Faculdade de Tecnologia do Estado de São Paulo – FATEC Dep. Ary Fossen

Projeto de banco de dados MySQL para o Banco de Tintas da Fatec Jundiaí

INTEGRANTES:

ARTHUR RENAN GUTIERREZ DIAS PEREIRA

JONATHAN JOSÉ GONZALEZ

PATRICK HIROSHI KATSUTA

RUAN VITOR SANTOS DE SOUZA

Profa. Orientadora:

LUCIANA FERREIRA BAPTISTA

JUNDIAÍ/2024

[UNIVERSO: Banco de Tintas 3](#_Toc182857694)

[DESCRIÇÃO 3](#_Toc182857695)

[Diagrama Entidade Relacionamento – DER 6](#_Toc182857696)

[Modelo Entidade Relacionamento – MER 7](#_Toc182857697)

[IMPLEMENTAÇÃO 8](#_Toc182857698)

# UNIVERSO: Banco de Tintas

## DESCRIÇÃO

O projeto a ser desenvolvido visa a criação de um Website para o projeto Banco de Tintas da Fatec Jundiaí - Deputado Ary Fossen, que tem como objetivo, conectar doadores de tintas comerciais não-utilizadas (a base de água), as pessoas que precisam delas - evitando o descarte inadequado e auxiliando famílias em situação de vulnerabilidade social.

O banco de dados registra os usuários que utilizam o web serviço (desenvolvido na linguagem de programação PHP com integração deste banco de dados MySQL) por meio da tabela **Usuarios**, a qual armazena: *id, usuario\_nome, usuario\_cep, usuario\_endereco, usuario\_endereco\_num, usuario\_endereco\_complemento, usuario\_bairro, usuario\_cidade, usuario\_estado, usuario\_email, senha\_hash, eh\_empresa, usuario\_documento e telefone*. O atributo eh\_empresa serve para identificar se o usuário cadastrado é uma Organização, no qual o registro do usuário é associado à tabela **Organizacao** (que contém os campos *id\_organizacao, tipo\_organizacao e area\_atuacao*) por meio de *id\_organizacao*.

O serviço web ainda prevê a liberação de acesso dos usuários ao sistema por meio da identificação do tipo de usuário, cujas informações são armazenadas dentro da tabela **Usuario\_tipos** (*usuario\_id, e tipo*), com associação por meio do atributo usuario\_id com a o atributo *id* da tabela **Usuarios**. O atributo *tipo* é um *ENUM(‘Gestor’, ‘Monitor’, ‘Doador’, ‘Beneficiario’)*, o qual identifica quais permissões o usuario terá no sistema.

A tabela **Monitor** (*id\_monitor, registro, curso, monitor\_dia, monitor\_periodo, eh\_gestor*) é associada por meio do atributo *id\_monitor* ao atributo *id* da tabela Usuarios. O atributo *eh\_gestor* identifica que o monitor cadastrado é um *‘Gestor’*, o qual possui mais responsabilidades na gestão da administração do sistema web, além de ser responsável por outros Monitores, cuja associação é possível observar por meio da tabela **Gestor** (*id\_gestor, id\_monitor*), sendo que *id\_gestor* identifica o Gestor, responsável por outro *id\_monitor*.

A tabela **Tintas** exerce um papel fundamental neste sistema, pois armazena as informações das tintas que são recebidas de *‘Doadores’* por meio de doações e que posteriormente serão entregues a *‘Beneficiarios’*. A tabela possui as informações: *id\_tintas, nome\_tintas, marca, linha, acabamento, quantidade\_tintas\_disponivel, data\_validade\_tintas, mistura, historico, data\_criacao, excluído, codigo\_RGB*. O atributo *mistura* indica se o registro foi alterado no sistema, o que significa que ele é a união de dois registros, cujo processo é armazenado em *historico*. O atributo *excluido* é um “soft delete”, o que significa que, após o registro ser entregue totalmente (o atributo *quantidade\_tintas\_disponivel* for igual a zero ou for utilizado para uma *mistura*), ele será marcado como 1 (um), simbolizando que ele acabou e não será mostrado nas consultas de tintas disponíveis, mas permanecerá no banco de dados para fins de verificações de quantas tintas já passaram foram processadas e ainda poderão ser removidas permanentemente posteriormente, com suas informações salvas em registros para consulta. O atributo *codigo\_RGB* é importante para a questão visual do web serviço, o qual armazena a informação da cor para que o utilizador possa ter uma informação visual da tinta que irá receber.

