

Proyecto 1 - ONG *Cruz Casi Roja*

Administración de Sistemas Gestores de Bases de Datos



Pablo González Troyano

2º ASIR

Diciembre de 2021

Índice

Proyecto 1 - ONG Cruz Casi Roja	1
Administración de Sistemas Gestores de Bases de Datos	1
Índice	2
Análisis de requisitos	3
Diseño conceptual	4
Diseño lógico (Modelo Relacional)	5
Creación de Tablas	6
Datos a insertar	9
Procedimientos/Funciones	10
Procedimiento 1	10
Procedimiento 2	11
Procedimiento 3	11
Procedimiento 4	12
Procedimiento 5	12
Triggers	14
Trigger 1	14
Trigger 2	14
Trigger 3	15
Cursores	16
Cursor 1	16
Cursor 2	16
Cursor 3	19
Eventos	20
Evento 1	20
Evento 2	20
Evento 3	21

Análisis de requisitos

Se propone la creación de una base de datos para una ONG (Organización No Gubernamental). Esta ONG, *Casi Cruz Roja*, se dedica a enviar material y ayuda humanitaria a zonas en conflicto y campos de refugiados.

Para los envíos de material se contempla el envío de equipo médico y alimentos. En tanto al envío de ayuda humanitaria, se envían equipos multidisciplinares de personal médico,.

Esta ONG se mantiene económicamente gracias a las cuotas de sus socios y socias. De estos se desea mantener una serie de datos personales (nombre, apellidos, dirección, formas de contacto, etc). Además, respecto a las cuotas de las donaciones se debe mantener la frecuencia de estas y el importe, así como la cuenta corriente bancaria en la que realizar el cobro. En tanto a los ingresos de las cuotas y con fines estadísticos y de administración, se debe mantener el histórico de cuotas. Una vez formalizada la baja de un socio o socia, debe mantenerse el histórico durante al menos 2 años.

Para la captación de socios y socias se organizan eventos. En estos eventos pueden colaborar entidades colaboradoras (empresas, centros comerciales, ayuntamientos...). Con fines estadísticos se desea conocer cuántos socios se han asociado en cada evento.

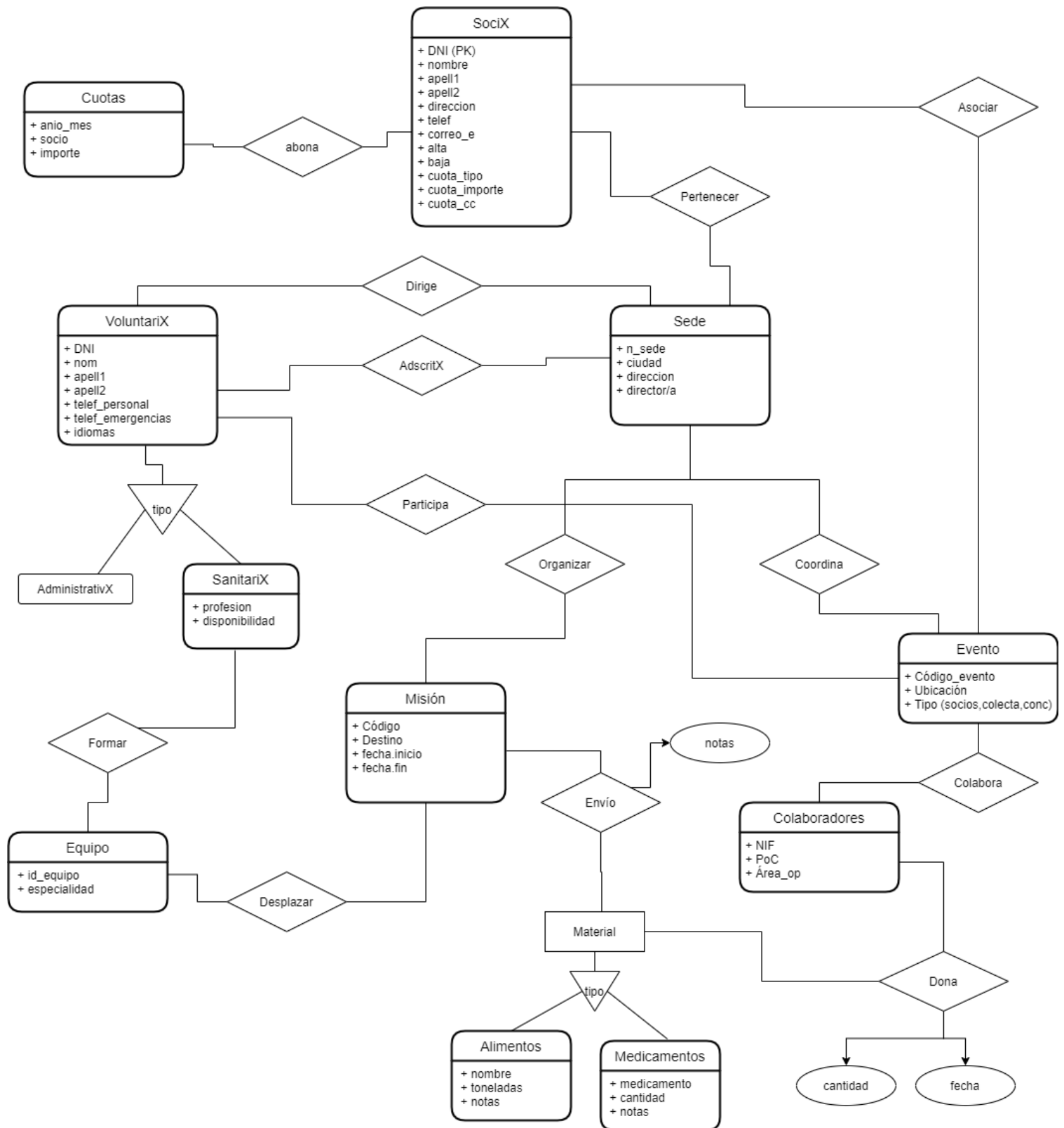
Los eventos los coordinan las distintas sedes repartidas por el territorio nacional. Los socios pertenecen a una sede. Una sede también tendrá un director, que será un voluntario o voluntaria.

