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

U<u>NIVERSO:</u> 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.

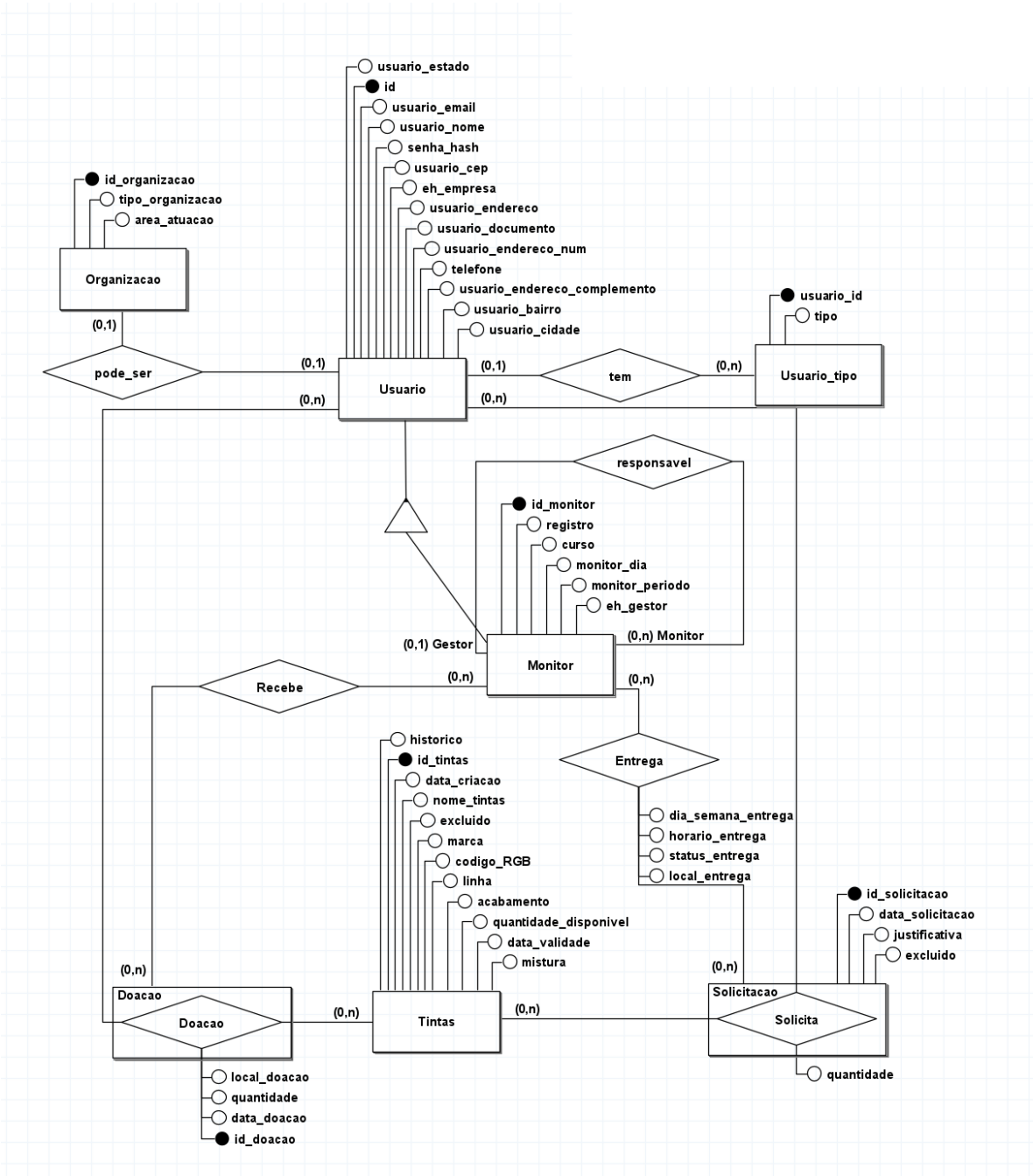
A tabela **Doacao** registra as doações de tintas, relacionando os Doadores, Monitores e as Tintas recebidas. Possui os atributos *id_doacao, id_doador, id_monitor, data_doacao, local_doado* e relaciona-se com as tabelas **Usuario** (*id*) e **Monitor** (*id_monitor*) por meio dos campos *id_doador* 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*).

