

**PROYECTO  
FINAL**

**BASES DE  
DATOS**

# ¿QUIÉNES SOMOS?



Nuestro equipo está hecho de ingenieros fuertes y apasionados, comprometidos en ofrecer la solución más óptima hacia nuestros clientes.

# Nuestro equipo



**David Tavera**

Project Manager



**Josué Jiménez**

Desarrollador Frontend



**David Nava**

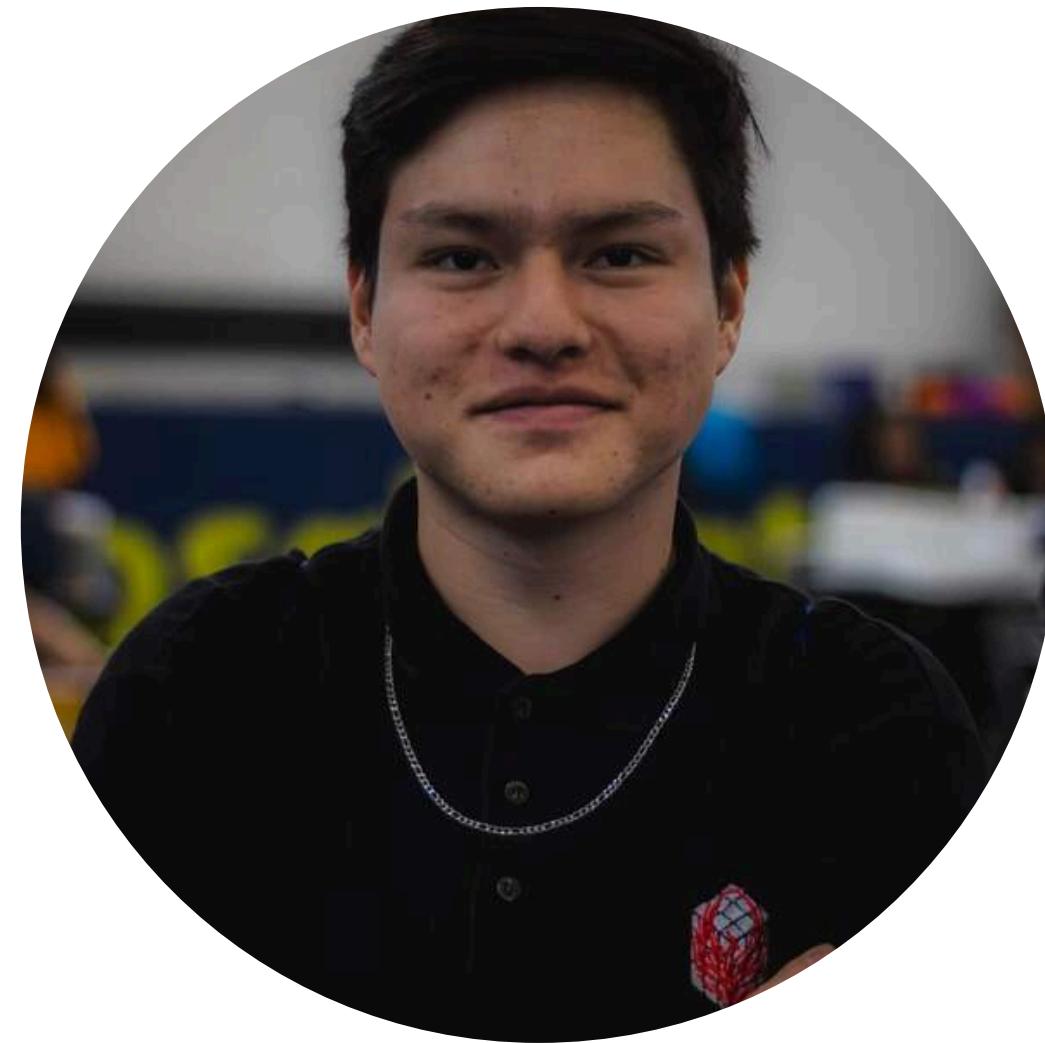
Analista de  
Requerimientos

# Nuestro equipo



Iván Rodríguez

Desarrollador Backend

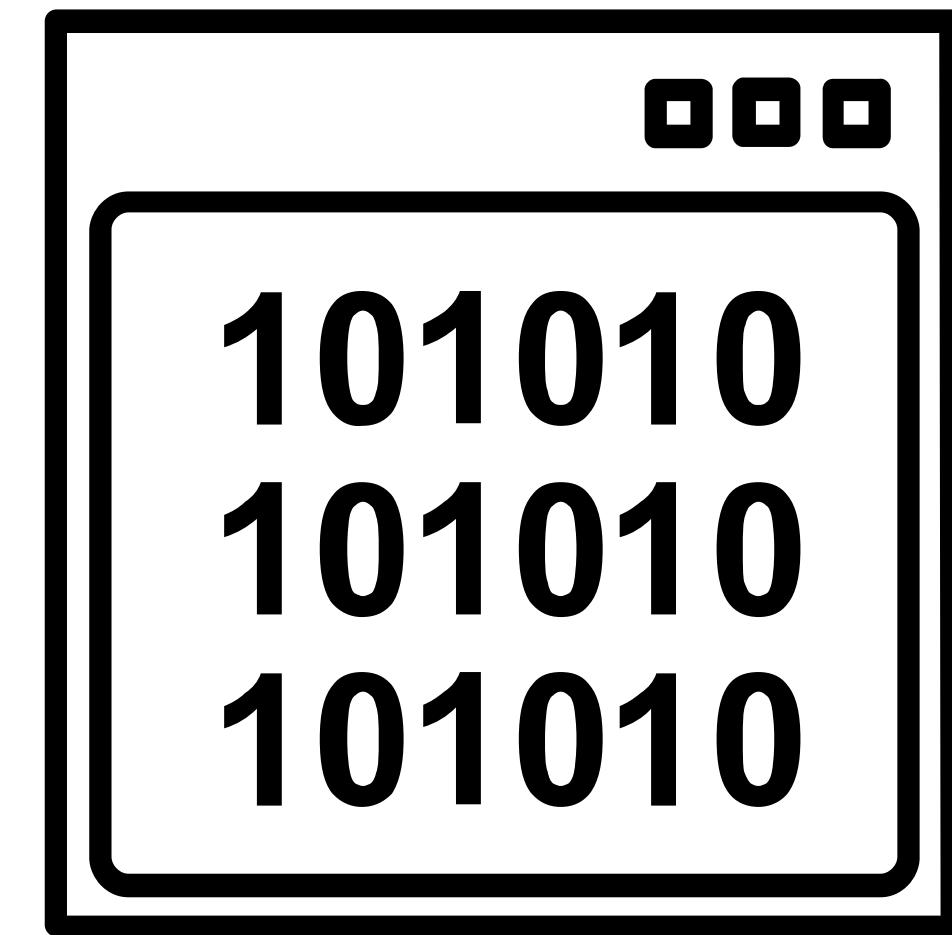
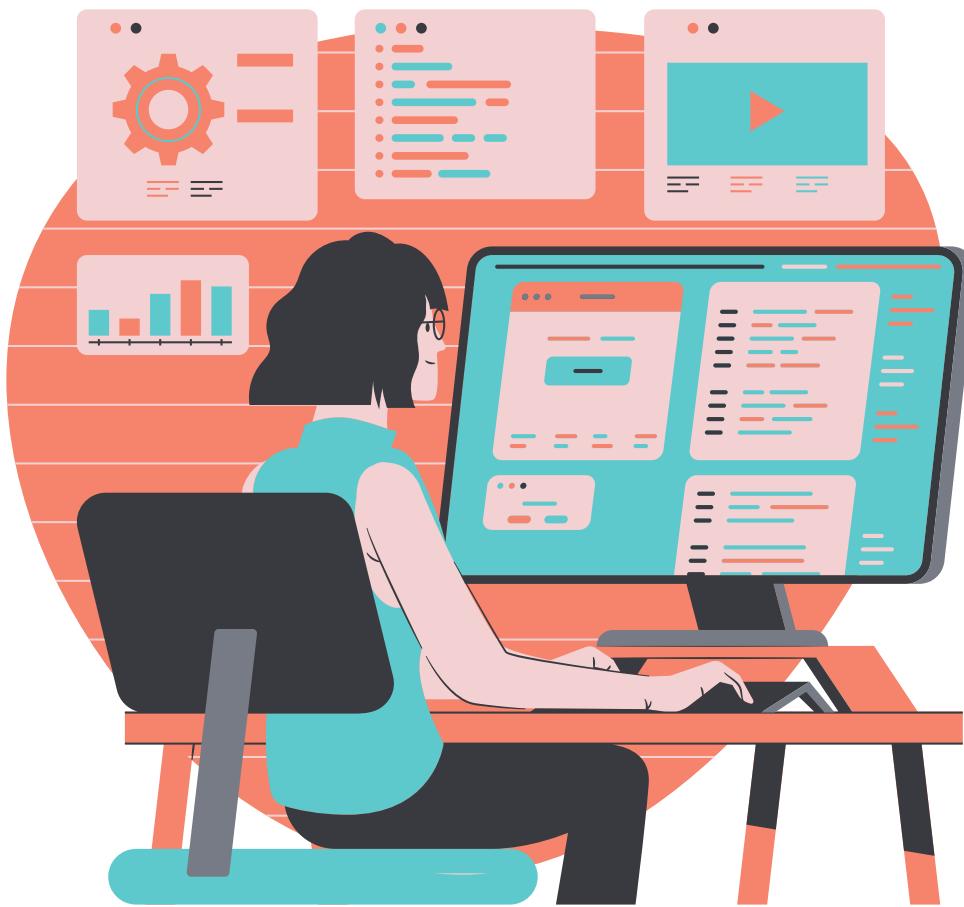