En tanto a los voluntarios y voluntarias de la ONG, se desea saber a qué sede están adscritos. Se contemplan dos tipos de voluntarios: sanitarios, de los que se desea mantener su especialidad (psicología, medicina, enfermería, etc) y su disponibilidad en un momento dado (sí o no); y administrativos. De todos los voluntarios y voluntarias, sin importar su tipo, se deben mantener una serie de datos personales y de contacto, como su DNI, nombres, apellidos, teléfono de contacto, idiomas, etc.

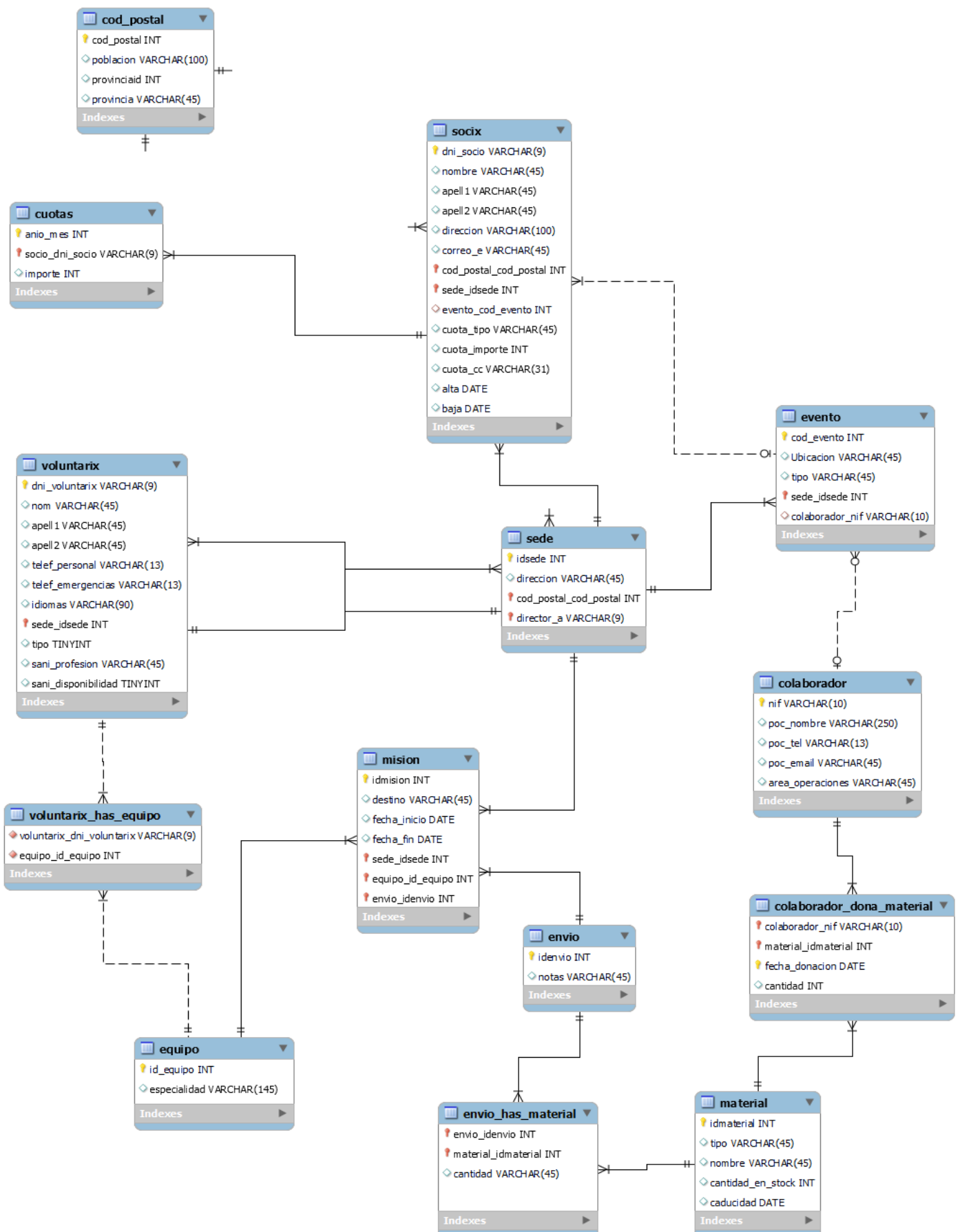
Aunque las misiones son a nivel nacional, por motivos operativos, son coordinadas por una sede. De las misiones se debe guardar el código de la misión, el destino y las fechas de inicio y finalización. Las misiones tienen envíos de medicamentos (de los que se desea conocer el medicamento y la cantidad) y de alimentos (de los que desea conocer el alimento dado, así como la cantidad enviada).

