Desarrollo de aplicación web en raspberry pi como uso de portal del empleado

Voy a desarrollar una aplicación web haciendo uso de HTML y PHP que tenga como objetivo gestionar y supervisar los turnos de trabajo de los empleados así como solicitar y aprobar sus vacaciones e incluso notificar y justificar ausencias.

Inicialización de la base de datos

En primer lugar para desarrollar la aplicación web, se debe elegir la base de datos que se va a utilizar y posteriormente diseñar las tablas que se van a utilizar según el uso que se le quiere dar a la aplicación web.

Se va a utilizar MariaDB ya que es una base de datos con la que me siento cómodo, y tiene una alta compatibilidad con MySQL que es la base de datos utilizada en clase este año, y es software libre, lo mas importante debido a la naturaleza de este proyecto.

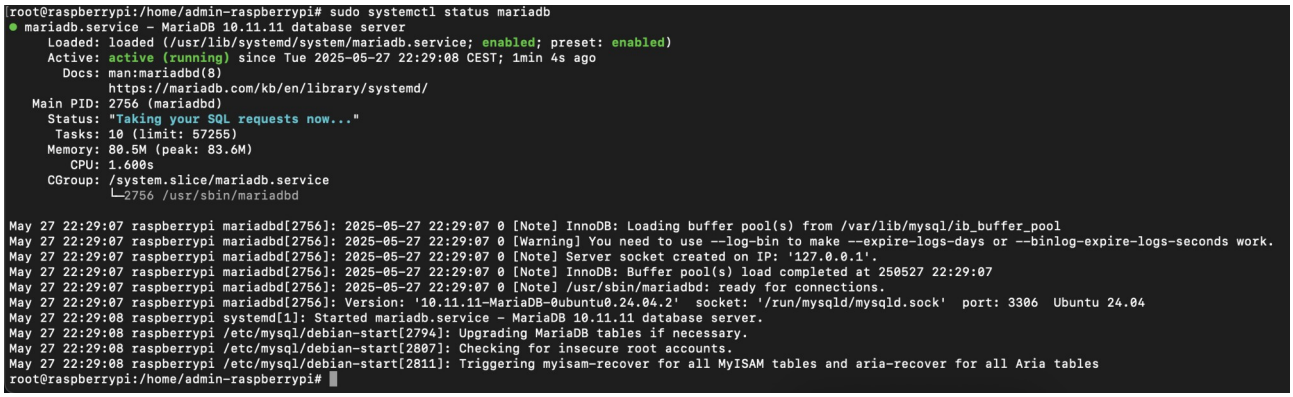
Instalación

Para instalar la base de datos haremos uso de los comandos:

·sudo apt update && sudo apt install mariadb-server -y

Tras instalarlo verificamos que esta funcionando correctamente con el comando:

·sudo systemctl status mariadb



La base de datos arrancó correctamente

Refuerzo de la seguridad

MariaDB hace uso de funciones de MySQL, y en base a esto podemos hacer uso del comando:

·sudo mysql_secure_installation

el cual mejora la seguridad de la base de datos con configuraciones como:

·Establecer o cambiar la contraseña de root : planetas-sa-db

·Restringir el acceso remoto de root

·Borrar la base de datos de prueba

Ejecuto el comando:

```
root@raspberrypi:/home/admin-raspberrypi# sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Tras ejecutar el comando se ha deshabilitado el acceso remoto al usuario root, se ha establecido la contraseña de root como “planetas-sa-db” y se ha borrado la base de datos de prueba y los usuarios anónimos

Creación de la base de datos

Ahora con la base de datos limpia, vamos a crear el usuario dueño de la base de datos de la aplicación para el portal de el empleado y el usuario que iniciará sesión desde la aplicación web para hacer consultas a la base de datos, por lo que será imprescindible ajustar los privilegios necesarios para el correcto funcionamiento de la aplicación pero sin tener potenciales amenazas a la seguridad de la base de datos por el uso de excesivos permisos

Procedo a crear el usuario dueño de la base de datos desde el usuario root, desde el cual crearemos las tablas, triggers y procedimientos

Base de datos “planetas_sa”

Desde el usuario “root” conectamos a la base de datos mediante el siguiente comando:

·sudo mysql -u root -p

```
root@raspberrypi:/home/admin-raspberrypi# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.11.11-MariaDB-0ubuntu0.24.04.2 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Creamos la base de datos “planetas_sa” con el comando:

CREATE DATABASE planetas_sa;

```
MariaDB [(none)]> CREATE DATABASE planetas_sa;
Query OK, 1 row affected (0.001 sec)
```

Base de datos creada

Usuario “admin-db”

Voy a crear un usuario que gestione la base de datos “planetas_sa” llamado “admin-db” con contraseña: planetas-sa-2025-admin-db con el siguiente comando:

·CREATE USER 'ADMIN_DB'@'localhost' IDENTIFIED BY 'planetas-sa-2025-admin-db';

```
MariaDB [(none)]> CREATE USER 'ADMIN_DB'@'localhost' IDENTIFIED BY 'planetas-sa-2025-admin-db';
Query OK, 0 rows affected (0.009 sec)
```

```
MariaDB [(none)]> SELECT User, Host FROM mysql.user WHERE User = 'ADMIN_DB';
+----------+-----------+
| User     | Host      |
+----------+-----------+
| ADMIN_DB | localhost |
+----------+-----------+
1 row in set (0.004 sec)
```

Usuario creado, ahora vamos a asignarle privilegios.

El usuario tendrá los siguientes privilegios:

·Conectarse a la base de datos

·Crear, modificar y eliminar tablas

·Crear y gestionar índices en las tablas

·Administrar registros en todas las tablas

·Crear y modificar triggers, procedimientos y funciones

·Aportar privilegios sobre las tablas a otros usuarios

·Aportar privilegios de ejecucion de procedimientos a otros usariosa

Con los siguientes comandos

GRANT CREATE, ALTER, DROP, INDEX, INSERT, UPDATE, DELETE, SELECT, EXECUTE ON planetas_sa.* TO 'admin_db'@'localhost';

```
MariaDB [(none)]> GRANT CREATE, ALTER, DROP, INDEX, INSERT, UPDATE, DELETE, SELECT, EXECUTE ON planetas_sa.* TO 'ADMIN_DB'@'localhost';
Query OK, 0 rows affected (0.009 sec)
```

GRANT CREATE ROUTINE, ALTER ROUTINE ON planetas_sa.* TO 'ADMIN_DB'@'localhost';

```
MariaDB [(none)]> GRANT CREATE ROUTINE, ALTER ROUTINE ON planetas_sa.* TO 'ADMIN_DB'@'localhost';
Query OK, 0 rows affected (0.008 sec)
```

GRANT USAGE ON *.* TO 'ADMIN_DB'@'localhost';

```
MariaDB [(none)]> GRANT USAGE ON *.* TO 'ADMIN_DB'@'localhost';
Query OK, 0 rows affected (0.008 sec)
```

GRANT SELECT, INSERT, UPDATE, DELETE ON planetas_sa.* TO 'ADMIN_DB'@'localhost' WITH GRANT OPTION;

```
MariaDB [(none)]> GRANT SELECT, INSERT, UPDATE, DELETE ON planetas_sa.* TO 'ADMIN_DB'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0.008 sec)
```

Objetos en “planetas_sa”

Tablas

Para crear los objetos de la base de datos se va a utilizar el usuario “admin-db” creado para este fin.