Santiago Medina

Tester

# INTRODUCCIÓN

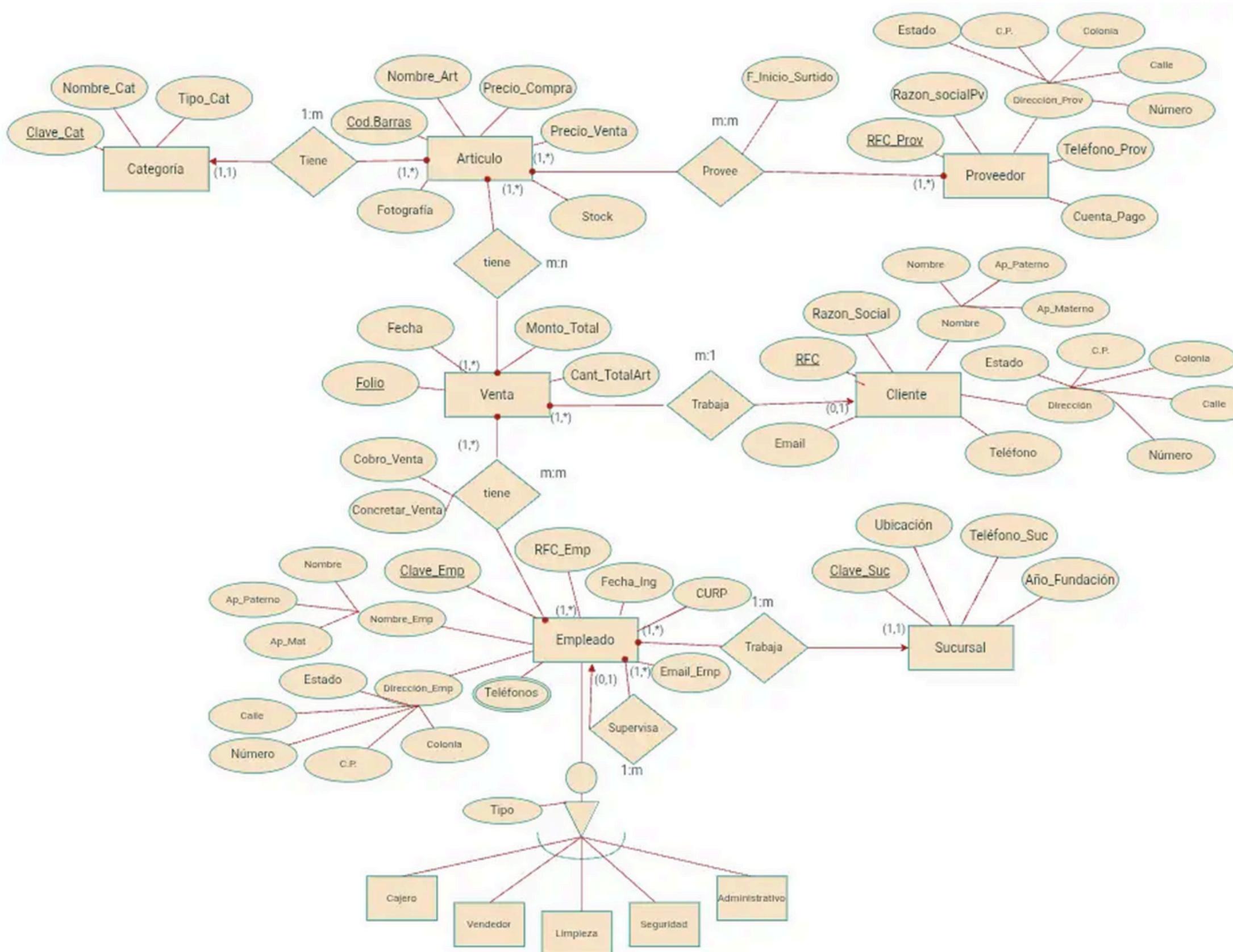
Se plantea diseñar e implementar una base de datos relacional en PostgreSQL que permita almacenar, gestionar y consultar de manera eficiente la información crítica del negocio, asegurando su integridad, consistencia y disponibilidad.