En las misiones también es desplazado un equipo formado por personal sanitario. Por motivos operacionales, se intenta que los equipos sean fijos. Se debe conocer qué sanitarios son enviados a cada misión, así como el histórico de los destinos. Los equipos, aunque pueden ser desplazados a diferentes entornos de misión, estarán especializados en uno: guerras, enfermedades, mujeres, inundaciones, etc

Diseño conceptual



Diseño lógico (Modelo Relacional)



Creación de Tablas

A ejecutar primero la columna de la izquierda.

```
-- MySQL Script generated by MySQL Workbench
-- Sat Dec 11 14:20:56 2021
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-----
-- Schema casicruzroja
-----

DROP SCHEMA IF EXISTS `casicruzroja` ;

-----
-- Schema casicruzroja
-----

CREATE SCHEMA IF NOT EXISTS `casicruzroja` DEFAULT CHARACTER SET utf8 ;
USE `casicruzroja` ;

-----
-- Table `casicruzroja`.`cod_postal`
-----

DROP TABLE IF EXISTS `casicruzroja`.`cod_postal` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`cod_postal` (
  `cod_postal` INT NOT NULL,
  `poblacion` VARCHAR(100) NULL,
  `provinciaid` INT NULL,
  `provincia` VARCHAR(45) NULL,
  PRIMARY KEY (`cod_postal`))
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`voluntarix`
-----

DROP TABLE IF EXISTS `casicruzroja`.`voluntarix` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`voluntarix` (
  `dni_voluntarix` VARCHAR(9) NOT NULL,
  `nom` VARCHAR(45) NULL,
  `apell1` VARCHAR(45) NULL,
  `apell2` VARCHAR(45) NULL,
  `telef_personal` VARCHAR(13) NULL,
  `telef_emergencias` VARCHAR(13) NULL,
  `idiomas` VARCHAR(90) NULL,
  `sede_idsede` INT NOT NULL,
  `tipo` TINYINT NULL,
  `sani_profesion` VARCHAR(45) NULL,
  `sani_disponibilidad` TINYINT NULL,
  PRIMARY KEY (`dni_voluntarix`, `sede_idsede`),
  INDEX `fk_voluntarix_sede1_idx` (`sede_idsede` ASC) VISIBLE,
  CONSTRAINT `fk_voluntarix_sede1`
    FOREIGN KEY (`sede_idsede`)
      REFERENCES `casicruzroja`.`sede` (`idsede`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`sede`
-----

DROP TABLE IF EXISTS `casicruzroja`.`sede` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`sede` (
  `idsede` INT NOT NULL,
  `direccion` VARCHAR(45) NULL,
  `cod_postal_cod_postal` INT NOT NULL,
  `director_a` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`idsede`, `cod_postal_cod_postal`, `director_a`),
  INDEX `fk_sede_cod_postal1_idx` (`cod_postal_cod_postal` ASC) VISIBLE,
  INDEX `fk_sede_voluntarix1_idx` (`director_a` ASC) VISIBLE,

-----
-- Table `casicruzroja`.`cuotas`
-----

DROP TABLE IF EXISTS `casicruzroja`.`cuotas` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`cuotas` (
  `anio_mes` INT NOT NULL,
  `socio_dni_socio` VARCHAR(9) NOT NULL,
  `importe` INT NULL,
  PRIMARY KEY (`anio_mes`, `socio_dni_socio`),
  INDEX `fk_cuotas_socio_idx` (`socio_dni_socio` ASC) VISIBLE,
  CONSTRAINT `fk_cuotas_socio`
    FOREIGN KEY (`socio_dni_socio`)
      REFERENCES `casicruzroja`.`socio` (`dni_socio`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`equipo`
-----

DROP TABLE IF EXISTS `casicruzroja`.`equipo` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`equipo` (
  `id_equipo` INT NOT NULL,
  `especialidad` VARCHAR(145) NULL,
  PRIMARY KEY (`id_equipo`))
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`voluntarix_has_equipo`
-----

DROP TABLE IF EXISTS `casicruzroja`.`voluntarix_has_equipo` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`voluntarix_has_equipo` (
  `voluntarix_dni_voluntarix` VARCHAR(9) NOT NULL,
  `equipo_id_equipo` INT NOT NULL,
  INDEX `fk_voluntarix_has_equipo_equipo1_idx` (`equipo_id_equipo` ASC) VISIBLE,
  INDEX `fk_voluntarix_has_equipo_voluntarix1_idx` (`voluntarix_dni_voluntarix` ASC) VISIBLE,
  CONSTRAINT `fk_voluntarix_has_equipo_voluntarix1`
    FOREIGN KEY (`voluntarix_dni_voluntarix`)
      REFERENCES `casicruzroja`.`voluntarix` (`dni_voluntarix`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_voluntarix_has_equipo_equipo1`
    FOREIGN KEY (`equipo_id_equipo`)
      REFERENCES `casicruzroja`.`equipo` (`id_equipo`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`envio`
-----

DROP TABLE IF EXISTS `casicruzroja`.`envio` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`envio` (
  `idenvio` INT NOT NULL,
  `notas` VARCHAR(45) NULL,
  PRIMARY KEY (`idenvio`))
ENGINE = InnoDB;

-----
-- Table `casicruzroja`.`mision`
-----

DROP TABLE IF EXISTS `casicruzroja`.`mision` ;

CREATE TABLE IF NOT EXISTS `casicruzroja`.`mision` (
  `idmision` INT NOT NULL,
  `destino` VARCHAR(45) NULL,
```

```

CONSTRAINT `fk_sede_cod_postal1`
  FOREIGN KEY (`cod_postal_cod_postal`)
    REFERENCES `casicruzroja`.`cod_postal` (`cod_postal`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_sede_voluntari1`
  FOREIGN KEY (`director_a`)
    REFERENCES `casicruzroja`.`voluntari` (`dni_voluntari`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`colaborador`
-----
DROP TABLE IF EXISTS `casicruzroja`.`colaborador` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`colaborador` (
  `nif` VARCHAR(10) NOT NULL,
  `poc_nombre` VARCHAR(250) NULL,
  `poc_tel` VARCHAR(13) NULL,
  `poc_email` VARCHAR(45) NULL,
  `area_operaciones` VARCHAR(45) NULL,
  PRIMARY KEY (`nif`))
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`evento`
-----
DROP TABLE IF EXISTS `casicruzroja`.`evento` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`evento` (
  `cod_evento` INT NOT NULL,
  `Ubicacion` VARCHAR(45) NULL,
  `tipo` VARCHAR(45) NULL,
  `sede_idsede` INT NOT NULL,
  `colaborador_nif` VARCHAR(10) NULL,
  PRIMARY KEY (`cod_evento`, `sede_idsede`),
  INDEX `fk_evento_sede1_idx` (`sede_idsede` ASC) VISIBLE,
  INDEX `fk_evento_colaborador1_idx` (`colaborador_nif` ASC) VISIBLE,
  CONSTRAINT `fk_evento_sede1`
    FOREIGN KEY (`sede_idsede`)
      REFERENCES `casicruzroja`.`sede` (`idsede`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_evento_colaborador1`
    FOREIGN KEY (`colaborador_nif`)
      REFERENCES `casicruzroja`.`colaborador` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`socix`
-----
DROP TABLE IF EXISTS `casicruzroja`.`socix` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`socix` (
  `dni_socio` VARCHAR(9) NOT NULL,
  `nombre` VARCHAR(45) NULL,
  `apelli1` VARCHAR(45) NULL,
  `apelli2` VARCHAR(45) NULL,
  `direccion` VARCHAR(100) NULL,
  `correo_e` VARCHAR(45) NULL,
  `cod_postal_cod_postal` INT NOT NULL,
  `sede_idsede` INT NOT NULL,
  `evento_cod_evento` INT NULL,
  `cuota_tipo` VARCHAR(45) NULL,
  `cuota_importe` INT NULL,
  `cuota_cc` VARCHAR(31) NULL,
  `alta` DATE NULL,
  `baja` DATE NULL,
  PRIMARY KEY (`dni_socio`, `cod_postal_cod_postal`, `sede_idsede`),
  INDEX `fk_socio_cod_postal1_idx` (`cod_postal_cod_postal` ASC) VISIBLE,
  INDEX `fk_socio_sede1_idx` (`sede_idsede` ASC) VISIBLE,
  INDEX `fk_socix_evento1_idx` (`evento_cod_evento` ASC) VISIBLE,
  CONSTRAINT `fk_socio_cod_postal1`
    FOREIGN KEY (`cod_postal_cod_postal`)
      REFERENCES `casicruzroja`.`cod_postal` (`cod_postal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_socio_sede1`

