SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
SENAI – DENDEZEIROS

DANIEL JORGE SANTOS CORREIA
EVERSON CERQUEIRA DA CRUZ MAIA
ISAIAS ARAÚJO NETO
JONATHAN DE JESUS ANDRADE DOS SANTOS
SAMUEL SOUSA OLIVEIRA DA CRUZ

PROJETO PRÁTICO:

MY EPI

SALVADOR – BA
2024

DANIEL JORGE SANTOS CORREIA
EVERSON CERQUEIRA DA CRUZ MAIA
ISAIAS ARAÚJO NETO
JONATHAN DE JESUS ANDRADE DOS SANTOS
SAMUEL SOUSA OLIVEIRA DA CRUZ

PROJETO PRÁTICO:

MY EPI

Proposta solicitada como requisito de avaliação do 2º semestre, na disciplina de **Banco de dados** do Senai – Dendezeiros.

Orientador: Prof. Esp. Christiane.

SALVADOR – BA
2024

SUMÁRIO

MY EPI

1. CONTEXTUALIZAÇÃO

A ausência de um controle adequado dos estoques de Equipamentos de Proteção Individual (EPIs) pode acarretar uma série de complicações, desde a exposição dos colaboradores a riscos ocupacionais até impactos financeiros substanciais para as empresas.

Este projeto visa abordar essa necessidade, propondo um banco de dados de uma plataforma que não apenas monitora o inventário de EPIs, mas também automatiza o controle de estoque. Além disso, busca-se proporcionar agilidade na distribuição dos EPIs, assegurando que os colaboradores tenham acesso rápido aos equipamentos de proteção adequados.

Outro objetivo fundamental é garantir a segurança das informações, implementando medidas robustas de proteção de dados para resguardar a confidencialidade e integridade das informações relacionadas ao estoque de EPIs.

Ao oferecer uma solução abrangente que inclui controle de estoque automatizado, agilidade na distribuição e segurança das informações, este projeto não apenas simplificará as operações diárias das empresas, mas também contribuirá para um ambiente de trabalho mais seguro e produtivo.

2. MINIMUNDO

- Cada EPI será registrado no sistema
- Nesse registro deve conter ID, Nome, Tipo, Quantidade, Descrição registrado no sistema
- Cada EPI pode ser entregue para nenhum ou vários funcionários
- Cada funcionário deve ser registrado no sistema
- Nesse registro deve conter Nome completo, Número de admissão, Data de nascimento, Data de admissão, CPF, Cargo, Setor, Tipo sanguíneo, telefone e endereço
- Cada funcionário pode pegar nenhum ou vários EPIS
- Cada cargo terá uma quantidade de EPI limitada por seu tipo
- Essa quantidade será escolhida e limitada pelo cliente
- Cada cargo terá acesso a vários tipos de EPI específicos para o seu cargo, determinado pelo cliente
- Ao entregar o EPI será registrado no sistema a entrega
- Nesse registro deve conter ID da entrega, Data e hora, situação da entrega (primeira vez pegando, troca ou devolução), quantidade entregue, ID do funcionário, ID do EPI e descrição da entrega

3. REGRA DE NEGÓCIOS

- A entrega de EPI deve ser registrada no sistema antes de ser efetuada
- Todos os funcionários devem ser cadastrados no sistema para receber EPI
- O acesso total ao sistema deve ser restrito a funcionários autorizados
- O usuário deverá ser capaz de alterar e visualizar as informações de disponibilidade de EPIS
- Os EPIs vencidos não devem ser disponibilizados para uso e devem ser descartados de acordo com as regulamentações

4. REQUISITOS NÃO FUNCIONAIS

DESEMPENHO: O sistema deve processar e responder a consultas de estoque em no máximo 2 segundos.

DISPONIBILIDADE: O sistema deve estar disponível 99,9% do tempo.

USABILIDADE: A interface deve ser intuitiva e fácil de usar, permitindo que novos usuários se familiarizem com o sistema em até 2 horas de treinamento.

PORTABILIDADE: O sistema deve ser compatível com os principais sistemas operacionais (Windows, Linux, MacOS) e navegadores web (Chrome, Firefox, Safari).

5. REQUISITOS FUNCIONAIS

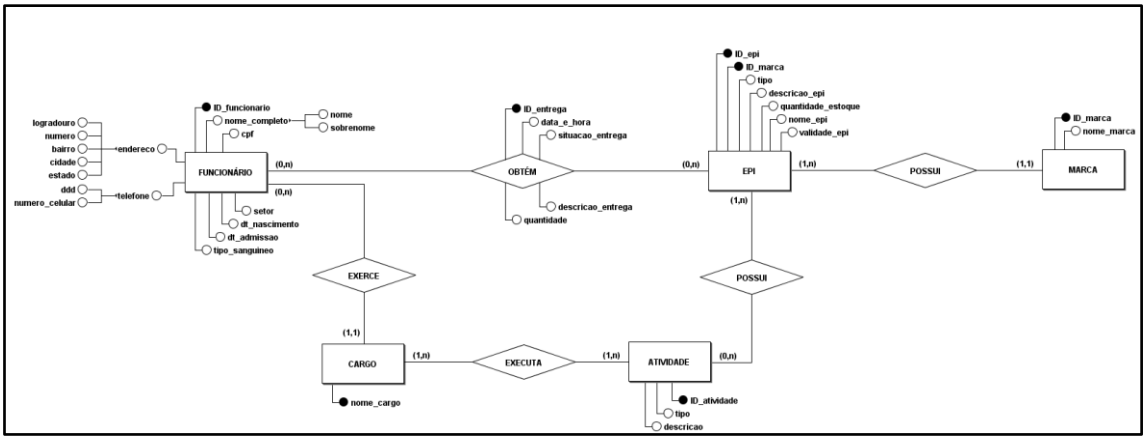
- O usuário deve acessar o sistema através de um login com usuário e senha
- O sistema deve possibilitar a Inclusão de cadastro/registo de informações do EPI através do leitor de QR code
- Deve ser capaz de realizar a alteração e visualização de informações
- Deve controlar o estoque atual de cada tipo de EPI, atualizando automaticamente com as entradas e saídas registradas