La base de datos va a contener la siguiente estructura de tablas:

·Tabla empleados:

```
CREATE TABLE empleados (
    id_empleado INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    dni VARCHAR(20) UNIQUE NOT NULL,
    id_departamento INT,
    id_turno INT,
    puesto VARCHAR(50),
    telefono VARCHAR(15),
    uid_ldap VARCHAR(100) UNIQUE NOT NULL,
    cuenta_pidgin VARCHAR(100) UNIQUE NOT NULL,
    fecha_contratacion DATE NOT NULL,
    dias_vacaciones_disponibles INT DEFAULT 30,
    FOREIGN KEY (id_departamento) REFERENCES departamentos(id_departamento),
    FOREIGN KEY (id_turno) REFERENCES turnos(id_turno)
);
```

·Tabla departamentos:

```
CREATE TABLE departamentos (
    id_departamento INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL UNIQUE,
    id_responsable INT NULL,
    FOREIGN KEY (id_responsable) REFERENCES empleados(id_empleado)
);
```

·Tabla turnos:

```
CREATE TABLE turnos (
    id_turno INT AUTO_INCREMENT PRIMARY KEY,
    descripcion VARCHAR(50) NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_fin TIME NOT NULL,
    dias_semana SET('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo') NOT NULL,
    horario_flexible BOOLEAN DEFAULT FALSE
);
```

·Tabla fichajes:

```
CREATE TABLE fichajes (
    id_fichaje INT AUTO_INCREMENT PRIMARY KEY,
    id_empleado INT NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    tipo ENUM('entrada', 'salida') NOT NULL,
    fichaje_modificado BOOLEAN DEFAULT FALSE;
    FOREIGN KEY (id_empleado) REFERENCES empleados(id_empleado)
);
```

·Tabla incidencias_fichaje:

```
CREATE TABLE incidencias_fichaje (
    id_incidencia INT AUTO_INCREMENT PRIMARY KEY,
    id_fichaje INT NOT NULL,
    descripcion TEXT NOT NULL,
    FOREIGN KEY (id_fichaje) REFERENCES fichajes(id_fichaje)
);
```

·Tabla ausencias:

```
CREATE TABLE ausencias (
    id_ausencia INT AUTO_INCREMENT PRIMARY KEY,
    id_empleado INT NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL,
```

```
    motivo TEXT NOT NULL,
    justificada BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (id_empleado) REFERENCES empleados(id_empleado)
);
```

·Tabla solicitudes_vacaciones:

```
CREATE TABLE solicitudes_vacaciones (
    id_solicitud INT AUTO_INCREMENT PRIMARY KEY,
    id_empleado INT NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL,
    estado_solicitud ENUM('pendiente', 'aprobada', 'rechazada') DEFAULT 'pendiente',
    FOREIGN KEY (id_empleado) REFERENCES empleados(id_empleado)
);
```

·Tabla correciones_fichaje:

```
CREATE TABLE correcciones_fichaje (
    id_correccion INT AUTO_INCREMENT PRIMARY KEY,
    id_fichaje INT NOT NULL,
    id_empleado_modificador INT NOT NULL,
    motivo TEXT NOT NULL,
    tipo_antiguo ENUM('entrada', 'salida'),
    tipo_nuevo ENUM('entrada', 'salida'),
    hora_antigua TIME,
    hora_nueva TIME,
    fecha_modificacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_fichaje) REFERENCES fichajes(id_fichaje),
    FOREIGN KEY (id_empleado_modificador) REFERENCES empleados(id_empleado)
);
```

·Tabla para registrar las modificaciones sobre el estado de las solicitudes de vacaciones:

```
CREATE TABLE modificaciones_estado_vacaciones (
    id_modificacion INT AUTO_INCREMENT PRIMARY KEY,
    id_solicitud INT NOT NULL,
    id_empleado_modificador INT NOT NULL,
```

```
    estado_anterior ENUM('pendiente', 'aprobado', 'rechazado') NOT NULL,
    estado_nuevo ENUM('pendiente', 'aprobado', 'rechazado') NOT NULL,
    motivo TEXT NOT NULL,
    fecha_modificacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_solicitud) REFERENCES solicitudes_vacaciones(id_solicitud),
    FOREIGN KEY (id_empleado_modificador) REFERENCES empleados(id_empleado)
);
```

·Tabla para registrar quien justifica ausencias:

```
CREATE TABLE modificaciones_ausencias (
    id_modificacion INT AUTO_INCREMENT PRIMARY KEY,
    id_ausencia INT NOT NULL,
    id_empleado_modificador INT NOT NULL,
    fecha_modificacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_ausencia) REFERENCES ausencias(id_ausencia),
    FOREIGN KEY (id_empleado_modificador) REFERENCES empleados(id_empleado)
);
```

Indices

CREATE INDEX idx_empleado_fichajes ON fichajes(id_empleado, fecha);

CREATE INDEX idx_empleado_ausencias ON ausencias(id_empleado, fecha_inicio, fecha_fin);

CREATE INDEX idx_empleado_vacaciones ON solicitudes_vacaciones(id_empleado, fecha_inicio, fecha_fin);

Triggers

Trigger para descontar días de vacaciones tras aprobar una solicitud:

```
DELIMITER //

CREATE TRIGGER actualizar_vacaciones
AFTER UPDATE ON solicitudes_vacaciones
FOR EACH ROW
BEGIN
    IF NEW.estado_solicitud = 'aprobada' THEN
        UPDATE empleados
        SET dias_vacaciones_disponibles = dias_vacaciones_disponibles - DATEDIFF(NEW.fecha_fin, NEW.fecha_inicio)
        WHERE id_empleado = NEW.id_empleado;
    END IF;
END;
//

DELIMITER ;
```