```

```

  `fecha_inicio` DATE NULL,
  `fecha_fin` DATE NULL,
  `sede_idsede` INT NOT NULL,
  `equipo_id_equipo` INT NOT NULL,
  `envio_idenvio` INT NOT NULL,
  PRIMARY KEY (`idmision`, `sede_idsede`, `equipo_id_equipo`,
`envio_idenvio`),
  INDEX `fk_mision_sede1_idx` (`sede_idsede` ASC) VISIBLE,
  INDEX `fk_mision_equipo1_idx` (`equipo_id_equipo` ASC) VISIBLE,
  INDEX `fk_mision_envio1_idx` (`envio_idenvio` ASC) VISIBLE,
  CONSTRAINT `fk_mision_sede1`
    FOREIGN KEY (`sede_idsede`)
      REFERENCES `casicruzroja`.`sede` (`idsede`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_mision_equipo1`
    FOREIGN KEY (`equipo_id_equipo`)
      REFERENCES `casicruzroja`.`equipo` (`id_equipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_mision_envio1`
    FOREIGN KEY (`envio_idenvio`)
      REFERENCES `casicruzroja`.`envio` (`idenvio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`material`
-----
DROP TABLE IF EXISTS `casicruzroja`.`material` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`material` (
  `idmaterial` INT NOT NULL,
  `tipo` VARCHAR(45) NULL,
  `nombre` VARCHAR(45) NULL,
  `cantidad_en_stock` INT NULL,
  `caducidad` DATE NULL,
  PRIMARY KEY (`idmaterial`))
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`envio_has_material`
-----
DROP TABLE IF EXISTS `casicruzroja`.`envio_has_material` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`envio_has_material` (
  `envio_idenvio` INT NOT NULL,
  `material_idmaterial` INT NOT NULL,
  `cantidad` VARCHAR(45) NULL,
  PRIMARY KEY (`envio_idenvio`, `material_idmaterial`),
  INDEX `fk_envio_has_material_material1_idx` (`material_idmaterial` ASC)
  INVISIBLE,
  INDEX `fk_envio_has_material_envio1_idx` (`envio_idenvio` ASC)
  INVISIBLE,
  CONSTRAINT `fk_envio_has_material_envio1`
    FOREIGN KEY (`envio_idenvio`)
      REFERENCES `casicruzroja`.`envio` (`idenvio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_envio_has_material_material1`
    FOREIGN KEY (`material_idmaterial`)
      REFERENCES `casicruzroja`.`material` (`idmaterial`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `casicruzroja`.`colaborador_dona_material`
-----
DROP TABLE IF EXISTS `casicruzroja`.`colaborador_dona_material` ;