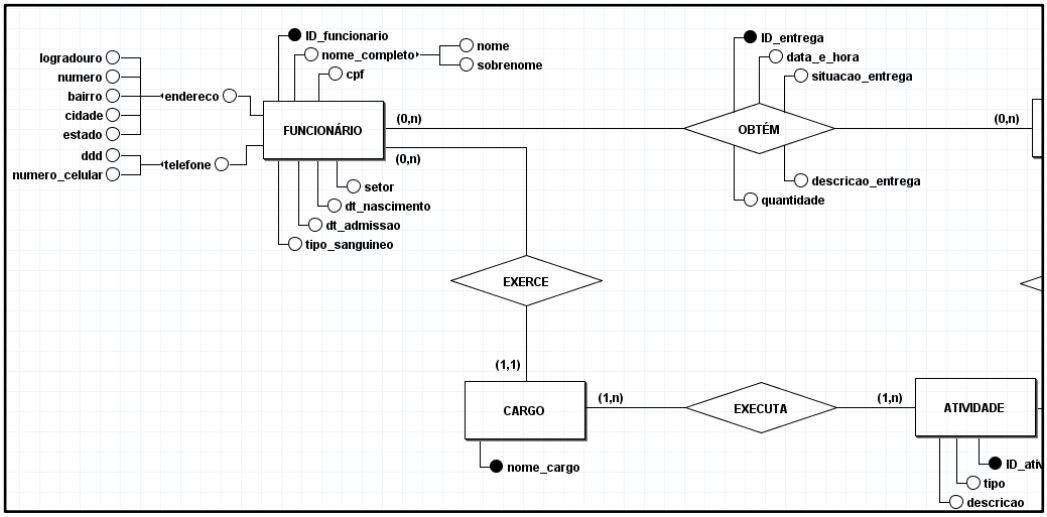
6. REQUISITOS NORMATIVOS

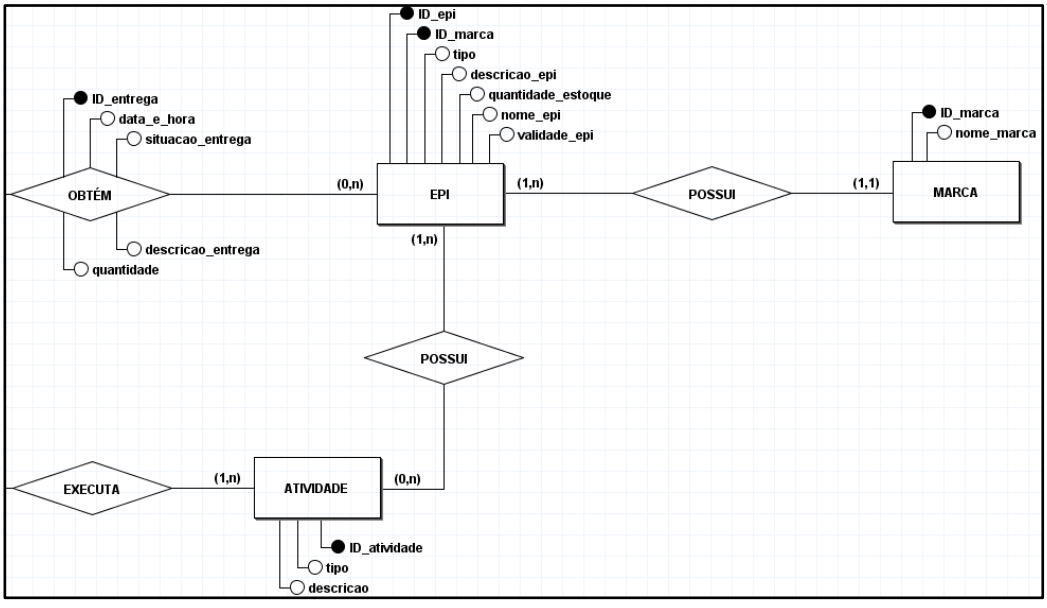
- O sistema deve garantir o cumprimento das normas de segurança do trabalho da legislação vigente para a indústria da construção civil
- Cumprir os regulamentos de proteção de dados e garantir a segurança das informações dos funcionários
- Cumprir as normas de segurança no trabalho e garantir equipamento de proteção individual (EPI) adequado