Trigger para validar la fecha de las ausencias:

```
DELIMITER //

CREATE TRIGGER validar_fechas_ausencias
BEFORE INSERT ON ausencias
FOR EACH ROW
BEGIN
    IF NEW.fecha_fin < NEW.fecha_inicio THEN
        SIGNAL SQLSTATE '45000'
```

```
        SET MESSAGE_TEXT = 'Error: fecha_fin no puede ser menor que fecha_inicio en ausencias';
    END IF;
END;
//

DELIMITER ;
```

Trigger para validar la fecha de las vacaciones:

```
DELIMITER //

CREATE TRIGGER validar_fechas_vacaciones
BEFORE INSERT ON solicitudes_vacaciones
FOR EACH ROW
BEGIN
    IF NEW.fecha_fin < NEW.fecha_inicio THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: fecha_fin no puede ser menor que fecha_inicio en solicitudes de vacaciones';
    END IF;
END;
//

DELIMITER ;
```

Trigger para registrar fichajes tardios:

```
DELIMITER //

CREATE TRIGGER registrar_incidente_entrada
AFTER INSERT ON fichajes
FOR EACH ROW
BEGIN
    DECLARE turno_inicio TIME;
    DECLARE flexible BOOLEAN;
```

```
    -- Obtener la hora de inicio del turno y si es flexible
    SELECT hora_inicio, horario_flexible INTO turno_inicio, flexible
    FROM turnos
    WHERE id_turno = (SELECT id_turno FROM empleados WHERE id_empleado =
NEW.id_empleado);

    -- Comprobar si el fichaje de entrada es tardío (más de 3 minutos después de la hora de inicio)
    IF NEW.tipo = 'entrada' AND NEW.hora > ADDTIME(turno_inicio, '00:03:00') THEN
        IF flexible = FALSE OR TIMESTAMPDIFF(HOUR, turno_inicio, NEW.hora) > 2 THEN
            INSERT INTO incidencias_fichaje (id_fichaje, descripcion)
            VALUES (NEW.id_fichaje, CONCAT('Fichaje tardío a las ', NEW.hora));
        END IF;
    END IF;
END;
//

DELIMITER ;
```

Trigger para registrar fichajes de salida antes de tiempo:

```
DELIMITER //

CREATE TRIGGER registrar_incidente_salida
AFTER INSERT ON fichajes
FOR EACH ROW
BEGIN
    DECLARE turno_fin TIME;
    DECLARE flexible BOOLEAN;

    -- Obtener hora de fin del turno y si es flexible
    SELECT hora_fin, horario_flexible INTO turno_fin, flexible
    FROM turnos
    WHERE id_turno = (SELECT id_turno FROM empleados WHERE id_empleado =
NEW.id_empleado);
```

```
    -- Comprobar si es fichaje de salida temprano
    IF NEW.tipo = 'salida' AND NEW.hora < turno_fin THEN
        IF flexible = FALSE OR TIMESTAMPDIFF(HOUR, NEW.hora, turno_fin) > 2 THEN
            INSERT INTO incidencias_fichaje (id_fichaje, descripcion)
            VALUES (NEW.id_fichaje, CONCAT('Salida temprana a las ', NEW.hora));
        END IF;
    END IF;
END;
//

DELIMITER ;
```

·Trigger para actualizar fichaje_modificado y registrar en correcciones_fichaje:

```
DELIMITER //

CREATE TRIGGER registrar_correccion_fichaje
BEFORE UPDATE ON fichajes
FOR EACH ROW
BEGIN
    -- Registrar modificación de tipo de fichaje
    IF OLD.tipo <> NEW.tipo AND OLD.hora = NEW.hora THEN
        INSERT INTO correcciones_fichaje (id_fichaje, id_empleado_modificador, motivo, tipo_antiguo, tipo_nuevo, hora_antigua, hora_nueva)
        VALUES (OLD.id_fichaje, @id_empleado_modificador, 'Modificación de tipo de fichaje', OLD.tipo, NEW.tipo, NULL, NULL);
    END IF;

    -- Registrar modificación de hora de fichaje
    IF OLD.hora <> NEW.hora AND OLD.tipo = NEW.tipo THEN
        INSERT INTO correcciones_fichaje (id_fichaje, id_empleado_modificador, motivo, tipo_antiguo, tipo_nuevo, hora_antigua, hora_nueva)
```

```
        VALUES (OLD.id_fichaje, @id_empleado_modificador, 'Modificación de hora de fichaje', NULL, NULL, OLD.hora, NEW.hora);
    END IF;

    -- Registrar modificación de tipo y hora de fichaje
    IF OLD.tipo <> NEW.tipo AND OLD.hora <> NEW.hora THEN
        INSERT INTO correcciones_fichaje (id_fichaje, id_empleado_modificador, motivo, tipo_antiguo, tipo_nuevo, hora_antigua, hora_nueva)
        VALUES (OLD.id_fichaje, @id_empleado_modificador, 'Modificación de tipo y hora de fichaje', OLD.tipo, NEW.tipo, OLD.hora, NEW.hora);
    END IF;

    -- Marcar `fichaje_modificado = TRUE` si hubo cambios
    IF OLD.tipo <> NEW.tipo OR OLD.;hora <> NEW.hora THEN
        SET NEW.fichaje_modificado = TRUE;
    END IF;
END;
//

DELIMITER ;
```

·Trigger para impedir que el campo fichaje_modificado de la tabla fichajes sea modificado manualmente de true a false:

```
DELIMITER //

CREATE TRIGGER bloquear_cambio_fichaje_modificado
BEFORE UPDATE ON fichajes
FOR EACH ROW
BEGIN
    -- Si fichaje_modificado ya es TRUE y alguien intenta cambiarlo a FALSE, bloquearlo
    IF OLD.fichaje_modificado = TRUE AND NEW.fichaje_modificado = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: No puedes cambiar fichaje_modificado de TRUE a FALSE.';
    END IF;
```

```
END;
//

DELIMITER ;
DELIMITER //

DROP TRIGGER IF EXISTS registrar_modificacion_estado_vacaciones;

CREATE TRIGGER registrar_modificacion_estado_vacaciones
AFTER UPDATE ON solicitudes_vacaciones
FOR EACH ROW
BEGIN
    --  Verificar si ha cambiado el estado de la solicitud
    IF OLD.estado_solicitud <> NEW.estado_solicitud THEN
        --  Insertar en modificaciones_estado_vacaciones con el ID del empleado modificador actualizado
        INSERT INTO modificaciones_estado_vacaciones (id_solicitud, id_empleado_modificador, estado_anterior, estado_nuevo, motivo)
        VALUES (OLD.id_solicitud, NEW.id_empleado_modificador, OLD.estado_solicitud, NEW.estado_solicitud, 'Cambio de estado registrado automáticamente');
    END IF;
END //

DELIMITER ;
```