```

```

CREATE TABLE IF NOT EXISTS `casicruzroja`.`colaborador_dona_material` (
  `colaborador_nif` VARCHAR(10) NOT NULL,
  `material_idmaterial` INT NOT NULL,
  `fecha_donacion` DATE NOT NULL,
  `cantidad` INT NULL,
  PRIMARY KEY (`colaborador_nif`, `material_idmaterial`,
`fecha_donacion`),
  INDEX `fk_colaborador_has_material_material1_idx`
  (`material_idmaterial` ASC) VISIBLE,
  INDEX `fk_colaborador_has_material_colaborador1_idx` (`colaborador_nif`

```

```

FOREIGN KEY (`sede_idsede`)
REFERENCES `casicruzroja`.`sede` (`idsede`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_socix_evento1`
FOREIGN KEY (`evento_cod_evento`)
REFERENCES `casicruzroja`.`evento` (`cod_evento`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

ASC) VISIBLE,
CONSTRAINT `fk_colaborador_has_material_colaborador1`
FOREIGN KEY (`colaborador_nif`)
REFERENCES `casicruzroja`.`colaborador` (`nif`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_colaborador_has_material_material1`
FOREIGN KEY (`material_idmaterial`)
REFERENCES `casicruzroja`.`material` (`idmaterial`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

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

```

Las tablas se crearán utilizando el código anteriormente mostrado y el software MySQL Workbench.

También está disponible el archivo .sql en [este enlace](#).

Datos a insertar

Nombre de la tabla	Enlace a archivo CSV	Enlace a archivo SQL
cod_postal	Enlace	Enlace
colaborador	Enlace	Enlace
colaborador_dona_material	Enlace	Enlace
cuotas	Enlace	Enlace
envio	Enlace	Enlace
envio_has_material	Enlace	Enlace
equipo	Enlace	Enlace
evento	Enlace	Enlace
material	Enlace	Enlace
mision	Enlace	Enlace
sede	Enlace	Enlace
socix	Enlace	Enlace
voluntarix	Enlace	Enlace
voluntarix_has_equipo	Enlace	Enlace

Junto con los documentos arriba enlazados, se hace entrega de:

- [Hoja de cálculo](#) utilizada para la generación de los datos.
- [Archivo con todas las sentencias SQL](#) necesarias para la replicación de la Base de datos, con las inserciones realizadas.
- El [dump de la base de datos](#); que incluirá la creación de las tablas, los datos así como toda la programación realizada.

Procedimientos/Funciones

Procedimiento 1

Con el fin de promover entre las empresas colaboradoras donaciones más grandes, se desea obtener la cantidad donada de un colaborador introducido por parámetros.

El colaborador "A14764794" nos ha sobornado, por lo que el procedimiento deberá aumentar un 20% las donaciones de este. También nos pide que para el colaborador "A14764795" se reduzca en un 25% el resultado.

```
DELIMITER $$
DROP FUNCTION IF EXISTS mat_don_colab$$
CREATE FUNCTION mat_don_colab (colab_intro VARCHAR(10))
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE local_total INT;
SELECT SUM(cantidad) AS total
FROM colaborador_dona_material
GROUP BY colaborador_nif
HAVING colaborador_nif = colab_intro
INTO local_total;

IF colab_intro = 'A14764794' THEN
    SET local_total = local_total *1.20;
RETURN local_total;
END IF;

IF colab_intro = 'A14764795' THEN
    SET local_total = local_total *0.75;
RETURN local_total;
END IF;
RETURN local_total;
END$$

SELECT mat_don_colab('A14764795')$$
```

El resultado de la sentencia select es:

	mat_don_colab('A14764795')
▶	525

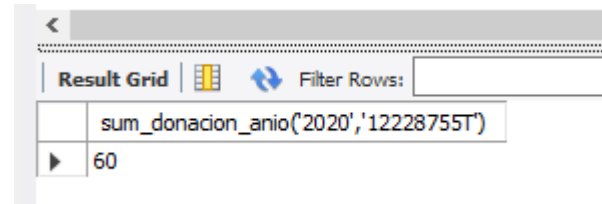
La suma real es de 700 unidades.