7. MODELAGEM DO BANCO DE DADOS

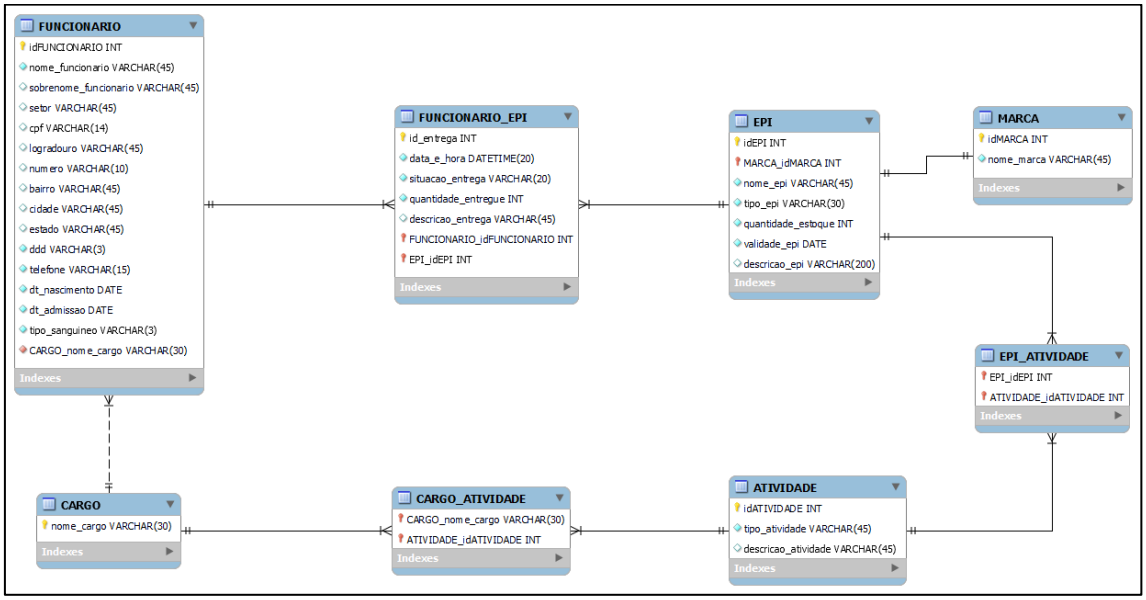
7.1. MODELO CONCEITUAL

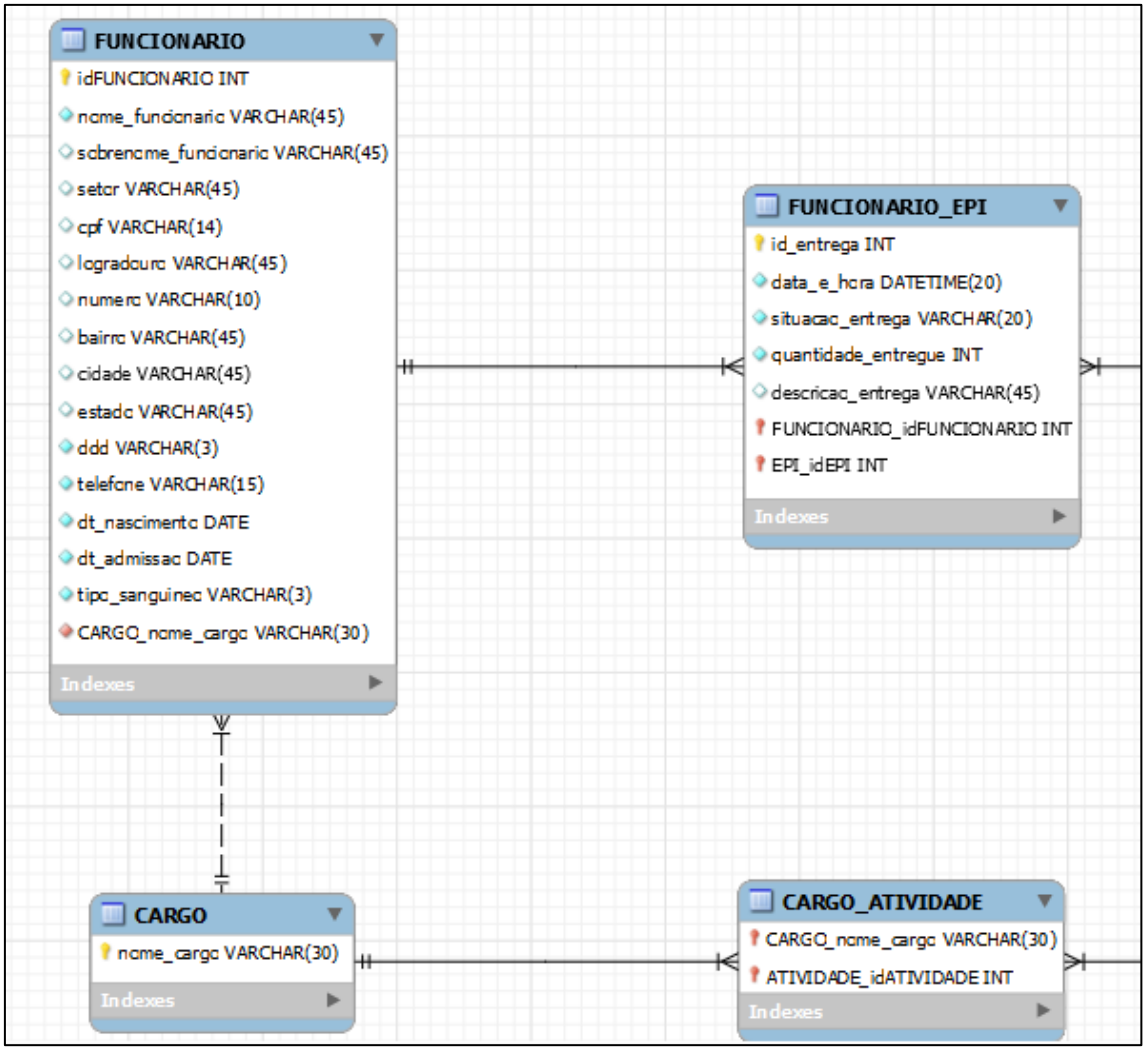


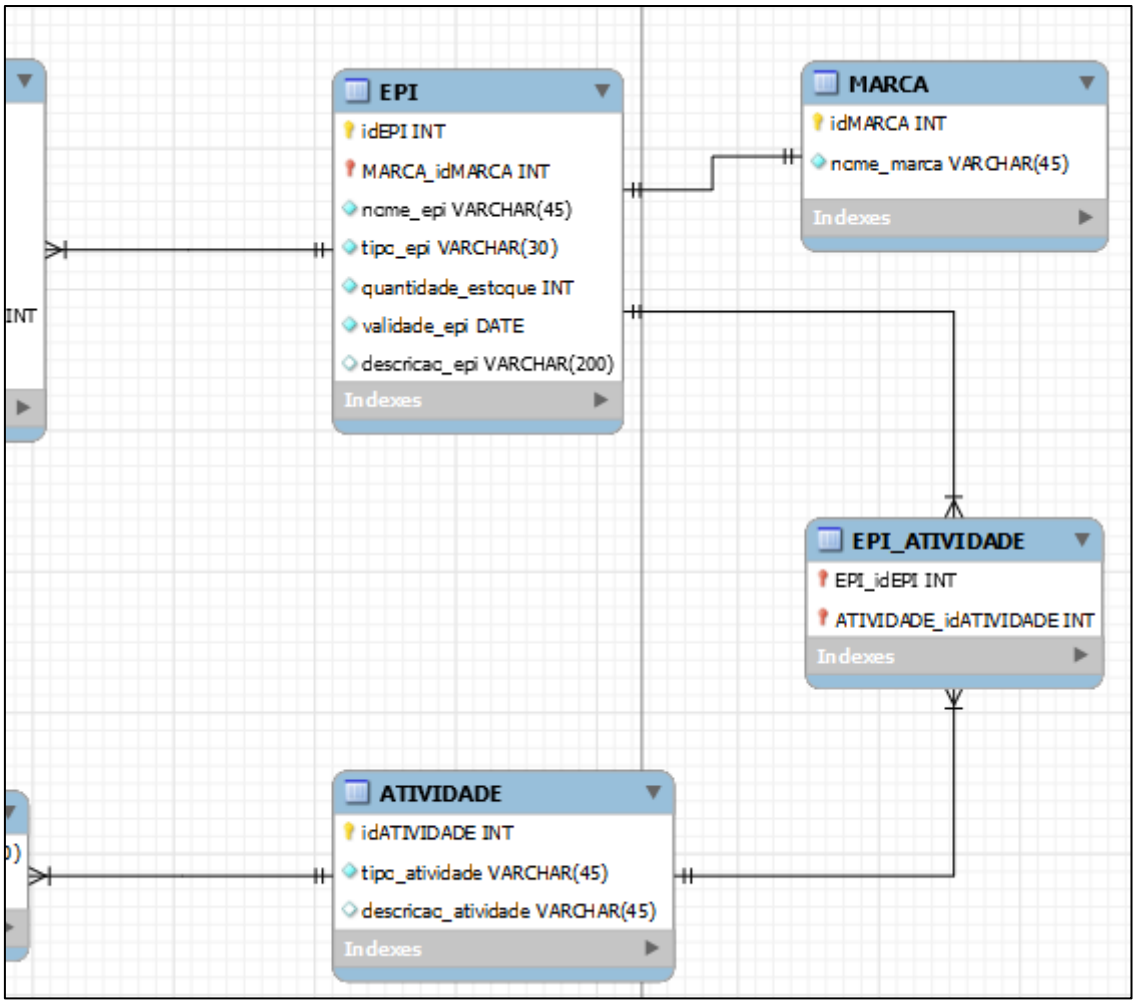




7.2. MODELO LÓGICO






EPI
idEPI INT
MARCA_idMARCA INT
nome_epi VARCHAR(45)
tipo_epi VARCHAR(30)
quantidade_estoque INT
validade_epi DATE
descricao_epi VARCHAR(200)
Indexes
MARCA
idMARCA INT
nome_marca VARCHAR(45)
Indexes
EPI_ATIVIDADE
EPI_idEPI INT
ATIVIDADE_idATIVIDADE INT
Indexes
ATIVIDADE
idATIVIDADE INT
tipo_atividade VARCHAR(45)
descricao_atividade VARCHAR(45)
Indexes

7.3. MODELO FÍSICO

7.3.1. Script de criação do banco

```
-- -----------------------------------------------------
-- Schema db_my_epi
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `db_my_epi` DEFAULT CHARACTER SET utf8 ;
USE `db_my_epi` ;
-- -----------------------------------------------------
-- Table `db_my_epi`.`CARGO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`CARGO` (
  `nome_cargo` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`nome_cargo`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `db_my_epi`.`FUNCIONARIO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`FUNCIONARIO` (
  `idFUNCIONARIO` INT NOT NULL AUTO_INCREMENT,
  `nome_funcionario` VARCHAR(45) NOT NULL,
  `sobrenome_funcionario` VARCHAR(45) NULL,
  `setor` VARCHAR(45) NULL,
  `cpf` VARCHAR(14) NULL,
  `logradouro` VARCHAR(45) NULL,
  `numero` VARCHAR(10) NULL,
  `bairro` VARCHAR(45) NULL,
  `cidade` VARCHAR(45) NULL,
  `estado` VARCHAR(45) NULL,
  `ddd` VARCHAR(3) NOT NULL,
  `telefone` VARCHAR(15) NOT NULL,
  `dt_nascimento` DATE NOT NULL,
  `dt_admissao` DATE NOT NULL,
  `tipo_sanguineo` VARCHAR(3) NOT NULL,
```

```
  `CARGO_nome_cargo` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`idFUNCIONARIO`),
  INDEX `fk_FUNCIONARIO_CARGO1_idx` (`CARGO_nome_cargo` ASC) ,
  CONSTRAINT `fk_FUNCIONARIO_CARGO1`
    FOREIGN KEY (`CARGO_nome_cargo`)
    REFERENCES `db_my_epi`.`CARGO` (`nome_cargo`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`MARCA`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`MARCA` (
  `idMARCA` INT NOT NULL AUTO_INCREMENT,
  `nome_marca` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMARCA`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`EPI`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`EPI` (
  `idEPI` INT NOT NULL AUTO_INCREMENT,
  `MARCA_idMARCA` INT NOT NULL,
  `nome_epi` VARCHAR(45) NOT NULL,
  `tipo_epi` VARCHAR(30) NOT NULL,
  `quantidade_estoque` INT NOT NULL,
  `validade_epi` DATE NOT NULL,
  `descricao_epi` VARCHAR(200) NULL,
  PRIMARY KEY (`idEPI`, `MARCA_idMARCA`),
  INDEX `fk_EPI_MARCA1_idx` (`MARCA_idMARCA` ASC),
  CONSTRAINT `fk_EPI_MARCA1`
    FOREIGN KEY (`MARCA_idMARCA`)
    REFERENCES `db_my_epi`.`MARCA` (`idMARCA`)
    ON DELETE RESTRICT
```

```
  ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`ATIVIDADE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`ATIVIDADE` (
  `idATIVIDADE` INT NOT NULL AUTO_INCREMENT,
  `tipo_atividade` VARCHAR(45) NOT NULL,
  `descricao_atividade` VARCHAR(45) NULL,
  PRIMARY KEY (`idATIVIDADE`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`FUNCIONARIO_EPI`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`FUNCIONARIO_EPI` (
  `id_entrega` INT NOT NULL AUTO_INCREMENT,
  `data_e_hora` DATETIME NOT NULL,
  `situacao_entrega` VARCHAR(20) NOT NULL,
  `quantidade_entregue` INT NOT NULL,
  `descricao_entrega` VARCHAR(45) NULL,
  `FUNCIONARIO_idFUNCIONARIO` INT NOT NULL,
  `EPI_idEPI` INT NOT NULL,
  PRIMARY KEY (`id_entrega`, `FUNCIONARIO_idFUNCIONARIO`, `EPI_idEPI`),
  INDEX `fk_FUNCIONARIO_has_EPI_EPI1_idx` (`EPI_idEPI` ASC) ,
  INDEX `fk_FUNCIONARIO_has_EPI_FUNCIONARIO_idx` (`FUNCIONARIO_idFUNCIONARIO` ASC) ,
  CONSTRAINT `fk_FUNCIONARIO_has_EPI_FUNCIONARIO`
    FOREIGN KEY (`FUNCIONARIO_idFUNCIONARIO`)
    REFERENCES `db_my_epi`.`FUNCIONARIO` (`idFUNCIONARIO`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_FUNCIONARIO_has_EPI_EPI1`
    FOREIGN KEY (`EPI_idEPI`)
    REFERENCES `db_my_epi`.`EPI` (`idEPI`)
    ON DELETE RESTRICT
```

```
  ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`CARGO_ATIVIDADE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`CARGO_ATIVIDADE` (
 `CARGO_nome_cargo` VARCHAR(30) NOT NULL,
 `ATIVIDADE_idATIVIDADE` INT NOT NULL,
 PRIMARY KEY (`CARGO_nome_cargo`, `ATIVIDADE_idATIVIDADE`),
 INDEX `fk_CARGO_has_ATIVIDADE_ATIVIDADE1_idx` (`ATIVIDADE_idATIVIDADE` ASC) ,
 INDEX `fk_CARGO_has_ATIVIDADE_CARGO1_idx` (`CARGO_nome_cargo` ASC) ,
 CONSTRAINT `fk_CARGO_has_ATIVIDADE_CARGO1`
  FOREIGN KEY (`CARGO_nome_cargo`)
  REFERENCES `db_my_epi`.`CARGO` (`nome_cargo`)
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
 CONSTRAINT `fk_CARGO_has_ATIVIDADE_ATIVIDADE1`
  FOREIGN KEY (`ATIVIDADE_idATIVIDADE`)
  REFERENCES `db_my_epi`.`ATIVIDADE` (`idATIVIDADE`)
  ON DELETE RESTRICT
  ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `db_my_epi`.`EPI_ATIVIDADE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_my_epi`.`EPI_ATIVIDADE` (
 `EPI_idEPI` INT NOT NULL,
 `ATIVIDADE_idATIVIDADE` INT NOT NULL,
 PRIMARY KEY (`EPI_idEPI`, `ATIVIDADE_idATIVIDADE`),
 INDEX `fk_EPI_has_ATIVIDADE_ATIVIDADE1_idx` (`ATIVIDADE_idATIVIDADE` ASC) ,
 INDEX `fk_EPI_has_ATIVIDADE_EPI1_idx` (`EPI_idEPI` ASC) ,
 CONSTRAINT `fk_EPI_has_ATIVIDADE_EPI1`
  FOREIGN KEY (`EPI_idEPI`)
  REFERENCES `db_my_epi`.`EPI` (`idEPI`)
```

```
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_EPI_has_ATIVIDADE_ATIVIDADE1`
    FOREIGN KEY (`ATIVIDADE_idATIVIDADE`)
    REFERENCES `db_my_epi`.`ATIVIDADE` (`idATIVIDADE`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

7.3.2. Script de manipulação do banco