·Trigger para validar la cantidad de dias de vacaciones que puede solicitar el empleado en relacion a los dias que le quedan:

```
DELIMITER //

CREATE TRIGGER validar_dias_vacaciones
BEFORE INSERT ON solicitudes_vacaciones
FOR EACH ROW
BEGIN
```

```
    -- Validar que el empleado tenga suficientes días de vacaciones
    IF (SELECT dias_vacaciones_disponibles FROM empleados WHERE id_empleado =
NEW.id_empleado) <
       (DATEDIFF(NEW.fecha_fin, NEW.fecha_inicio) + 1) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: No tienes suficientes días de vacaciones disponibles';
    END IF;
END;
//
DELIMITER ;
```

Eventos

Evento para renovar los dias de vacaciones cada 1 de enero:

```
DELIMITER //

CREATE EVENT renovar_vacaciones
ON SCHEDULE EVERY 1 YEAR
STARTS '2026-01-01 00:00:00'
DO
BEGIN
    UPDATE empleados
    SET dias_vacaciones_disponibles = dias_vacaciones_disponibles + 30;
END;
//

DELIMITER ;
```

Evento para registrar ausencias injustificadas:

```
DELIMITER //

CREATE EVENT verificar_ausencias_diarias
ON SCHEDULE EVERY 1 DAY
```

```
STARTS '2025-05-29 00:00:00'
DO
BEGIN
    INSERT INTO ausencias (id_empleado, fecha_inicio, fecha_fin, motivo, justificada)
    SELECT e.id_empleado, CURDATE(), CURDATE(), 'Ausencia injustificada', FALSE
    FROM empleados e
    LEFT JOIN fichajes f ON e.id_empleado = f.id_empleado AND f.fecha = CURDATE()
    WHERE f.id_fichaje IS NULL
    AND e.id_empleado NOT IN (
        SELECT id_empleado FROM solicitudes_vacaciones WHERE fecha_inicio <= CURDATE() AND fecha_fin >= CURDATE() AND estado_solicitud = 'aprobada'
    )
    AND e.id_empleado NOT IN (
        SELECT id_empleado FROM ausencias WHERE fecha_inicio <= CURDATE() AND fecha_fin >= CURDATE() AND justificada = TRUE
    );
END;
//

DELIMITER ;
```

·Trigger para registrar justificaciones de ausencias:

```
DELIMITER //

DROP TRIGGER IF EXISTS registrar_modificacion_ausencia;

CREATE TRIGGER registrar_modificacion_ausencia
AFTER UPDATE ON ausencias
FOR EACH ROW
BEGIN
    --  Registrar la modificación solo si la ausencia ha sido justificada
    IF OLD.justificada = FALSE AND NEW.justificada = TRUE THEN
        INSERT INTO modificaciones_ausencias (id_ausencia, id_empleado_modificador)
```

```
        VALUES (NEW.id_ausencia, NEW.id_empleado_modificador);
    END IF;
END //

DELIMITER ;
```

Procedimientos

·Procedimiento para insertar empleado:

```
DELIMITER //

CREATE PROCEDURE insertar_empleado(
    IN p_nombre VARCHAR(100),
    IN p_apellido VARCHAR(100),
    IN p_dni VARCHAR(20),
    IN p_id_departamento INT,
    IN p_id_turno INT,
    IN p_puesto VARCHAR(50),
    IN p_telefono VARCHAR(15),
    IN p_uid_ldap VARCHAR(100),
    IN p_cuenta_pidgin VARCHAR(100),
    IN p_fecha_contratacion DATE
)
BEGIN
    -- Insertar el nuevo empleado sin especificar `dias_vacaciones_disponibles`
    INSERT INTO empleados (nombre, apellido, dni, id_departamento, id_turno, puesto, telefono, uid_ldap, cuenta_pidgin, fecha_contratacion)
    VALUES (p_nombre, p_apellido, p_dni, p_id_departamento, p_id_turno, p_puesto, p_telefono, p_uid_ldap, p_cuenta_pidgin, p_fecha_contratacion);

    -- Confirmación del procedimiento
    SELECT 'Empleado insertado correctamente' AS mensaje;
END;
```

```
//

DELIMITER ;
·Procedimiento para borrar empleado:
DELIMITER //

CREATE PROCEDURE borrar_empleado(IN p_id_empleado INT)
BEGIN
    -- Verificar si el empleado existe antes de borrarlo
    IF EXISTS (SELECT 1 FROM empleados WHERE id_empleado = p_id_empleado) THEN
        DELETE FROM empleados WHERE id_empleado = p_id_empleado;
        SELECT 'Empleado eliminado correctamente' AS mensaje;
    ELSE
        SELECT 'Error: El empleado no existe' AS mensaje;
    END IF;
END;
//

DELIMITER ;

·Procedimiento para insertar departamento:
DELIMITER //

CREATE PROCEDURE insertar_departamento(
    IN p_nombre VARCHAR(100),
    IN p_id_responsable INT
)
BEGIN
    -- Verificar si el departamento ya existe
    IF EXISTS (SELECT 1 FROM departamentos WHERE nombre = p_nombre) THEN
        SELECT 'Error: El departamento ya existe' AS mensaje;
    ELSE
```

```
        -- Insertar el nuevo departamento
        INSERT INTO departamentos (nombre, id_responsable)
        VALUES (p_nombre, p_id_responsable);

        -- Confirmación del procedimiento
        SELECT 'Departamento insertado correctamente' AS mensaje;
    END IF;
END;
//
```

·Procedimiento para borrar departamento:

```
DELIMITER //

CREATE PROCEDURE borrar_departamento(IN p_id_departamento INT)
BEGIN
    -- Verificar si el departamento existe
    IF EXISTS (SELECT 1 FROM departamentos WHERE id_departamento = p_id_departamento) THEN

        -- Verificar si el departamento tiene empleados asignados
        IF EXISTS (SELECT 1 FROM empleados WHERE id_departamento = p_id_departamento) THEN
            SELECT 'Error: No se puede borrar, hay empleados asignados a este departamento' AS mensaje;
        ELSE
            DELETE FROM departamentos WHERE id_departamento = p_id_departamento;
            SELECT 'Departamento eliminado correctamente' AS mensaje;
        END IF;

    ELSE
        SELECT 'Error: El departamento no existe' AS mensaje;
    END IF;
END;
//
```

```
DELIMITER ;

DELIMITER ;
```