Procedimiento 2

Suma de todas las donaciones de un socio dado, agrupado por año (también introducido por parámetro), para la declaración de la renta.

```
DELIMITER $$

DROP FUNCTION IF EXISTS sum_donacion_anio$$
CREATE FUNCTION sum_donacion_anio(in_anio CHAR(4), in_socix VARCHAR(9))
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE resultado INT;
    SELECT SUM(importe)
    FROM cuotas
    WHERE LEFT(CONVERT(anio_mes,CHAR(4)),2) = RIGHT(in_anio,2)
    GROUP BY socio_dni_socio
    HAVING socio_dni_socio = in_socix INTO resultado;
    RETURN resultado;
END $$
```



sum_donacion_anio('2020','12228755T')
60

Procedimiento 3

Desde el Departamento de Calidad y Cumplimiento nos solicitan crear un procedimiento que verifica que todas los tipos de cuota se han introducido de forma correcta. Esto es debido a que el equipo de programación del formulario aceptaba “mensual”, “trimestral”, “anual” y “testBB”.

Nos piden que, como castigo para los graciosos y graciosas que utilizaron las herramientas de desarrollo del explorador para adivinar la posibilidad de “testBB”, se les defina la cuota en 20€ mensuales. En relación con el [cursor 3](#).

```
DELIMITER $$
DROP PROCEDURE IF EXISTS correct_form_tipo_cuota_1$$
CREATE PROCEDURE correct_form_tipo_cuota_1()
BEGIN
    UPDATE socix
        SET cuota_tipo = 'mensual'
        AND cuota_importe = 20
        WHERE cuota_tipo = 'testBB';
```

END\$\$

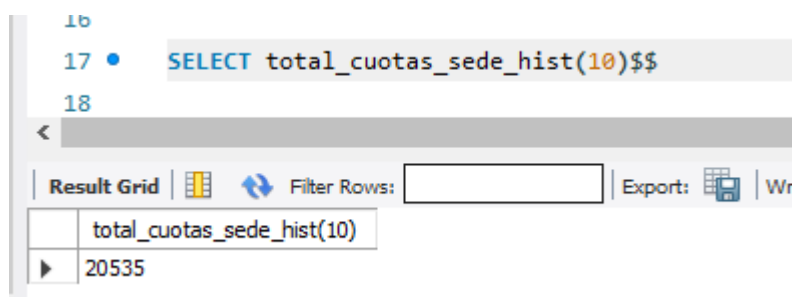
Procedimiento 4

Crear una función que devuelva, para una sede dada mediante parámetro, el histórico de las cuotas ingresadas por sus socios

```
-- Función histórica:
DELIMITER $$

DROP FUNCTION IF EXISTS total_cuotas_sede_hist$$

CREATE FUNCTION total_cuotas_sede_hist(in_sede INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_sede INT;
    SELECT SUM(c.importe)
    FROM cuotas as c, socix as s
    WHERE c.socio_dni_socio = s.dni_socio
    GROUP BY s.sede_idsede
    HAVING sede_idsede=in_sede INTO total_sede;
    RETURN total_sede;
END $$
```



Procedimiento 5

En relación al [procedimiento 3](#).

Calcular el porcentaje que se podrá deducir el socio teniendo en cuenta la [normativa vigente](#) en materia tributaria.

Base de la deducción, importe hasta	Porcentaje de la deducción
Hasta 150 euros	75
Resto base de deducción	30

```

DELIMITER $$
DROP FUNCTION IF EXISTS deduccion_hacienda$$
CREATE FUNCTION deduccion_hacienda (in_anio CHAR(4), in_socix
VARCHAR(9))
RETURNS DECIMAL(6,2)
DETERMINISTIC
BEGIN
    DECLARE total_anio INT;
    DECLARE a_deducir DECIMAL(6,2);
    SELECT sum_donacion_anio('2020',in_socix) INTO total_anio;

    IF total_anio < 151 THEN
        SET a_deducir = total_anio * 0.75;
        RETURN a_deducir;
    END IF;
    IF total_anio > 151 THEN
        SET a_deducir = ((150*0.75)+((total_anio-150)*0.30));
        RETURN a_deducir;
    END IF;
END $$

```

	sum_donacion_anio('2020','24724662F')
▶	1200

	deduccion_hacienda('2020','24724662F')
▶	427.50

Triggers

Trigger 1

Cuando se flete un envío, se reste la cantidad de este en el stock del producto.

```
DELIMITER $$
DROP TRIGGER IF EXISTS act_stock$$
CREATE TRIGGER act_stock AFTER INSERT
ON envio_has_material
FOR EACH ROW
BEGIN
    UPDATE material
    SET material.cantidad_en_stock =
        material.cantidad_en_stock - NEW.envio_has_material.cantidad
    WHERE material.idmaterial =
        NEW.envio_has_material.material_idmaterial;
END$$
```

Trigger 2

No permitir fletar un envío que contenga algún material que caduque en los siguientes 30 días.

```
DELIMITER $$
DROP TRIGGER IF EXISTS cad_prod_envio$$
CREATE TRIGGER cad_prod_envio BEFORE INSERT
ON envio_has_material
FOR EACH ROW
BEGIN
    DECLARE local_caducidad DATE;
    SELECT material.caducidad
    FROM material
    WHERE material.idmaterial =
        NEW.envio_has_material.material_idmaterial
    INTO local_caducidad;
    IF local_caducidad < DATE_ADD(current_date(),INTERVAL 30 day) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se permite el flete de envío
con productos con caducidad próxima';
        SET NEW.material_idmaterial = 99;
    END IF;
END$$
```