```
-- CARGO

INSERT INTO cargo (nome_cargo)
VALUES
   ('Ajudante'),
   ('Almoxerife'),
   ('Arquiteto'),
   ('Carpinteiro'),
   ('Eletricista'),
   ('Encanador'),
   ('Engenheiro'),
   ('Faxineiro'),
   ('Pedreiro'),
   ('Pintor'),
   ('Técnica de Segurança');

-- FUNCIONARIO

INSERT INTO funcionario (
   idFUNCIONARIO,
   nome_funcionario,
   sobrenome_funcionario,
   setor,
   cpf,
   logradouro,
   numero,
   bairro,
   cidade,
   estado,
   ddd,
   telefone,
   dt_nascimento,
   dt_admissao,
   tipo_sanguineo,
   CARGO_nome_cargo
) VALUES
   (1, 'João', 'Silva', 'Construção', '123.456.789-00', 'Rua das Flores', '100', 'Centro', 'São Paulo', 'São Paulo', '11', '987654321', '1980-01-15', '2010-05-20', 'O+', 'Ajudante'),
   (2, 'Maria', 'Santos', 'Logística', '987.654.321-00', 'Avenida Brasil', '200', 'Jardim', 'Rio de Janeiro', 'Rio de Janeiro', '21', '987654322', '1985-02-25', '2012-03-15', 'A-', 'Almoxerife'),
   (3, 'Carlos', 'Oliveira', 'Planejamento', '456.789.123-00', 'Rua das Palmeiras', '300', 'Copacabana', 'Rio de Janeiro', 'Rio de Janeiro', '21', '987654323', '1978-03-10', '2008-07-30', 'B+', 'Arquiteto'),
   (4, 'Ana', 'Pereira', 'Construção', '789.123.456-00', 'Rua da Paz', '400', 'Centro', 'São Paulo', 'São Paulo', '11', '987654324', '1990-04-20', '2015-06-10', 'O-', 'Carpinteiro'),
   (5, 'Pedro', 'Almeida', 'Manutenção', '321.654.987-00', 'Avenida Paulista', '500', 'Bela Vista', 'São Paulo', 'São Paulo', '11', '987654325', '1982-05-05', '2011-09-01', 'AB+', 'Eletricista'),
   (6, 'Fernanda', 'Costa', 'Manutenção', '654.987.321-00', 'Rua das Acácias', '600', 'Botafogo', 'Rio de Janeiro', 'Rio de Janeiro', '21', '987654326', '1987-06-15', '2013-11-20', 'B-', 'Encanador'),
   (7, 'Rafael', 'Rodrigues', 'Engenharia', '147.258.369-00', 'Avenida das Nações', '700', 'Barra', 'Salvador', 'Bahia', '71', '987654327', '1975-07-30', '2005-10-25', 'A+', 'Engenheiro'),
   (8, 'Juliana', 'Lima', 'Limpeza', '258.369.147-00', 'Rua das Magnólias', '800', 'Graça', 'Salvador', 'Bahia', '71', '987654328', '1995-08-10', '2017-02-14', 'O-', 'Faxineiro'),
   (9, 'Bruno', 'Martins', 'Construção', '369.147.258-00', 'Avenida Central', '900', 'Centro', 'Belo Horizonte', 'Minas Gerais', '31', '987654329', '1983-09-25', '2009-01-05', 'AB-', 'Pedreiro'),
   (10, 'Patrícia', 'Barbosa', 'Pintura', '159.357.486-00', 'Rua das Oliveiras', '1000', 'Savassi', 'Belo Horizonte', 'Minas Gerais', '31', '987654330', '1992-10-15', '2016-04-18', 'A+', 'Pintor'),
```

```
(11, 'Lucas', 'Souza', 'Segurança', '753.951.258-00', 'Avenida das Américas', '1100', 'Recreio', 'Rio de Janeiro', 'Rio de Janeiro', '21', '987654331', '1988-11-05', '2014-08-08', 'O+', 'Técnica de Segurança');

-- ATIVIDADE

INSERT INTO atividade (tipo_atividade, descricao_atividade) VALUES
('Instalação Elétrica', 'Instalar e manter sistemas elétricos, realizar reparos e garantir a segurança'),
('Manutenção de Estoque', 'Gerenciar o estoque de materiais e equipamentos, registrar entradas e saídas e garantir a organização'),
('Desenho Arquitetônico', 'Desenvolver projetos arquitetônicos, realizar esboços e acompanhar a execução das obras'),
('Montagem de Estruturas de Madeira', 'Construir, montar e reparar estruturas de madeira, como formas para concretagem e telhados'),
('Instalação Hidráulica', 'Instalar e reparar sistemas de tubulação e encanamentos, garantir a funcionalidade dos sistemas de água e esgoto'),
('Fiscalização de Obras', 'Supervisionar e coordenar as atividades de construção, garantir a conformidade com os projetos e normas de segurança'),
('Limpeza e Organização', 'Manter a limpeza e organização dos ambientes de trabalho, realizar tarefas de limpeza geral'),
('Assentamento de Tijolos', 'Realizar assentamento de tijolos, blocos e outros materiais de construção, garantir o alinhamento e a resistência das paredes'),
('Pintura de Superfícies', 'Pintar e revestir superfícies internas e externas, aplicar técnicas de acabamento e decoração'),
('Segurança do Trabalho', 'Implementar medidas de segurança, realizar treinamentos e inspeções, garantir o cumprimento das normas de segurança'),
('Apoio Geral', 'Auxiliar nas diversas atividades de obra, transportar materiais, realizar pequenas tarefas conforme necessário');