·Procedimiento para insertar turno:

```
DELIMITER //

CREATE PROCEDURE insertar_turno(
    IN p_descripcion VARCHAR(50),
    IN p_hora_inicio TIME,
    IN p_hora_fin TIME,
    IN p_dias_semana SET('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo'),
    IN p_horario_flexible BOOLEAN
)
BEGIN
    -- Insertar el nuevo turno
    INSERT INTO turnos (descripcion, hora_inicio, hora_fin, dias_semana, horario_flexible)
    VALUES (p_descripcion, p_hora_inicio, p_hora_fin, p_dias_semana, p_horario_flexible);

    -- Confirmación del procedimiento
    SELECT 'Turno insertado correctamente' AS mensaje;
END;
//

DELIMITER ;
```

·Procedimiento para borrar turnos:

```
DELIMITER //

CREATE PROCEDURE borrar_turno(IN p_id_turno INT)
BEGIN
    -- Verificar si el turno existe antes de borrarlo
    IF EXISTS (SELECT 1 FROM turnos WHERE id_turno = p_id_turno) THEN
```

```
        -- Verificar si hay empleados asignados a este turno
        IF EXISTS (SELECT 1 FROM empleados WHERE id_turno = p_id_turno) THEN
            SELECT 'Error: No se puede borrar, hay empleados asignados a este turno' AS mensaje;
        ELSE
            DELETE FROM turnos WHERE id_turno = p_id_turno;
            SELECT 'Turno eliminado correctamente' AS mensaje;
        END IF;

    ELSE
        SELECT 'Error: El turno no existe' AS mensaje;
    END IF;
END;
//

DELIMITER ;
·Procedimiento para insertar fichaje:
DELIMITER //

CREATE PROCEDURE insertar_fichaje(
    IN p_id_empleado INT,
    IN p_fecha DATE,
    IN p_hora TIME,
    IN p_tipo ENUM('entrada', 'salida')
)
BEGIN
    -- Insertar el fichaje
    INSERT INTO fichajes (id_empleado, fecha, hora, tipo)
    VALUES (p_id_empleado, p_fecha, p_hora, p_tipo);

    -- Confirmación del procedimiento
    SELECT 'Fichaje insertado correctamente' AS mensaje;
```

```
END;
//

DELIMITER ;
```

·Procedimiento para insertar solicitudes de vacaciones:

```
DELIMITER //

CREATE PROCEDURE insertar_solicitud_vacaciones(
    IN p_id_empleado INT,
    IN p_fecha_inicio DATE,
    IN p_fecha_fin DATE
)
BEGIN
    -- Verificar que la fecha de inicio no sea posterior a la fecha de fin
    IF p_fecha_inicio > p_fecha_fin THEN
        SELECT 'Error: La fecha de inicio no puede ser posterior a la fecha de fin' AS mensaje;
    ELSE
        -- Insertar la solicitud con estado por defecto 'pendiente'
        INSERT INTO solicitudes_vacaciones (id_empleado, fecha_inicio, fecha_fin)
        VALUES (p_id_empleado, p_fecha_inicio, p_fecha_fin);

        -- Confirmación del procedimiento
        SELECT 'Solicitud de vacaciones insertada correctamente' AS mensaje;
    END IF;
END;
//

DELIMITER ;
```

·Procedimiento para aprobar solicitudes de vacaciones:

```
DELIMITER //

CREATE PROCEDURE aprobar_solicitud_vacaciones(
    IN p_id_solicitud INT,
    IN p_id_empleado_modificador INT
)
BEGIN
    -- Verificar si la solicitud existe y sigue pendiente
    IF EXISTS (SELECT 1 FROM solicitudes_vacaciones WHERE id_solicitud = p_id_solicitud AND estado_solicitud = 'pendiente') THEN
        -- Actualizar el estado a 'aprobada'
        UPDATE solicitudes_vacaciones
        SET estado_solicitud = 'aprobada'
        WHERE id_solicitud = p_id_solicitud;

        -- Registrar la modificación en modificaciones_estado_vacaciones
        INSERT INTO modificaciones_estado_vacaciones (id_solicitud, id_empleado_modificador, estado_anterior, estado_nuevo, motivo)
        VALUES (p_id_solicitud, p_id_empleado_modificador, 'pendiente', 'aprobada', 'Solicitud aprobada por RRHH');

        -- Confirmación del procedimiento
        SELECT 'Solicitud de vacaciones aprobada correctamente' AS mensaje;
    ELSE
        SELECT 'Error: La solicitud no existe o ya fue procesada' AS mensaje;
    END IF;
END;
//

DELIMITER ;
```

·Procedimiento para denegar solicitudes de vacaciones:

```
DELIMITER //

CREATE PROCEDURE denegar_solicitud_vacaciones(
    IN p_id_solicitud INT,
    IN p_id_empleado_modificador INT,
    IN p_motivo TEXT
)
BEGIN
    -- Verificar si la solicitud existe y sigue pendiente
    IF EXISTS (SELECT 1 FROM solicitudes_vacaciones WHERE id_solicitud = p_id_solicitud AND estado_solicitud = 'pendiente') THEN
        -- Actualizar el estado a 'rechazada'
        UPDATE solicitudes_vacaciones
        SET estado_solicitud = 'rechazada'
        WHERE id_solicitud = p_id_solicitud;

        -- Registrar la modificación en modificaciones_estado_vacaciones
        INSERT INTO modificaciones_estado_vacaciones (id_solicitud, id_empleado_modificador, estado_anterior, estado_nuevo, motivo)
        VALUES (p_id_solicitud, p_id_empleado_modificador, 'pendiente', 'rechazada', p_motivo);

        -- Confirmación del procedimiento
        SELECT 'Solicitud de vacaciones denegada correctamente' AS mensaje;
    ELSE
        SELECT 'Error: La solicitud no existe o ya fue procesada' AS mensaje;
    END IF;
END;
//

DELIMITER ;
```