A tabela **Doacao** registra as doações de tintas, relacionando os Doadores, Monitores e as Tintas recebidas. Possui os atributos *id\_doacao, id\_doador, id\_monitor, data\_doacao, local\_doado* e relaciona-se com as tabelas **Usuario** (*id*) e **Monitor** (*id\_monitor*) por meio dos campos *id\_doador* e *id\_monitor* respectivamente. Como existe a possibilidade de mais de uma tinta ser doada em uma única doação, a tabela **Doacao\_tintas** armazena os registros das tintas que estão presentes na doação por meio dos seguintes atributos: *id\_doacao, id\_tintas, quantidade\_tintas\_doada*. Essa tabela é ligada com **Doacao** (*id\_doacao*) e **Tintas** (*id\_tintas*) por meio das chaves *id\_doacao* e *id\_tintas*, respectivamente.

**Solicitacao** é uma tabela para registrar os pedidos de retirada de tintas por *‘Beneficiario’*s interessados em receber tintas deste projeto. Esse registro é feito por meio dos seguintes atributos: *id\_solicitação, id\_beneficiario, data\_solicitacao, justificativa* e *excluido*. O atributo *excluido* desta tabela segue o mesmo intuito do atributo da tabela **Tintas** de mesmo nome. O atributo *id\_beneficiario* é uma associação a **Usuarios** (*id*). E, assim como uma doação pode possuir mais de uma tinta em sua operação, a tabela **Solicitacao\_tintas** possui a mesma finalidade, ou seja, armazenar as tintas que serão solicitadas neste pedido. Isso ocorre por meio dos atributos *id\_solicitacao, id\_tintas* e *quantidade*, sendo que possui associação direta com os campos **Solicitacao** (*id\_solicitacao*) e **Tintas** (*id\_tintas*).

No entanto, o processo de solicitação requer uma **Analise**, para verificação da justificativa da quantidade de tintas solicitadas, visto que uma **Organizacao** também pode fazer um requerimento. A tabela **Analise** armazena os seguintes atributos: *id\_analise, id\_gestor, id\_solicitacao, status\_solicitacao, data\_analise* e *justificativa*. Para identificar a Solicitacao (*id\_solicitacao*) por meio do campo id\_solicitacao e o *‘Gestor’* **Monitor** (*id\_monitor*) que fez a análise da solicitação por meio de *id\_gestor*. A análise pode ser *‘negada’* pela justificativa não ser coesa, ou *‘parcialmente aprovada’* devido a indisponibilidade do total de tintas requerida, mas caso tudo esteja correto, o pedido será ‘aprovado’ e prosseguirá para a **Entrega**. A tabela Entrega armazena as informações: *id\_entrega, id\_analise* (a qual identifica de qual **Solicitacao** (*id\_solicitacao*) foi a **Analise** (*id\_analise*)), *id\_monitor* (quem será o **Monitor** (*id\_monitor*) que realizará a entrega ao **Usuario** (*id*) **Usuario\_tipo** (*tipo = ’Beneficiario’*)), *dia\_semana\_entrega, horario\_entrega, status\_entrega e local\_entrega*. **Entrega\_tintas** tem por função elencar todas as tintas que serão entregues, visto que o pedido pode ter sido parcialmente aprovado e o *‘Beneficiario’* acordou em receber apenas parte de sua **Solicitacao**, isso ocorre por meio dos seguintes campos: *id\_entrega, id\_tintas* e *quantidade*, sendo que *id\_entrega* refere a **Entrega** (*id\_entrega*) e *id\_tintas* é referência de **Tintas** (*id\_tintas*).