-- CARGO-ATIVIDADE

INSERT INTO cargo_atividade (CARGO_nome_cargo, ATIVIDADE_idATIVIDADE) VALUES
('Eletricista', 1),  -- Instalação Elétrica
('Almoxerife', 2),   -- Manutenção de Estoque
('Arquiteto', 3),    -- Desenho Arquitetônico
('Carpinteiro', 4),  -- Montagem de Estruturas de Madeira
('Encanador', 5),    -- Instalação Hidráulica
('Engenheiro', 6),   -- Fiscalização de Obras
('Faxineiro', 7),    -- Limpeza e Organização
('Pedreiro', 8),     -- Assentamento de Tijolos
('Pintor', 9),       -- Pintura de Superfícies
('Técnica de Segurança', 10), -- Segurança do Trabalho
('Ajudante', 11);    -- Apoio Geral

-- MARCA

INSERT INTO marca (nome_marca) VALUES
  ('SegurançaMáxima'),
  ('ProteçãoTotal'),
  ('DefesaSegura'),
  ('GuardiãoProtetor'),
  ('EscudoSeguro');

-- EPI

INSERT INTO epi (
  idEPI,
  MARCA_idMARCA,
  nome_epi,
  tipo_epi,
```

```
    quantidade_estoque,
    validade_epi,
    descricao_epi
) VALUES
    (1,1, 'Capacete de Segurança', 'Proteção para Cabeça', 23, '2024-12-31', 'Capacete de segurança com ajuste e proteção para impactos.'),
    (2,2, 'Luvas de Proteção', 'Proteção para Mãos', 95, '2025-06-30', 'Luvas de proteção em couro para trabalhos manuais.'),
    (3,3, 'Óculos de Segurança', 'Proteção para Olhos', 80, '2024-11-30', 'Óculos de segurança com proteção lateral e anti-embaçante.'),
    (4,4, 'Protetor Auricular', 'Proteção para Ouvidos', 30, '2025-02-28', 'Protetor auricular para redução de ruídos.'),
    (5,5, 'Máscara Respiratória', 'Proteção para Respiração', 60, '2024-09-30', 'Máscara respiratória com filtro para partículas.'),
    (6,1, 'Cinto de Segurança', 'Proteção para Tronco', 40, '2025-03-31', 'Cinto de segurança para trabalhos em altura.'),
    (7,2, 'Botas de Segurança', 'Proteção para Pés', 70, '2024-10-31', 'Botas de segurança com biqueira de aço e antiderrapantes.'),
    (8,3, 'Protetor Facial', 'Proteção para Rosto', 20, '2025-04-30', 'Protetor facial transparente para proteção contra respingos e partículas.'),
    (9,4, 'Colete Refletivo', 'Proteção para Torso', 25, '2024-08-31', 'Colete refletivo para visibilidade em ambientes de baixa luminosidade.'),
    (10,5, 'Mangote de Proteção', 'Proteção para Braços', 35, '2025-01-31', 'Mangote de proteção em material resistente para proteção dos braços.'),
    (11,1, 'Calça de Segurança', 'Proteção para Pernas', 45, '2024-07-31', 'Calça de segurança com refletivos e resistente a rasgos.'),
    (12,2, 'Camisa de Segurança', 'Proteção para Tronco', 55, '2025-05-31', 'Camisa de segurança com faixas refletivas e ventilação.'),
    (13,3, 'Protetor Solar', 'Proteção para Pele', 90, '2024-06-30', 'Protetor solar com fator de proteção elevado para exposição ao sol.'),
    (14,4, 'Capa de Chuva', 'Proteção para Corpo', 75, '2025-07-31', 'Capa de chuva resistente e impermeável para dias chuvosos.'),
    (15,5, 'Bota de Borracha', 'Proteção para Pés', 65, '2024-05-31', 'Bota de borracha resistente para proteção contra umidade e líquidos.'),
    (16,1, 'Avental de Segurança', 'Proteção para Corpo', 85, '2025-08-31', 'Avental de segurança em material impermeável para proteção do tronco.'),
    (17,2, 'Guarda-chuva', 'Proteção para Corpo', 40, '2024-04-30', 'Guarda-chuva resistente e de fácil manuseio para proteção contra chuvas.'),
    (18,3, 'Fita Antiderrapante', 'Proteção para Ambiente', 30, '2025-09-30', 'Fita antiderrapante para aplicação em superfícies escorregadias.'),
    (19,4, 'Extintor de Incêndio', 'Proteção contra Incêndios', 25, '2024-03-31', 'Extintor de incêndio com capacidade para extinguir diferentes tipos de fogo.'),
    (20,5, 'Sinalizador de Emergência', 'Proteção para Ambiente', 15, '2025-10-31', 'Sinalizador luminoso para indicação de rotas de fuga e emergência.');

