREATE TRIGGER verifica_data_lote
     ON lote
     AFTER INSERT
     AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE data_val < GETDATE()
    )
    BEGIN
        RAISERROR('Lote com data de validade expirada não pode
ser inserido.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- Trigger para bloquear a dispensa antes da data permitida
CREATE TRIGGER verifica_data_retirada
     ON dispensa
     AFTER INSERT
     AS
BEGIN
    DECLARE @cpf_pac CHAR(14), @data_atendimento DATE,
@data_prox_retirada DATE;
```

```
    SELECT @cpf_pac = cpf_pac, @data_atendimento =
data_atendimento, @data_prox_retirada = data_prox_retirada
    FROM inserted;

    IF EXISTS (
        SELECT 1
        FROM dispensa
        WHERE cpf_pac = @cpf_pac
          AND data_atendimento < @data_prox_retirada
    )
    BEGIN
        RAISERROR('A retirada não pode ser feita antes da data
estipulada.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- Trigger para calcular automaticamente a proxima data de
dispensa para tratamentos continuos
CREATE TRIGGER define_proxima_retirada
     ON dispensa
     AFTER INSERT
     AS
BEGIN
    DECLARE @codigo INT, @cpf_paciente CHAR(14), @medicamento
CHAR(30), @posologia INT, @quantidade INT;

    SELECT @codigo = codigo, @cpf_paciente = cpf_pac
    FROM inserted;

    -- Recebendo a posologia e a quantidade de
'receita_controlada'
    SELECT @medicamento = medicamento, @posologia =
CAST(posologia AS INT), @quantidade = quantidade
    FROM receita_controlada
    WHERE cpf_paciente = @cpf_paciente;

    IF @posologia > 0
    BEGIN
        -- Calculando a proxima data de dispensa
        DECLARE @dias_para_prox_retirada INT = @quantidade /
@posologia;
        UPDATE dispensa
        SET data_prox_retirada = DATEADD(DAY,
@dias_para_prox_retirada, data_atendimento)
        WHERE codigo = @codigo;
    END
END;
GO
```