# Diagrama Entidade Relacionamento – DER

Diagrama, Esquemático

Descrição gerada automaticamente

# Modelo Entidade Relacionamento – MER

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

# IMPLEMENTAÇÃO

CREATE DATABASE Banco\_de\_Tintas;

USE Banco\_de\_Tintas;

*-- Usuarios: Armazena as informações sobre os usuários do sistema do Banco de Tintas*

CREATE TABLE Usuarios (

id INT AUTO\_INCREMENT PRIMARY KEY,

usuario\_nome VARCHAR (100) NOT NULL,

usuario\_cep VARCHAR(8), *-- Armazena o CEP (apenas números)*

usuario\_endereco VARCHAR(200), *-- Armazena o logradouro*

usuario\_endereco\_num VARCHAR(20), *-- Armazena o número*

usuario\_endereco\_complemento VARCHAR(100), *-- Armazena o complemento do logradouro*

usuario\_bairro VARCHAR(100), *-- Bairro do usuário*

usuario\_cidade VARCHAR(100), *-- Cidade do usuário*

usuario\_estado VARCHAR(2), *-- UF (estado)*

usuario\_email VARCHAR (100) NOT NULL, *-- Login*

senha\_hash VARCHAR(255), *-- Login*

eh\_empresa TINYINT DEFAULT 0, -- Indica se é empresa (1 = Sim, 0 = Não)

usuario\_documento VARCHAR (14), *-- CPF ou CNPJ*

telefone VARCHAR (11)

);

*-- Usuario\_tipos: Como existe a possibilidade de um Doador também ser um Beneficiario, faz-se necessário a utilização desta tabela intermediária para evitar a criação de outro cadastro para alguém que já consta no sistema.*

CREATE TABLE Usuario\_Tipos (

usuario\_id INT NOT NULL,

tipo ENUM('Gestor', 'Monitor', 'Doador', 'Beneficiario') NOT NULL,

PRIMARY KEY (usuario\_id, tipo),

FOREIGN KEY (usuario\_id) REFERENCES Usuarios(id) ON DELETE CASCADE

);

*-- Organizacao: Armazena informações básicas sobre organizações (doadoras ou receptoras).*

CREATE TABLE Organizacao(

id\_organizacao INT PRIMARY KEY,

tipo\_organizacao VARCHAR (100),

area\_atuacao VARCHAR (100),

FOREIGN KEY (id\_organizacao) REFERENCES Usuarios (id)

);

*-- Monitor: Registra informações sobre os monitores responsáveis pelo sistema, incluindo um relacionamento hierárquico de responsabilidade.*

CREATE TABLE Monitor (

id\_monitor INT PRIMARY KEY,

registro VARCHAR (20) NOT NULL, *-- RA do aluno ou código/cadastro do professor*

curso VARCHAR (100),

monitor\_dia INT (5), *-- Dia da semana que o monitor atua*

monitor\_periodo VARCHAR(100), *-- Período que o monitor atua: 'Manhã', 'Tarde', 'Noite'),*

eh\_gestor TINYINT DEFAULT 0,

FOREIGN KEY (id\_monitor) REFERENCES Usuarios(id)

);

*-- Gestor: Registra os Gestores (professores) responsáveis pelos monitores. Inseridos pelo ADM*

CREATE TABLE Gestor(

id\_gestor INT,

id\_monitor INT,

PRIMARY KEY (id\_gestor, id\_monitor),

FOREIGN KEY (id\_gestor) REFERENCES Monitor (id\_monitor),

FOREIGN KEY (id\_monitor) REFERENCES Monitor (id\_monitor)

);

*-- Tintas: Contém informações sobre as tintas, incluindo data de validade e se a tinta é uma mistura.*

CREATE TABLE Tintas (

id\_tintas INT AUTO\_INCREMENT PRIMARY KEY,

nome\_tintas VARCHAR (20),

marca VARCHAR (50),

linha VARCHAR (50),

acabamento VARCHAR (50),

quantidade\_tintas\_disponivel decimal(5,2), *-- Medida em litros. Para medidas menores do que um litro, utilizar unidade decimal (0,7L = 700ml)*

data\_validade\_tintas date NOT NULL,

mistura TINYINT NOT NULL DEFAULT 0,

historico VARCHAR (200), *-- Caso tenha havido mistura, informa quais tintas foram utilizadas*