-- EPI-ATIVIDADE

INSERT INTO epi_atividade (EPI_idEPI, ATIVIDADE_idATIVIDADE) VALUES
(1, 1),  -- Capacete de Segurança - Instalação Elétrica
(2, 2),  -- Luvas de Proteção - Manutenção de Estoque
(3, 3),  -- Óculos de Segurança - Desenho Arquitetônico
(4, 4),  -- Protetor Auricular - Montagem de Estruturas de Madeira
(5, 5),  -- Máscara Respiratória - Instalação Hidráulica
(6, 6),  -- Cinto de Segurança - Fiscalização de Obras
(7, 7),  -- Botas de Segurança - Limpeza e Organização
(8, 8),  -- Protetor Facial - Assentamento de Tijolos
(9, 9),  -- Colete Refletivo - Pintura de Superfícies
(10, 10), -- Mangote de Proteção - Segurança do Trabalho
(11, 11), -- Calça de Segurança - Apoio Geral
```

(12, 1), -- Camisa de Segurança - Instalação Elétrica
(13, 2), -- Protetor Solar - Manutenção de Estoque
(14, 3), -- Capa de Chuva - Desenho Arquitetônico
(15, 4), -- Bota de Borracha - Montagem de Estruturas de Madeira
(16, 5), -- Avental de Segurança - Instalação Hidráulica
(17, 6), -- Guarda-chuva - Fiscalização de Obras
(18, 7), -- Fita Antiderrapante - Limpeza e Organização
(19, 8), -- Extintor de Incêndio - Assentamento de Tijolos
(20, 9), -- Sinalizador de Emergência - Pintura de Superfícies
(1, 10), -- Capacete de Segurança - Segurança do Trabalho
(2, 11), -- Luvas de Proteção - Apoio Geral
(3, 1), -- Óculos de Segurança - Instalação Elétrica
(4, 2), -- Protetor Auricular - Manutenção de Estoque
(5, 3), -- Máscara Respiratória - Desenho Arquitetônico
(6, 4), -- Cinto de Segurança - Montagem de Estruturas de Madeira
(7, 5), -- Botas de Segurança - Instalação Hidráulica
(8, 6), -- Protetor Facial - Fiscalização de Obras
(9, 7), -- Colete Refletivo - Limpeza e Organização
(10, 8), -- Mangote de Proteção - Assentamento de Tijolos
(11, 9), -- Calça de Segurança - Pintura de Superfícies
(12, 10), -- Camisa de Segurança - Segurança do Trabalho
(13, 11), -- Protetor Solar - Apoio Geral
(14, 1), -- Capa de Chuva - Instalação Elétrica
(15, 2), -- Bota de Borracha - Manutenção de Estoque
(16, 3), -- Avental de Segurança - Desenho Arquitetônico
(17, 4), -- Guarda-chuva - Montagem de Estruturas de Madeira
(18, 5), -- Fita Antiderrapante - Instalação Hidráulica
(19, 6), -- Extintor de Incêndio - Fiscalização de Obras
(20, 7), -- Sinalizador de Emergência - Limpeza e Organização
(1, 8), -- Capacete de Segurança - Assentamento de Tijolos
(2, 9), -- Luvas de Proteção - Pintura de Superfícies
(3, 10), -- Óculos de Segurança - Segurança do Trabalho
(4, 11); -- Protetor Auricular - Apoio Geral

7.3.3. SCRIPT DAS VIEWS E ROTINAS

```
-- VIEW DO ESTOQUE DE EPI

CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `estoque_epi` AS
   SELECT
      `e`.`idEPI` AS `idEPI`,
      `e`.`nome_epi` AS `nome_epi`,
      `e`.`tipo_epi` AS `tipo_epi`,
      `e`.`quantidade_estoque` AS `quantidade_estoque`,
      `e`.`descricao_epi` AS `descricao_epi`
   FROM
      `epi` `e`
   ORDER BY `e`.`idEPI`

-- VIEW DO HISTÓRICO DE ENTREGAS

DELIMITER //
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `historico_entregas` AS
   SELECT
      `f`.`idFUNCIONARIO` AS `idFUNCIONARIO`,
      `f`.`nome_funcionario` AS `nome_funcionario`,
      `e`.`nome_epi` AS `nome_epi`,
      `e`.`idEPI` AS `idEPI`,
      `fe`.`data_e_hora` AS `data_e_hora`,
      `fe`.`situacao_entrega` AS `situacao_entrega`,
      `fe`.`quantidade_entregue` AS `quantidade_entregue`,
      `fe`.`descricao_entrega` AS `descricao_entrega`
   FROM
      ((`funcionario_epi` `fe`
      JOIN `funcionario` `f` ON (`fe`.`FUNCIONARIO_idFUNCIONARIO` = `f`.`idFUNCIONARIO`))
      JOIN `epi` `e` ON (`fe`.`EPI_idEPI` = `e`.`idEPI`))
   ORDER BY `fe`.`data_e_hora` DESC;//
DELIMITER ;

-- VIEW DOS EPIS POR CARGO

DELIMITER //

CREATE VIEW epis_por_cargo AS
SELECT
   c.nome_cargo,
   e.nome_epi
FROM
   cargo c
JOIN
   cargo_atividade ca ON c.nome_cargo = ca.CARGO_nome_cargo
JOIN
   epi_atividade ea ON ca.ATIVIDADE_idATIVIDADE = ea.ATIVIDADE_idATIVIDADE
```

```
JOIN
  epi e ON ea.EPI_idEPI = e.idEPI
ORDER BY
  c.nome_cargo, e.nome_epi;

//
DELIMITER ;
-- PROCEDUREQUE CADASTRA EPI

DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `cadastrar_epi`(
  IN p_nome_epi VARCHAR(100),
  IN p_tipo_epi VARCHAR(50),
  IN p_quantidade_estoque INT,
  IN p_descricao_epi TEXT,
  IN p_id_marca INT,  -- Adicionando o parâmetro para o idMARCA
  IN p_validade_epi DATE -- Adicionando o parâmetro para a validade do EPI
)
BEGIN
  -- Inserir um novo EPI na tabela epi
  INSERT INTO epi (nome_epi, tipo_epi, quantidade_estoque, descricao_epi, MARCA_idMARCA, validade_epi)
  VALUES (p_nome_epi, p_tipo_epi, p_quantidade_estoque, p_descricao_epi, p_id_marca, p_validade_epi);

  -- Exibir a tabela epi atualizada
  SELECT * FROM epi;
END
//
DELIMITER ;

-- PROCEDURE QUE ATUALIZA EPI

DELIMITER //

CREATE PROCEDURE atualizar_epi(
  IN p_idEPI INT,
  IN p_nome_epi VARCHAR(100),
  IN p_tipo_epi VARCHAR(50),
  IN p_quantidade_estoque INT,
  IN p_descricao_epi TEXT,
  IN p_id_marca INT,  -- Adicionando o parâmetro para o idMARCA
  IN p_validade_epi DATE -- Adicionando o parâmetro para a validade do EPI
)
BEGIN
  DECLARE v_nome_epi VARCHAR(100);
  DECLARE v_tipo_epi VARCHAR(50);
  DECLARE v_quantidade_estoque INT;
  DECLARE v_descricao_epi TEXT;
  DECLARE v_id_marca INT;  -- Variável para armazenar o idMARCA atual
  DECLARE v_validade_epi DATE; -- Variável para armazenar a validade do EPI atual