Trigger 3

No permitir enviar un equipo que está de misión en ese momento. No permitir enviarlo si el equipo ha estado en los últimos 3 meses.

```
DELIMITER $$
DROP TRIGGER IF EXISTS limit_fecha_mision$$
CREATE TRIGGER limit_fecha_mision BEFORE INSERT ON mision
FOR EACH ROW
BEGIN
    IF (SELECT idmision
        FROM mision
        WHERE equipo_id_equipo = NEW.equipo_id_equipo
        AND date_add(fecha_fin, INTERVAL 3 MONTH) < current_date) IS NOT
    NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se permite el envío de equipos
en misiones, con misiones recientes.';
    END IF;
END $$
```

Cursores

Cursor 1

En relación con el [evento 2](#) de este documento.
Cumplimentar la tabla “contabilidad_mensual” con el histórico.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS actualizar_historico_cuotas$$
CREATE PROCEDURE actualizar_historico_cuotas( )
BEGIN
    DECLARE num_filas INT DEFAULT 0;
    DECLARE cuenta_bucle INT DEFAULT 0;
    DECLARE anio_mes_local INT;
    DECLARE local_suma INT;
    DECLARE cursor_anio_mes_cuotas CURSOR FOR
        SELECT DISTINCT anio_mes FROM cuotas;
    SELECT found_rows() INTO num_filas;
    OPEN cursor_anio_mes_cuotas;
    WHILE cuenta_bucle < num_filas DO
        BEGIN
            FETCH cursor_anio_mes_cuotas INTO anio_mes_local;
            SELECT SUM(importe)
                FROM cuotas
                WHERE anio_mes = anio_mes_local
                GROUP BY anio_mes
                INTO local_suma;
            INSERT INTO contabilidad_mensual
            VALUES (CONVERT(LEFT(anio_mes_local,2),UNSIGNED),
                    CONVERT(RIGHT(anio_mes_local,2),UNSIGNED),
                    local_suma);
            SET cuenta_bucle = cuenta_bucle + 1;
        END;
    END WHILE;
    CLOSE cursor_anio_mes_cuotas;
END $$
```

Cursor 2

En relación con el [evento 1](#) de este documento.
Recolección de cuotas de socios, insertar en la tabla cuotas.
Teniendo en cuenta el tipo de cuota (anual, mensual, trimestral). Las cuotas mensuales se pasan por banco el día 1 de cada mes. Las trimestrales los meses 1,3,6,9 y los anuales en junio.


```

DELIMITER $$
DROP PROCEDURE IF EXISTS cobro_cuotas$$
CREATE PROCEDURE cobro_cuotas( )
BEGIN
    DECLARE num_filas INT DEFAULT 0;
    DECLARE cuenta_bucle INT DEFAULT 0;
    DECLARE local_dni_socio varchar(9);
    DECLARE local_cuota_tipo varchar(45);
    DECLARE local_cuota_importe INT;
    DECLARE cursor_cobro_cuotas CURSOR FOR
        SELECT dni_socio,cuota_tipo,cuota_importe
            FROM socix;
    SELECT found_rows() INTO num_filas;
    OPEN cursor_cobro_cuotas;
    WHILE cuenta_bucle < num_filas DO
        BEGIN
            FETCH cursor_cobro_cuotas INTO
local_dni_socio,local_cuota_tipo,local_cuota_importe;
            -- Todos los meses, tipo mensual se cobra.
            IF local_cuota_tipo = 'mensual' THEN
                INSERT INTO cuotas VALUES (DATE_FORMAT(NOW(),
'%y%m'),local_dni_socio,local_cuota_importe);
            END IF;
            -- Los meses que sean 1,3,6,9, se cobran las trimestrales
            IF ((local_cuota_tipo = 'trimestral') AND
(MONTH(current_date()) = (1 OR 3 OR 6 OR 9) )) THEN
                INSERT INTO cuotas VALUES (DATE_FORMAT(NOW(),
'%y%m'),local_dni_socio,local_cuota_importe);
            END IF;
            -- Los meses 6, se cobran las anuales
            IF ((local_cuota_tipo = 'anual') AND
(MONTH(current_date())=6)) THEN
                INSERT INTO cuotas VALUES (DATE_FORMAT(NOW(),
'%y%m'),local_dni_socio,local_cuota_importe);
            END IF;
            SET cuenta_bucle = cuenta_bucle + 1;
        END;
    END WHILE;
    CLOSE cursor_cobro_cuotas;
END $$



```

Si ejecutamos el procedimiento/cursor de forma aislada, se insertan los valores correspondientes:

3

4 • `SELECT * FROM casicruzroja.cuotas;`

<

Result Grid   Filter Rows: Edit:

	anio_mes	socio_dni_socio	importe
▶	2112	00079063N	5
	2112	00189107R	15
	2112	00193318A	5
	2112	00320827T	10
	2112	00344217E	15
	2112	00453322S	15
	2112	00688932J	50
	2112	00813999Y	5
	2112	00943709L	50
	2112	01129401D	50
	2112	01193156P	100
	2112	01291998L	10
	2112	01348201X	100
	2112	01582234H	100
	2112	01807440P	10
	2112	01865757C	10
	2112	01952276J	5
	2112	01961723F	100
	2112	02117675L	10
	2112	02206990W	10
	2112	02287047L	5
	2112	02413091T	5
	2112	02624447D	5
	2112	03229121J	5
	2112	03400988R	5
	2112	03598932F	100
	2112	03958106J	5

Se creará el [evento.1](#)

Cursor 3

Hacer un cursor que permita introducir en la tabla de nueva creación "contabilidad_mensual_sede" los ingresos del mes para todas las sedes.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS proc_conta_mensual_sede$$
CREATE PROCEDURE proc_conta_mensual_sede()
BEGIN
    DECLARE num_filas INT DEFAULT 0;
    DECLARE cuenta_bucle INT DEFAULT 0;
    DECLARE local_idsede INT DEFAULT 0;
    DECLARE local_suma_sede INT DEFAULT 0;
    DECLARE cursor_conta_mensual_sede CURSOR FOR
        SELECT DISTINCT idsede FROM sede;
    SELECT found_rows() INTO num_filas;
    OPEN cursor_conta_mensual_sede;
    WHILE cuenta_bucle < num_filas DO
        BEGIN
            FETCH cursor_conta_mensual_sede INTO local_idsede;
            SELECT SUM(importe)
            FROM cuotas as c, socix as s
            WHERE c.socio_dni_socio = s.dni_socio
                AND anio_mes = concat(
                    (CAST DATE_FORMAT(NOW(), '%y') AS UNSIGNED),
                    CAST DATE_FORMAT(NOW(), '%m') AS UNSIGNED))
                AND socix.sede_idsede = local_idsede
            INTO local_suma_sede;
            INSERT INTO contabilidad_mensual_sede VALUES
                (CAST DATE_FORMAT(NOW(), '%y') AS UNSIGNED),
                CAST DATE_FORMAT(NOW(), '%m') AS UNSIGNED,
                local_idsede,
                local_suma_sede);
            SET cuenta_bucle = cuenta_bucle + 1;
        END;
    END WHILE;
    CLOSE cursor_conta_mensual_sede;
END $$
CALL proc_conta_mensual_sede$$
```