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Controle de inserção no banco de dados*

excluido TINYINT DEFAULT 0, *-- Soft delete. Se 1, significa que o item foi excluído, mas não removido do banco de dados*

codigo\_RGB VARCHAR (100) *-- Recebe o valor RGB a partir do informado no sistema*

);

*-- Doacao: Registra as doações de tintas, relacionando doadores, monitores e tintas.*

CREATE TABLE Doacao (

id\_doacao INT AUTO\_INCREMENT PRIMARY KEY,

id\_doador INT,

id\_monitor INT,

data\_doacao TIMESTAMP,

local\_doado VARCHAR (100), *-- FATEC ou Posto de Coleta (Lojas Saci Tintas)*

FOREIGN KEY (id\_doador) REFERENCES Usuarios (id),

FOREIGN KEY (id\_monitor) REFERENCES Monitor (id\_monitor)

);

*-- Doacao\_tintas: registra as tintas recebidas em Doação*

CREATE TABLE Doacao\_tintas (

id\_doacao INT,

id\_tintas INT,

quantidade\_tintas\_doada decimal(5,2), *-- Medida em litros. Para medidas menores do que um litro, utilizar unidade decimal (0,7L = 700ml)*

PRIMARY KEY (id\_doacao, id\_tintas),

FOREIGN KEY (id\_doacao) REFERENCES Doacao (id\_doacao),

FOREIGN KEY (id\_tintas) REFERENCES Tintas (id\_tintas)

);

*-- Solicitacao: Registra a solicitação de retirada de tintas pelo Beneficiario (sujeito a aprovação do gestor na tabela Analisa)*

CREATE TABLE Solicitacao (

id\_solicitacao INT AUTO\_INCREMENT,

id\_beneficiario INT,

data\_solicitacao TIMESTAMP, *-- Indica quando a solicitação entrou no sistema*

justificativa VARCHAR (200),

excluido TINYINT DEFAULT 0, *-- Soft delete. Se 1, significa que a solicitação foi excluída, mas não removida do banco de dados. Funciona para verificar os pedidos recebidos*

PRIMARY KEY (id\_solicitacao, data\_solicitacao),

FOREIGN KEY (id\_beneficiario) REFERENCES Usuarios (id)

);

*-- Solicitacao\_tintas: Registra as tintas da Solicitação do Beneficiario*

CREATE TABLE Solicitacao\_tintas (

id\_solicitacao INT,

id\_tintas INT,

quantidade DECIMAL (5,2),

PRIMARY KEY (id\_solicitacao, id\_tintas),

FOREIGN KEY (id\_solicitacao) REFERENCES Solicitacao (id\_solicitacao),

FOREIGN KEY (id\_tintas) REFERENCES Tintas (id\_tintas)

);

*-- Análise da Solicitação: O Gestor analisa a Solicitação de retirada de tintas, decidindo se aprova ou não. Caso haja indisponibilidade de determinada tinta no sistema ou outros casos (ex.: quantidade de tinta solicitada), o Gestor irá entrar em contato com o Beneficiario para combinar a situação)*

CREATE TABLE Analise (

id\_analise INT AUTO\_INCREMENT PRIMARY KEY,

id\_gestor INT, *-- Identificação do Gestor, responsável por outros monitores*

id\_solicitacao INT,

status\_solicitacao ENUM ('Em analise', 'Aprovado', 'Parcialmente aprovado', 'Negado') NOT NULL DEFAULT 'Em analise',

data\_analise TIMESTAMP,

justificativa VARCHAR(200), *-- Motivo de não aprovar uma solicitação*

FOREIGN KEY (id\_gestor) REFERENCES Monitor (id\_monitor),

FOREIGN KEY (id\_solicitacao) REFERENCES Solicitacao (id\_solicitacao)

);

*-- Entrega: Registra as entregas de tintas, relacionando o monitor que irá realizar a entrega, as tintas e o status da entrega. Pode ser preenchida na hora da retirada, mas não deve ser negligenciada.*