  -- Obter os valores atuais do EPI
  SELECT nome_epi, tipo_epi, quantidade_estoque, descricao_epi, MARCA_idMARCA, validade_epi
  INTO v_nome_epi, v_tipo_epi, v_quantidade_estoque, v_descricao_epi, v_id_marca, v_validade_epi
  FROM epi
  WHERE idEPI = p_idEPI;
```

```
    -- Atualizar os valores com base nos parâmetros de entrada
    IF p_nome_epi IS NOT NULL THEN
        SET v_nome_epi = p_nome_epi;
    END IF;
    IF p_tipo_epi IS NOT NULL THEN
        SET v_tipo_epi = p_tipo_epi;
    END IF;
    IF p_quantidade_estoque IS NOT NULL THEN
        SET v_quantidade_estoque = p_quantidade_estoque;
    END IF;
    IF p_descricao_epi IS NOT NULL THEN
        SET v_descricao_epi = p_descricao_epi;
    END IF;
    IF p_id_marca IS NOT NULL THEN
        SET v_id_marca = p_id_marca;
    END IF;
    IF p_validade_epi IS NOT NULL THEN
        SET v_validade_epi = p_validade_epi;
    END IF;

    -- Atualizar o EPI na tabela epi com base no idEPI
    UPDATE epi
    SET
        nome_epi = v_nome_epi,
        tipo_epi = v_tipo_epi,
        quantidade_estoque = v_quantidade_estoque,
        descricao_epi = v_descricao_epi,
        MARCA_idMARCA = v_id_marca,
        validade_epi = v_validade_epi
    WHERE idEPI = p_idEPI;

    -- Exibir a tabela epi atualizada
    SELECT * FROM epi;
END;
//

DELIMITER ;

-- PROCEDURE QUE REGISTRA AS ENTREGAS DE EPIS

DELIMITER //
CREATE PROCEDURE registrar_entrega_epi (
    IN p_idFUNCIONARIO INT,
    IN p_idEPI INT,
    IN p_quantidade_entregue INT,
    IN p_situacao_entrega VARCHAR(20),
    IN p_descricao_entrega VARCHAR(200)
)
BEGIN
    -- Verifica se a quantidade solicitada está disponível no estoque
    DECLARE v_quantidade_estoque INT;
    SELECT quantidade_estoque INTO v_quantidade_estoque
    FROM epi
    WHERE idEPI = p_idEPI;

    IF v_quantidade_estoque < p_quantidade_entregue THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Quantidade insuficiente no estoque';
    ELSE
```

```
        -- Registra a entrega do EPI com data e hora atual
        INSERT INTO funcionario_epi (FUNCIONARIO_idFUNCIONARIO, EPI_idEPI, situacao_entrega, quantidade_entregue, descricao_entrega, data_e_hora)
        VALUES (p_idFUNCIONARIO, p_idEPI, p_situacao_entrega, p_quantidade_entregue, p_descricao_entrega, NOW()); -- ou CURRENT_TIMESTAMP() dependendo do sistema de gerenciamento de banco de dados
    END IF;
END;
//
DELIMITER ;

-- PROCEDURE QUE FILTRA AS ENTREGAS POR FUNCIONÁRIO

DELIMITER //
CREATE PROCEDURE filtrar_entregas_por_funcionario (
  IN p_idFUNCIONARIO INT
)
BEGIN
  SELECT
f.idFUNCIONARIO,
    f.nome_funcionario,
    e.nome_epi,
    e.idEPI,
    fe.data_e_hora,
    fe.situacao_entrega,
    fe.quantidade_entregue,
    fe.descricao_entrega
  FROM
    funcionario_epi fe
  JOIN
    funcionario f ON fe.FUNCIONARIO_idFUNCIONARIO = f.idFUNCIONARIO
  JOIN
    epi e ON fe.EPI_idEPI = e.idEPI
  WHERE
    fe.FUNCIONARIO_idFUNCIONARIO = p_idFUNCIONARIO
  ORDER BY
    fe.data_e_hora DESC;
END //
DELIMITER ;

-- PROCEDURE QUE FILTRA O ESTOQUE DE EPI

DELIMITER //

CREATE DEFINER=`root`@`localhost` PROCEDURE `filtrar_estoque_epi`(
  IN p_idEPI INT
)
BEGIN
  -- Verificar se o ID do EPI foi fornecido
  IF p_idEPI IS NOT NULL THEN
    -- Filtrar pelo ID do EPI
    SELECT idEPI, nome_epi, quantidade_estoque
    FROM epi
    WHERE idEPI = p_idEPI;
  ELSE
    -- Se nenhum ID do EPI foi fornecido, retornar uma mensagem de erro
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Nenhum ID de EPI fornecido.';
  END IF;
END
```

```
//
DELIMITER ;

-- TRIGGER QUE ATUALIZA O ESTOQUE DE EPI

DELIMITER //
CREATE TRIGGER atualizar_estoque_epi
AFTER INSERT ON funcionario_epi
FOR EACH ROW
BEGIN

  -- Atualiza o estoque do EPI

  UPDATE epi
  SET quantidade_estoque = quantidade_estoque - NEW.quantidade_entregue
  WHERE idEPI = NEW.EPI_idEPI;


END //
DELIMITER ;


-- TRIGGER QUE EMITE UM ALERTA AO ATINGIR ESTOQUE MÍNIMO

DELIMITER //

CREATE TRIGGER alerta_estoque_minimo AFTER UPDATE ON epi
FOR EACH ROW
BEGIN
  DECLARE v_quantidade_minima INT;

  -- Defina a quantidade mínima desejada
  SET v_quantidade_minima = 10; -- Você pode ajustar esse valor conforme necessário

  -- Verifique se a nova quantidade de estoque é menor ou igual à quantidade mínima
  IF NEW.quantidade_estoque <= v_quantidade_minima THEN
    -- Emita um alerta usando a função SIGNAL
    SET @alerta_message = CONCAT('Alerta: A quantidade de estoque do EPI ', NEW.nome_epi, '
atingiu a quantidade mínima de ', v_quantidade_minima, '.');
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @alerta_message;
  END IF;
END;
//

DELIMITER ;


-- FUNCTION QUE RETORNA A QUANTIDADE EM ESTOQUE DE UM EPI ESPECÍFICO

DELIMITER //
CREATE DEFINER=`root`@`localhost` FUNCTION `estoque_epi_especifico`(p_idEPI INT
) RETURNS int(11)
BEGIN
  DECLARE v_quantidade_estoque INT;

  SELECT quantidade_estoque INTO v_quantidade_estoque
  FROM epi
  WHERE idEPI = p_idEPI;
```

```
    RETURN v_quantidade_estoque;
END

//
DELIMITER ;
```