Eventos

Evento 1

En relación con el [cursor 2](#)

Recolección de cuotas de socios, insertar en la tabla cuotas.

Teniendo en cuenta el tipo de cuota (anual, mensual, trimestral). Las cuotas mensuales se pasan por banco el día 1 de cada mes. Las trimestrales los meses 1,3,6,9 y los anuales en junio.

```
DELIMITER $$
DROP EVENT IF EXISTS conta_mensual$$
CREATE EVENT conta_mensual ON SCHEDULE
EVERY 1 MONTH
STARTS "2021-12-01 04:00:00"
ENDS "2025-12-02 05:00:00"
DO
BEGIN
    CALL cobro_cuotas( );
END$$
```

Evento 2

Crear una tabla llamada "contabilidad_mensual". El día 15 de cada mes se deberán contabilizar la suma de los ingresos del mes dado.

```
CREATE TABLE `contabilidad_mensual` (
  `anio` INT NOT NULL,
  `mes` INT NOT NULL,
  `ingresos` INT NOT NULL,
  PRIMARY KEY (`anio`, `mes`))
ENGINE = InnoDB;
DELIMITER $$
DROP EVENT IF EXISTS conta_mensual$$
CREATE EVENT conta_mensual ON SCHEDULE
EVERY 1 MONTH
STARTS "2021-12-15 05:00:00"
ENDS "2025-12-15 05:00:00"
DO
BEGIN
    DECLARE local_suma INT;
```

```

SELECT SUM(importe)
FROM casicruzroja.cuotas
WHERE anio_mes = (SELECT DATE_FORMAT(NOW(), '%y%m'))
INTO local_suma;

INSERT INTO contabilidad_mensual
VALUES (CAST(DATE_FORMAT(NOW(), '%y') AS UNSIGNED),
        CAST(DATE_FORMAT(NOW(), '%m') AS UNSIGNED),
        local_suma);
END$$

```

Para la comprobación se puede utilizar la siguiente sentencia:

```

INSERT INTO contabilidad_mensual
VALUES (CAST(DATE_FORMAT(NOW(), '%y') AS UNSIGNED),
        CAST(DATE_FORMAT(NOW(), '%m') AS UNSIGNED),
        25);

```

Que da como resultado la siguiente inserción en la tabla:

	anio	mes	ingresos
▶	21	12	25

Evento 3

En relación con la [función/procedimiento 4](#) y el , crear una tabla en la que cada mes se almacene la cotización de las cuotas de cada sede.

```

CREATE TABLE `contabilidad_mensual_sede` (
  `anio` INT NOT NULL,
  `mes` INT NOT NULL,
  `sede` INT NOT NULL,
  `ingresos` INT NOT NULL,
  PRIMARY KEY (`anio`, `mes`, `sede`))
ENGINE = InnoDB;
DROP EVENT IF EXISTS conta_mensual_sede$$
CREATE EVENT conta_mensual_sede ON SCHEDULE
EVERY 1 MONTH
STARTS "2021-12-15 05:00:00"
ENDS "2025-12-15 05:00:00"
DO
BEGIN
  CALL proc_conta_mensual_sede;
END $$

```