CREATE TABLE Entrega (

id\_entrega INT AUTO\_INCREMENT PRIMARY KEY,

id\_analise INT,

id\_monitor INT, *-- Define o monitor que realizará a entrega*

dia\_semana\_entrega INT, *-- Dias da semana para retirada*

horario\_entrega INT, *-- Período de retirada da tinta, de acordo com a disponibilidade*

status\_entrega ENUM('Agendado', 'Concluído', 'Cancelado') DEFAULT 'Agendado', *-- Campo a ser alterado conforme a necessidade*

local\_entrega VARCHAR (100),

FOREIGN KEY (id\_analise) REFERENCES Analise (id\_analise),

FOREIGN KEY (id\_monitor) REFERENCES Monitor (id\_monitor),

);

*-- Entrega\_tintas: Registra as tintas que serão entregues.*

CREATE TABLE Entrega\_tintas (

id\_entrega INT,

id\_tintas INT,

quantidade DECIMAL (5,2),

PRIMARY KEY (id\_entrega, id\_tintas),

FOREIGN KEY (id\_entrega) REFERENCES Entrega (id\_entrega),

FOREIGN KEY (id\_tintas) REFERENCES Tintas (id\_tintas)

);

*-- Doador anônimo*

INSERT INTO `usuarios` (usuario\_nome)

VALUES ('Doador Anônimo');

*-- Índices para consultas*

CREATE INDEX idx\_usuario\_email ON Usuarios(usuario\_email);

CREATE INDEX idx\_doacao\_data ON Doacao(data\_doacao);

CREATE INDEX idx\_solicitacao\_tintas ON Solicitacao\_tintas(id\_solicitacao, id\_tintas);

*-- Consulta de quem será o Beneficiario a receber a doação na Entrega*

SELECT

u.usuario\_nome AS Beneficiario,

e.id\_entrega

FROM

Entrega e

INNER JOIN Analise a ON e.id\_analise = a.id\_analise

INNER JOIN Solicitacao s ON a.id\_solicitacao = s.id\_solicitacao

INNER JOIN Usuarios u ON s.id\_beneficiario = u.id;

*-- Relatórios*

*-- Tintas mais doadas:*

SELECT t.nome\_tintas, SUM(dt.quantidade\_tintas\_doada) AS total\_doado

FROM Tintas t

INNER JOIN Doacao\_tintas dt ON t.id\_tintas = dt.id\_tintas

GROUP BY t.nome\_tintas

ORDER BY total\_doado DESC;

*-- Doadores mais ativos:*

SELECT u.usuario\_nome, COUNT(\*) AS numero\_doacoes

FROM Usuarios u

INNER JOIN Doacao d ON u.id = d.id\_doador

GROUP BY u.usuario\_nome

ORDER BY numero\_doacoes DESC;

*-- Tempo médio de processamento de solicitações:*

SELECT AVG(TIMESTAMPDIFF(DAY, s.data\_solicitacao, a.data\_analise)) AS dias\_para\_aprovacao

FROM Solicitacao s

INNER JOIN Analise a ON s.id\_solicitacao = a.id\_solicitacao

WHERE a.status\_solicitacao = 'Aprovado';

*-- Análises*

*-- Tintas próximas do vencimento por tipo:*

SELECT t.nome\_tintas, t.marca, t.data\_validade\_tintas

FROM Tintas t

WHERE t.data\_validade\_tintas BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY)

ORDER BY t.data\_validade\_tintas;

*-- Solicitações pendentes por tipo de usuário:*

SELECT s.id\_solicitacao, u.usuario\_nome, a.status\_solicitacao

FROM Solicitacao s

INNER JOIN Usuarios u ON s.id\_beneficiario = u.id

LEFT JOIN Analise a ON s.id\_solicitacao = a.id\_solicitacao

WHERE a.status\_solicitacao = 'Em análise'

LIMIT 0, 25;

*-- Consulta por nome de tinta similar em Solicitação*

DELIMITER $$

CREATE PROCEDURE procurar\_tinta(IN nome\_buscado VARCHAR(50))

BEGIN

SELECT \* FROM Tintas

WHERE nome\_tintas LIKE CONCAT('%', nome\_buscado, '%');

END $$

DELIMITER ;