·Procedimiento para modificar empleado:

```
DELIMITER //

CREATE PROCEDURE modificar_empleado(
    IN p_id_empleado INT,
    IN p_nombre VARCHAR(100),
    IN p_apellido VARCHAR(100),
    IN p_dni VARCHAR(20),
    IN p_id_departamento INT,
    IN p_id_turno INT,
    IN p_puesto VARCHAR(50),
    IN p_telefono VARCHAR(15),
    IN p_uid_ldap VARCHAR(100),
    IN p_cuenta_pidgin VARCHAR(100),
    IN p_fecha_contratacion DATE
)
BEGIN
    -- Verificar si el empleado existe antes de modificarlo
    IF EXISTS (SELECT 1 FROM empleados WHERE id_empleado = p_id_empleado) THEN
        UPDATE empleados
        SET nombre = p_nombre,
            apellido = p_apellido,
            dni = p_dni,
            id_departamento = p_id_departamento,
            id_turno = p_id_turno,
            puesto = p_puesto,
            telefono = p_telefono,
            uid_ldap = p_uid_ldap,
            cuenta_pidgin = p_cuenta_pidgin,
            fecha_contratacion = p_fecha_contratacion
        WHERE id_empleado = p_id_empleado;
```

```
        SELECT 'Empleado modificado correctamente' AS mensaje;
    ELSE
        SELECT 'Error: El empleado no existe' AS mensaje;
    END IF;
END;
//

DELIMITER ;
```

·Procedimiento para modificar departamentos:

```
DELIMITER //

CREATE PROCEDURE modificar_departamento(
    IN p_id_departamento INT,
    IN p_nombre VARCHAR(100),
    IN p_id_responsable INT
)
BEGIN
    -- Verificar si el departamento existe
    IF EXISTS (SELECT 1 FROM departamentos WHERE id_departamento = p_id_departamento) THEN

        -- Verificar que el nuevo nombre no esté repetido en otro departamento
        IF EXISTS (SELECT 1 FROM departamentos WHERE nombre = p_nombre AND id_departamento <> p_id_departamento) THEN
            SELECT 'Error: Ya existe otro departamento con ese nombre' AS mensaje;
        ELSE
            -- Actualizar el departamento
            UPDATE departamentos
            SET nombre = p_nombre,
                id_responsable = p_id_responsable
            WHERE id_departamento = p_id_departamento;
```

```
        SELECT 'Departamento modificado correctamente' AS mensaje;
    END IF;

  ELSE
    SELECT 'Error: El departamento no existe' AS mensaje;
  END IF;
END;
//

DELIMITER ;
·Procedimiento para modificar turnos:
DELIMITER //

CREATE PROCEDURE modificar_turno(
  IN p_id_turno INT,
  IN p_descripcion VARCHAR(50),
  IN p_hora_inicio TIME,
  IN p_hora_fin TIME,
  IN p_dias_semana SET('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo'),
  IN p_horario_flexible BOOLEAN
)
BEGIN
  -- Verificar si el turno existe
  IF EXISTS (SELECT 1 FROM turnos WHERE id_turno = p_id_turno) THEN

    -- Validar que la hora de inicio sea menor que la de fin
    IF p_hora_inicio >= p_hora_fin THEN
      SELECT 'Error: La hora de inicio no puede ser mayor o igual a la hora de fin' AS mensaje;
    ELSE
      -- Actualizar el turno
      UPDATE turnos
```

```
            SET descripcion = p_descripcion,
                hora_inicio = p_hora_inicio,
                hora_fin = p_hora_fin,
                dias_semana = p_dias_semana,
                horario_flexible = p_horario_flexible
            WHERE id_turno = p_id_turno;

            SELECT 'Turno modificado correctamente' AS mensaje;
        END IF;

    ELSE
        SELECT 'Error: El turno no existe' AS mensaje;
    END IF;
END;
//

DELIMITER ;
·Procedimiento para modificar fichajes:
DELIMITER //

CREATE PROCEDURE modificar_fichaje(
    IN p_id_fichaje INT,
    IN p_fecha DATE,
    IN p_hora TIME,
    IN p_tipo ENUM('entrada', 'salida')
)
BEGIN
    -- Verificar si el fichaje existe antes de modificarlo
    IF EXISTS (SELECT 1 FROM fichajes WHERE id_fichaje = p_id_fichaje) THEN

        -- Actualizar el fichaje
        UPDATE fichajes
```

```
        SET fecha = p_fecha,
            hora = p_hora,
            tipo = p_tipo,
            fichaje_modificado = TRUE
        WHERE id_fichaje = p_id_fichaje;

        SELECT 'Fichaje modificado correctamente' AS mensaje;

    ELSE
        SELECT 'Error: El fichaje no existe' AS mensaje;
    END IF;
END;
//

DELIMITER ;
·Procedimiento para insertar ausencias:
DELIMITER //

CREATE PROCEDURE insertar_ausencia(
    IN p_id_empleado INT,
    IN p_fecha_inicio DATE,
    IN p_fecha_fin DATE,
    IN p_motivo TEXT,
    IN p_justificada BOOLEAN
)
BEGIN
    DECLARE empleado_existe INT DEFAULT 0;

    -- Validar si el empleado existe
    SELECT COUNT(*) INTO empleado_existe FROM empleados WHERE id_empleado =
p_id_empleado;
```

```
    IF empleado_existe > 0 THEN
        -- Insertar ausencia en la tabla
        INSERT INTO ausencias (id_empleado, fecha_inicio, fecha_fin, motivo, justificada)
        VALUES (p_id_empleado, p_fecha_inicio, p_fecha_fin, p_motivo, p_justificada);
    ELSE
        -- Lanzar error si el empleado no existe
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: El empleado no existe.';
    END IF;
END //

DELIMITER ;
```

·Procedimiento para consultar fichajes:

```
DELIMITER //

CREATE PROCEDURE consultar_fichajes(IN p_id_empleado VARCHAR(10))
BEGIN
    IF p_id_empleado = '*' THEN
        -- Mostrar todos los fichajes
        SELECT id_fichaje, id_empleado, fecha, hora, tipo, fichaje_modificado
        FROM fichajes
        ORDER BY fecha DESC, hora DESC;
    ELSE
        -- Mostrar fichajes de un solo empleado
        SELECT id_fichaje, id_empleado, fecha, hora, tipo, fichaje_modificado
        FROM fichajes
        WHERE id_empleado = CAST(p_id_empleado AS UNSIGNED)
        ORDER BY fecha DESC, hora DESC;
    END IF;
END //

DELIMITER ;
```

Procedimiento para consultar ausencias:

```
DELIMITER //

CREATE PROCEDURE consultar_ausencias(IN p_id_empleado VARCHAR(10))
BEGIN
    IF p_id_empleado = '*' THEN
        -- Mostrar todas las ausencias de todos los empleados
        SELECT id_ausencia, id_empleado, fecha_inicio, fecha_fin, motivo, justificada
        FROM ausencias
        ORDER BY fecha_inicio DESC;
    ELSE
        -- Mostrar ausencias de un solo empleado
        SELECT id_ausencia, id_empleado, fecha_inicio, fecha_fin, motivo, justificada
        FROM ausencias
        WHERE id_empleado = CAST(p_id_empleado AS UNSIGNED)
        ORDER BY fecha_inicio DESC;
    END IF;
END //

DELIMITER ;
```