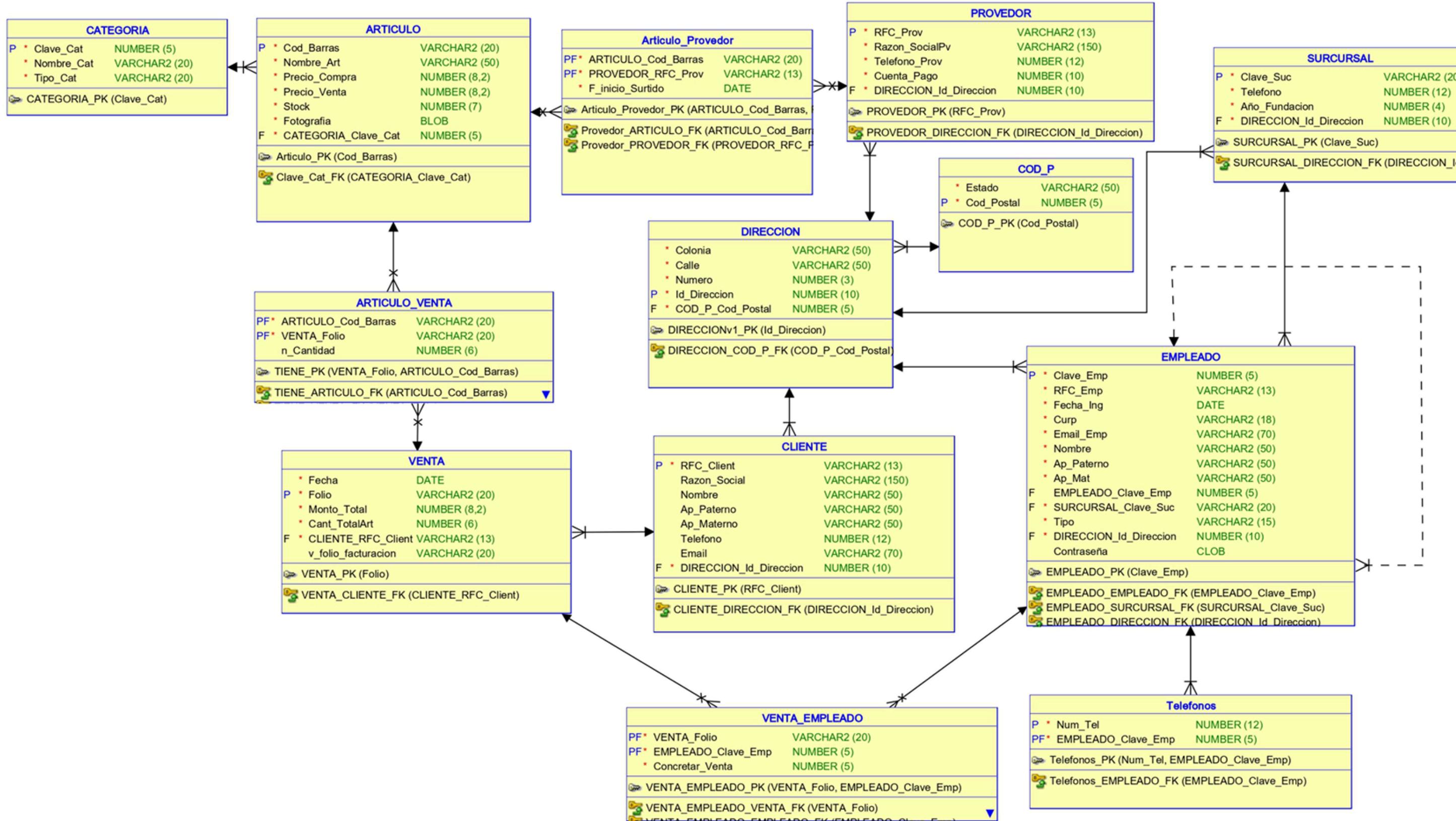
# **ANÁLISIS DE REQUERIMIENTOS**

Tuvo como objetivo identificar las entidades, atributos y relaciones clave a partir de la descripción funcional proporcionada para la digitalización de los procesos de una mueblería.

# DISEÑO CONCEPTUAL



# DISEÑO LÓGICO



# DISEÑO FÍSICO: TABLAS

Tabla CLIENTE:

```
CREATE TABLE IF NOT EXISTS public.cliente
(
    v_rfc_client character varying(13) COLLATE
        pg_catalog."default" NOT NULL,
    v_razon_social character varying(150) COLLATE
        pg_catalog."default",
    v_nombre character varying(50) COLLATE pg_catalog."
        default",
    v_ap_paterno character varying(50) COLLATE
        pg_catalog."default",
    v_ap_materno character varying(50) COLLATE
        pg_catalog."default",
    n_telefono numeric(12,0),
    v_email character varying(70) COLLATE pg_catalog."
        default",
    n_direccion_id_direccion numeric(10,0),
    "t_contrasena" text COLLATE pg_catalog."default",
    CONSTRAINT cliente_pk PRIMARY KEY (v_rfc_client),
    CONSTRAINT cliente_direccion_fk FOREIGN KEY (
        n_direccion_id_direccion)
        REFERENCES public.direccion (n_id_direccion)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

# DISEÑO FÍSICO: ÍNDICES

```
CREATE INDEX idx_articulo_clave_cat  
ON articulo (n_categoria_clave_cat);
```

# DISEÑO FÍSICO: FUNCIONES

## FUNCION DE VALIDACION EMPLEADO

```