*/\**

*-- Exemplo da chamada do procedimento*

*CALL procurar\_tinta('verde');*

*-- Exemplo da chamada na interface web (PHP)*

*<?php*

*$nome\_tinta = $\_POST['nome\_tinta'];*

*$sql = "SELECT \* FROM Tintas WHERE nome\_tintas LIKE '%" . $nome\_tinta . "%'";*

*// Executar a consulta e exibir os resultados*

*\*/*

*-- Local de Maior Coleta de Tintas*

SELECT local\_doado, COUNT(\*) AS total\_doacoes *-- COUNT(\*) Conta o número total de doações para cada local.*

FROM Doacao

GROUP BY local\_doado *-- Agrupa os dados por local de doação.*

ORDER BY total\_doacoes DESC *-- Ordena os resultados em ordem decrescente de quantidade de doações.*

LIMIT 1; *-- Retorna apenas a primeira linha, que corresponde ao local com a maior quantidade de doações. Para visualização de mais locais, alterar o valor de LIMIT*

*-- Regiões (CEP) com Maior Procura de Beneficiários*

SELECT usuario\_cep, COUNT(\*) AS total\_solicitacoes

FROM Usuarios u

INNER JOIN Solicitacao s ON u.id = s.id\_beneficiario *-- Relaciona as tabelas Usuarios e Solicitacao para obter os CEPs dos beneficiários.*

GROUP BY usuario\_cep -- Agrupa os dados por CEP.

ORDER BY total\_solicitacoes DESC; *-- Ordena os resultados em ordem decrescente de quantidade de solicitações.*

*-- Tintas mais solicitadas em determinada região:*

SELECT t.nome\_tintas, COUNT(\*) AS total\_solicitacoes

FROM Tintas t

INNER JOIN Solicitacao\_tintas st ON t.id\_tintas = st.id\_tintas

INNER JOIN Solicitacao s ON st.id\_solicitacao = s.id\_solicitacao

WHERE s.id\_beneficiario IN (SELECT id FROM Usuarios WHERE usuario\_cep LIKE '010%') *-- Exemplo: CEPs que começam com 010*

GROUP BY t.nome\_tintas

ORDER BY total\_solicitacoes DESC;

*/\**

*//Processo de segurança no PHP*

*// Exemplo em PHP usando a biblioteca password\_hash*

*$senha = 'minhaSenhaSegura';*

*$options = [*

*'cost' => 12, // Ajuste o custo para controlar a força do hash*

*];*

*$senha\_hash = password\_hash($senha, PASSWORD\_DEFAULT, $options);*

*// Inserir no banco de dados*

*INSERT INTO Monitor (nome\_monitor, senha\_hash)*

*VALUES ('João da Silva', '$senha\_hash');*

*// Verificação de senha no PHP*

*// Recuperar o hash armazenado no banco de dados*

*$hash\_armazenado = // ...*

*// Comparar o hash calculado com o hash armazenado*

*if (password\_verify($senha\_informada, $hash\_armazenado)) {*

*// Autenticação bem-sucedida*

*} else {*

*// Autenticação falhou*

*}*

*\*/*

*/\**

*Exemplo de consulta por CEP*

*SELECT \* FROM Usuarios WHERE usuario\_cep = '01001000';*

*Exemplo de consulta por cidade*

*SELECT \* FROM Usuarios WHERE usuario\_cidade = 'São Paulo';*

*\*/*

*-- Verificação de Gestores (Professores responsáveis por Monitores)*

SELECT \* FROM Monitor

WHERE id\_monitor IN (SELECT eh\_gestor FROM Monitor WHERE eh\_gestor IS NOT NULL);

*-- Tintas próximas da validade nos últimos 30 dias*

SELECT \* FROM Tintas

WHERE data\_validade\_tintas BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

*-- STORED PROCEDURE para mistura de tintas*

DELIMITER $$

CREATE PROCEDURE mesclar\_tintas(

IN id\_tinta1 INT,

IN id\_tinta2 INT,

OUT nova\_tinta\_id INT

)

BEGIN

DECLARE nova\_quantidade DECIMAL(5,2); -- Soma das duas quantidades

DECLARE nova\_data\_validade DATE; *-- Considerar a menor data das duas tintas*

DECLARE historico\_mistura VARCHAR(200); *-- Verifica se as tintas existem e não estão marcadas como excluídas*