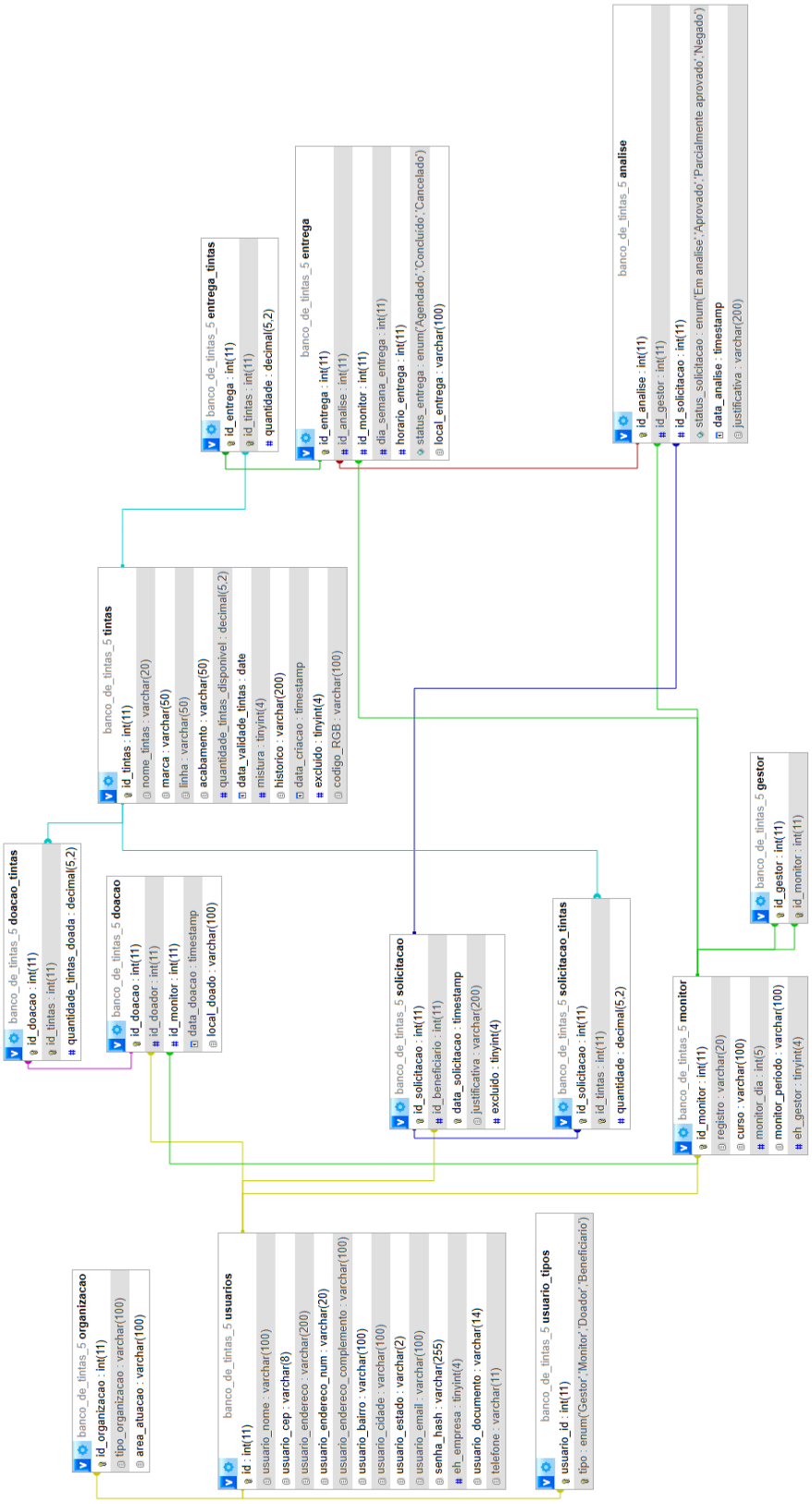
No entanto, o processo de solicitação requer uma **Analise**, para verificação da justificativa da quantidade de tintas solicitadas, visto que uma **Organizacao** também pode fazer um requerimento. A tabela **Analise** armazena os seguintes atributos: *id_analise, id_gestor, id_solicitacao, status_solicitacao, 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



Modelo Entidade Relacionamento – MER



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}";
}
*/
```