·Procedimiento para consultar solicitudes de vacaciones:

```
DELIMITER //


CREATE or replace PROCEDURE consultar_solicitudes_vacaciones(IN p_id_empleado
VARCHAR(10))
BEGIN
    IF p_id_empleado = '*' THEN
        --  Mostrar todas las solicitudes de vacaciones con el modificador
        SELECT id_solicitud, id_empleado, fecha_inicio, fecha_fin, estado_solicitud,
id_empleado_modificador
```

```
        FROM solicitudes_vacaciones
        ORDER BY fecha_inicio DESC;
    ELSE
        -- Mostrar solicitudes de un solo empleado con el modificador
        SELECT id_solicitud, id_empleado, fecha_inicio, fecha_fin, estado_solicitud,
id_empleado_modificador
        FROM solicitudes_vacaciones
        WHERE id_empleado = CAST(p_id_empleado AS UNSIGNED)
        ORDER BY fecha_inicio DESC;
    END IF;
END //

DELIMITER ;
```

·Procedimiento para consultar datos personales:

```
DELIMITER //

CREATE PROCEDURE consultar_datos_personales(IN p_id_empleado VARCHAR(10))
BEGIN
    IF p_id_empleado = '*' THEN
        -- Mostrar todos los datos personales de todos los empleados
        SELECT id_empleado, nombre, apellido, dni, id_departamento, id_turno, puesto, telefono,
            uid_ldap, cuenta_pidgin, fecha_contratacion, dias_vacaciones_disponibles
        FROM empleados
        ORDER BY nombre ASC, apellido ASC;
    ELSE
        -- Mostrar datos personales de un solo empleado
        SELECT id_empleado, nombre, apellido, dni, id_departamento, id_turno, puesto, telefono,
            uid_ldap, cuenta_pidgin, fecha_contratacion, dias_vacaciones_disponibles
        FROM empleados
        WHERE id_empleado = CAST(p_id_empleado AS UNSIGNED);
    END IF;
END //
```

```
DELIMITER ;
```

·Procedimiento para consultar las modificaciones de fichajes:

```
DELIMITER //

CREATE PROCEDURE consultar_correcciones_fichajes(IN p_id_fichaje VARCHAR(10))
BEGIN
    IF p_id_fichaje = '*' THEN
        -- Mostrar todas las correcciones de fichajes
        SELECT id_correccion, id_fichaje, id_empleado_modificador, motivo, tipo_antiguo, tipo_nuevo,
            hora_antigua, hora_nueva, fecha_modificacion
        FROM correcciones_fichaje
        ORDER BY fecha_modificacion DESC;
    ELSE
        -- Mostrar correcciones de un solo fichaje
        SELECT id_correccion, id_fichaje, id_empleado_modificador, motivo, tipo_antiguo, tipo_nuevo,
            hora_antigua, hora_nueva, fecha_modificacion
        FROM correcciones_fichaje
        WHERE id_fichaje = CAST(p_id_fichaje AS UNSIGNED)
        ORDER BY fecha_modificacion DESC;
    END IF;
END //

DELIMITER ;
```

·Procedimiento para consultar incidencias de fichaje:

```
DELIMITER //

CREATE PROCEDURE consultar_incidencias_fichajes()
```

```
BEGIN
    -- Mostrar todas las incidencias de fichajes ordenadas por ID
    SELECT id_incidencia, id_fichaje, descripcion
    FROM incidencias_fichaje
    ORDER BY id_incidencia DESC;
END //

DELIMITER ;
```

·Procedimiento para consultar las modificaciones en el estado de las solicitudes de vacaciones:

```
DELIMITER //

CREATE PROCEDURE consultar_modificaciones_estado_vacaciones()
BEGIN
    -- Mostrar todas las modificaciones de estado de vacaciones
    SELECT id_modificacion, id_solicitud, id_empleado_modificador, estado_anterior,
estado_nuevo,
        motivo, fecha_modificacion
    FROM modificaciones_estado_vacaciones
    ORDER BY fecha_modificacion DESC;
END //

DELIMITER ;
```

·Procedimiento para consultar turnos:

```
DELIMITER //

CREATE PROCEDURE consultar_turnos()
BEGIN
    -- Mostrar todos los turnos ordenados por ID
    SELECT id_turno, descripcion, hora_inicio, hora_fin, dias_semana, horario_flexible
    FROM turnos
    ORDER BY id_turno ASC;
END //
```

```
DELIMITER ;
·Procedimiento para consultar departamentos:
DELIMITER //

CREATE PROCEDURE consultar_departamentos()
BEGIN
    -- Mostrar todos los departamentos ordenados por nombre
    SELECT id_departamento, nombre, id_responsable
    FROM departamentos
    ORDER BY nombre ASC;
END //

DELIMITER ;
·Procedimiento para justificar ausencias:
DELIMITER //

DROP PROCEDURE IF EXISTS justificar_ausencia;

CREATE PROCEDURE justificar_ausencia(
    IN p_id_ausencia INT,
    IN p_id_empleado_modificador INT
)
BEGIN
    --  Verificar si la ausencia existe y aún no está justificada
    IF EXISTS (SELECT 1 FROM ausencias WHERE id_ausencia = p_id_ausencia AND justificada
= FALSE) THEN
        --  Marcar la ausencia como justificada y registrar el modificador
        UPDATE ausencias
        SET justificada = TRUE, id_empleado_modificador = p_id_empleado_modificador
        WHERE id_ausencia = p_id_ausencia;
```

```
        -- Registrar la modificación en la tabla de historial
        INSERT INTO modificaciones_ausencias (id_ausencia, id_empleado_modificador, motivo)
        VALUES (p_id_ausencia, p_id_empleado_modificador, 'Ausencia justificada');

        -- Confirmación del procedimiento
        SELECT 'Ausencia justificada correctamente' AS mensaje;
    ELSE
        SELECT 'Error: La ausencia no existe o ya está justificada' AS mensaje;
    END IF;
END //

DELIMITER ;
·Procedimiento para borrar_fichaje:
DELIMITER //



CREATE or replace PROCEDURE borrar_fichaje(
    IN p_id_fichaje INT,
    IN p_id_empleado_modificador INT
)
BEGIN
    -- Verificar si el fichaje existe
    IF EXISTS (SELECT 1 FROM fichajes WHERE id_fichaje = p_id_fichaje) THEN
        -- Eliminar registros relacionados en `correcciones_fichaje`
        DELETE FROM correcciones_fichaje WHERE id_fichaje = p_id_fichaje;

        -- Registrar la eliminación en `correcciones_fichaje`
        INSERT INTO correcciones_fichaje (id_fichaje, id_empleado_modificador, motivo,
tipo_antiguo, tipo_nuevo, hora_antigua, hora_nueva)
        SELECT id_fichaje, p_id_empleado_modificador, 'Fichaje eliminado por RRHH', tipo, NULL,
hora, NULL
        FROM fichajes WHERE id_fichaje = p_id_fichaje;
```

```
    -- Eliminar el fichaje después de borrar los registros dependientes
    DELETE FROM fichajes WHERE id_fichaje = p_id_fichaje;

    -- Confirmación del procedimiento
    SELECT 'Fichaje eliminado correctamente' AS mensaje;
  ELSE
    SELECT 'Error: El fichaje no existe' AS mensaje;
  END IF;
END //

DELIMITER ;
```