IF NOT EXISTS (SELECT \* FROM Tintas WHERE id\_tintas = id\_tinta1 AND excluido = 0) OR

NOT EXISTS (SELECT \* FROM Tintas WHERE id\_tintas = id\_tinta2 AND excluido = 0) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Uma ou ambas as tintas não existem ou estão marcadas como excluídas.';

END IF;

*-- Calcula a nova quantidade*

SELECT

t1.quantidade\_tintas\_disponivel + t2.quantidade\_tintas\_disponivel INTO nova\_quantidade

FROM Tintas t1, Tintas t2

WHERE t1.id\_tintas = id\_tinta1 AND t2.id\_tintas = id\_tinta2;

*-- Calcula a menor data de validade*

SELECT LEAST(t1.data\_validade\_tintas, t2.data\_validade\_tintas)

INTO nova\_data\_validade

FROM Tintas t1

JOIN Tintas t2 ON t1.id\_tintas = id\_tinta1 AND t2.id\_tintas = id\_tinta2;

*-- Verifica se a nova quantidade é maior que zero*

IF nova\_quantidade <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'A quantidade da nova tinta deve ser maior que zero.';

END IF;

*-- Constrói o histórico da mistura*

SET historico\_mistura = CONCAT(

'Tinta 1\n',

(SELECT CONCAT('nome: ', nome\_tintas, ', quantidade: ', quantidade\_tintas\_disponivel) FROM Tintas WHERE id\_tintas = id\_tinta1),

'\nTinta 2\n',

(SELECT CONCAT('nome: ', nome\_tintas, ', quantidade: ', quantidade\_tintas\_disponivel) FROM Tintas WHERE id\_tintas = id\_tinta2),

'\nData da Mistura: ', NOW()

);

*-- Insere o novo registro com o nome fornecido pelo usuário e o histórico da mistura*

INSERT INTO Tintas (nome\_tintas, quantidade\_tintas\_disponivel, data\_validade\_tintas, mistura, histórico)

VALUES (

nome\_novo\_tinta,

nova\_quantidade,

nova\_data\_validade,

1,

historico\_mistura

);

SET nova\_tinta\_id = LAST\_INSERT\_ID();

*-- Marca as tintas originais como excluídas*

UPDATE Tintas SET excluido = 1 WHERE id\_tintas IN (id\_tinta1, id\_tinta2);

END $$

DELIMITER ;

*-- FIM STORED PROCEDURE para mistura de tintas*

*/\**

*-- Exemplo de sintaxe do Stored Procedure*

*CALL mesclar\_tintas([#id\_tintas\_1], [#id\_tintas\_2], ['novo nome'], @nova\_tinta\_id);*

*SELECT @nova\_tinta\_id; -- Exibe o ID da nova tinta*

*-- Exemplo de chamada do Stored Procedure*

*CALL mesclar tintas(2, 5, 'Verde limão', @nova\_tinta\_id);*

*SELECT @nova\_tinta\_id;*

*\*/*

*-- STORED PROCEDURE para o TRIGGER de atualiazação de status de solicitação*

DELIMITER $$

CREATE PROCEDURE sp\_enviar\_notificacao\_aprovacao(IN id\_solicitacao INT)

BEGIN

DECLARE beneficiario\_email VARCHAR(100);

SELECT b.email INTO beneficiario\_email

FROM Analise a

INNER JOIN Solicitacao s ON a.id\_solicitacao = s.id\_solicitacao

INNER JOIN Beneficiario b ON s.id\_beneficiario = b.id\_beneficiario

WHERE a.id\_solicitacao = id\_solicitacao;

*-- Construir a mensagem de notificação*

SET @mensagem = CONCAT('Sua solicitação de número ', id\_solicitacao, ' foi aprovada.');

END $$

DELIMITER;

*-- FIM STORED PROCEDURE para o TRIGGER de atualiazação de status de solicitação*