Usuario “rrhh-db”

El usuario “rrhh-db” sera el encargado de gestionar los datos de las tablas, asi que tendrá los privilegios de conectarse a la DB, SELECT, INSERT, UPDATE y DELETE en todas las tablas de la base de datos “planetas_sa”.

Para ello desde el usuario “root” crearemos el usuario “rrhh-db” y le daremos privilegio de inicio de sesion en la BD “planetas_sa”:

CREATE USER 'rrhh-db'@'localhost' IDENTIFIED BY 'planetas-sa-2025-rrhh-db';

GRANT USAGE ON planetas_sa.* TO 'rrhh-db'@'localhost';

```
root@raspberrypi:/home/admin-raspberrypi# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.11.11-MariaDB-0ubuntu0.24.04.2 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'rrhh-db'@'localhost' IDENTIFIED BY 'planetas-sa-2025-rrhh-db';
Query OK, 0 rows affected (0.011 sec)

MariaDB [(none)]> GRANT USAGE ON planetas_sa.* TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.008 sec)
```

Y con el usuario “admin_db” le daremos privilegios sobre los objetos de la base de datos:

GRANT SELECT, INSERT, UPDATE, DELETE ON planetas_sa.* TO 'rrhh-db'@'localhost';

```
root@raspberrypi:/home/admin-raspberrypi# mysql -u ADMIN_DB -p'planetas-sa-2025-admin-db'
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 35
Server version: 10.11.11-MariaDB-0ubuntu0.24.04.2 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> GRANT SELECT, INSERT, UPDATE, DELETE ON planetas_sa.* TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.011 sec)
```

Y privilegio para ejecutar funciones desde root:

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_empleado TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_departamento TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_turno TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_empleado TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_departamento TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_turno TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_fichaje TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.aprobar_solicitud_vacaciones TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.denegar_solicitud_vacaciones TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_empleado TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_departamento TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_turno TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_fichajes TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_ausencias TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_solicitudes_vacaciones TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_datos_personales TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_correcciones_fichajes TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_incidencias_fichajes TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_modificaciones_estado_vacaciones TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_turnos TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_departamentos TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.justificar_ausencia TO 'rrhh-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_fichaje TO 'rrhh-db'@'localhost';

```
MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_empleado TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.015 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_departamento TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_turno TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_empleado TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_departamento TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_turno TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.modificar_fichaje TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.aprobar_solicitud_vacaciones TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.denegar_solicitud_vacaciones TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_empleado TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_departamento TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.borrar_turno TO 'rrhh-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)
```

Usuario “user-db”

El usuario “user-db” será el utilizado por los empleados en la aplicación web para registrar sus fichajes y consultar lus datos en las tablas, por lo que tendra el privilegio de conectarse a la BD, SELECT en todas las tablas de la base de datos “planetas_sa” excepto en correcciones_fichaje y en incidencias_fichaje, y además el privilegio INSERT en la tabla fichajes

Para ello desde el usuario “root” crearemos el usuario “user-db” y le daremos privilegio de inicio de sesion en la BD “planetas_sa”:

CREATE USER 'user-db'@'localhost' IDENTIFIED BY 'planetas-sa-2025-user-db';

GRANT USAGE ON planetas_sa.* TO 'user-db'@'localhost';

```
root@raspberrypi:/home/admin-raspberrypi# sudo mysql -u root -p'planetas-sa-2025-db'
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.11.11-MariaDB-0ubuntu0.24.04.2 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'user-db'@'localhost' IDENTIFIED BY 'planetas-sa-2025-user-db';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT USAGE ON planetas_sa.* TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.008 sec)
```

Y con el usuario “admin_db” le daremos privilegios sobre los objetos de la base de datos:

GRANT SELECT ON planetas_sa.empleados TO 'user-db'@'localhost';

GRANT SELECT ON planetas_sa.departamentos TO 'user-db'@'localhost';

GRANT SELECT ON planetas_sa.turnos TO 'user-db'@'localhost';

GRANT SELECT, INSERT ON planetas_sa.fichajes TO 'user-db'@'localhost';

GRANT SELECT, INSERT ON planetas_sa.ausencias TO 'user-db'@'localhost';

GRANT SELECT, INSERT ON planetas_sa.solicitudes_vacaciones TO 'user-db'@'localhost';

```
root@raspberrypi:/home/admin-raspberrypi# mysql -u ADMIN_DB -p'planetas-sa-2025-admin-db'
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.11.11-MariaDB-0ubuntu0.24.04.2 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> GRANT SELECT ON planetas_sa.empleados TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.013 sec)

MariaDB [(none)]> GRANT SELECT ON planetas_sa.departamentos TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT SELECT ON planetas_sa.turnos TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT SELECT, INSERT ON planetas_sa.fichajes TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> GRANT SELECT, INSERT ON planetas_sa.ausencias TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT SELECT, INSERT ON planetas_sa.solicitudes_vacaciones TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.008 sec)
```

Y privilegio para ejecutar procedimientos:

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_fichaje TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_solicitud_vacaciones TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_ausencia TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_fichajes TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_ausencias TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_solicitudes_vacaciones TO 'user-db'@'localhost';

GRANT EXECUTE ON PROCEDURE planetas_sa.consultar_datos_personales TO 'user-db'@'localhost';

```
MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_fichaje TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.008 sec)

MariaDB [(none)]> GRANT EXECUTE ON PROCEDURE planetas_sa.insertar_solicitud_vacaciones TO 'user-db'@'localhost';
Query OK, 0 rows affected (0.003 sec)
```