*-- TRIGGER para atualização de status de solicitação*

DELIMITER $$

CREATE TRIGGER tr\_notificar\_aprovacao\_solicitacao

AFTER UPDATE ON Analise

FOR EACH ROW

BEGIN

IF NEW.status\_solicitacao = 'Aprovado' AND OLD.status\_solicitacao != 'Aprovado' THEN

CALL sp\_enviar\_notificacao\_aprovacao(NEW.id\_solicitacao);

END IF;

END $$

DELIMITER ;

*-- FIM TRIGGER para atualização de status de solicitação*

*/\**

*\*Código PHP para integração com PHP*

*CREATE PROCEDURE sp\_enviar\_email(IN email\_destino VARCHAR(100), IN assunto VARCHAR(100), IN corpo VARCHAR(1000))*

*BEGIN*

*-- // Bloco DECLARE exit handler: Garante que qualquer exceção seja capturada e um rollback seja realizado.*

*DECLARE exit handler for sqlexception*

*BEGIN*

*ROLLBACK; -- Logar o erro*

*END;*

*-- Código PHP para enviar o email usando PHPMailer*

*-- // Variável @php\_code: Armazena o código PHP como uma string.*

*SET @php\_code = CONCAT(*

*'use PHPMailer\\PHPMailer\\PHPMailer;',*

*'use PHPMailer\\PHPMailer\\Exception;',*

*'require \'vendor/autoload.php\';',*

*'*

*$mail = new PHPMailer(true);*

*$mail->SMTPDebug = 0; //Enable verbose debug output*

*$mail->isSMTP(); //Send using SMTP*

*$mail->Host = \'smtp.example.com\'; //Set the SMTP server to send through*

*$mail->SMTPAuth = true; //Enable SMTP authentication*

*$mail->Username = \'your\_email@example.com\'; //SMTP username*

*$mail->Password = \'your\_password\'; //SMTP password*

*$mail->SMTPSecure = PHPMailer::ENCRYPTION\_SMTPS; //Enable implicit TLS encryption*

*$mail->Port = 465; //TCP port to connect to*

*$mail->setFrom(\'your\_email@example.com\', \'Seu Nome\');*

*$mail->addAddress(\'', email\_destino, '\', \'Beneficiário\');*

*$mail->isHTML(true); //Set email format to HTML*

*$mail->Subject = assunto;*

*$mail->Body = corpo;*

*$mail->send();*

*',*

*'echo "Mensagem enviada com sucesso!";'*

*);*

*-- // PREPARE, EXECUTE, DEALLOCATE: Comandos SQL para preparar, executar e dealocar uma declaração preparada, permitindo a execução de código PHP dentro do MySQL.*

*PREPARE stmt FROM @php\_code;*

*EXECUTE stmt;*

*DEALLOCATE PREPARE stmt;*

*END;*

*\*/*

*/\**

*\* Código no PHP*

*<?php*

*use PHPMailer\PHPMailer\PHPMailer;*

*use PHPMailer\PHPMailer\Exception;*

*require 'vendor/autoload.php';*

*$mail = new PHPMailer(true);*

*try {*

*//Configurações do email*

*$mail->SMTPDebug = 0; //Enable verbose debug output*

*$mail->isSMTP(); //Send using SMTP*

*$mail->Host = 'smtp.example.com'; //Set the SMTP server to send through*

*$mail->SMTPAuth = true; //Enable SMTP authentication*

*$mail->Username = 'your\_email@example.com'; //SMTP username*

*$mail->Password = 'your\_password'; //SMTP password*

*$mail->SMTPSecure = PHPMailer::ENCRYPTION\_SMTPS; //Enable implicit TLS encryption*

*$mail->Port = 465; //TCP port to connect to*

*$mail->setFrom('your\_email@example.com', 'Seu Nome');*

*$mail->addAddress($email\_beneficiario, 'Beneficiário'); //Conteúdo do email*

*$mail->isHTML(true); //Set email format to HTML*

*$mail->Subject = 'Pedido Aprovado';*

*$mail->Body = $mensagem;*

*$mail->send();*

*echo 'Mensagem enviada com sucesso!';*

*} catch (Exception $e) {*

*echo "Mensagem não pode ser enviada. Mailer Error: {$mail->ErrorInfo}";*

*}*

*\*/*