CREATE OR REPLACE FUNCTION public.sp_inicio_sesion(
    p_n_clave_emp numeric,
    "p_contrasena" text)
RETURNS TABLE(validacion integer, n_clave_emp
    numeric, v_rfc_emp character varying, d_fecha_ing
    date, v_curp character varying, v_email_emp
    character varying, v_nombre character varying,
    v_ap_paterno character varying, v_ap_mat
    character varying, n_empleado_clave_emp numeric,
    v_sucursal_clave_suc character varying, v_tipo
    character varying, n_direccion_id_direccion
    numeric)
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        1 AS validacion,
        e.n_clave_emp,
        e.v_rfc_emp,
        e.d_fecha_ing,
        e.v_curp,
        e.v_email_emp,
        e.v_nombre,
        e.v_ap_paterno,
        e.v_ap_mat,
        e.n_empleado_clave_emp,
        e.v_sucursal_clave_suc,
        e.v_tipo,
        e.n_direccion_id_direccion
```

## FUNCION VALIDAR CAJERO

```
CREATE OR REPLACE FUNCTION public.
    sp_validar_vendedor_cajero_misma_sucursal(
        p_emp1 numeric,
        p_emp2 numeric)
RETURNS TABLE(validacion integer, mensaje text)
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
ROWS 1000

AS $BODY$
DECLARE
    emp1 RECORD;
    emp2 RECORD;
BEGIN
    SELECT v_tipo, v_sucursal_clave_suc INTO emp1
    FROM empleado
    WHERE n_clave_emp = p_emp1;

    IF NOT FOUND THEN
        RETURN QUERY SELECT 0, format('Empleado %s no
            existe', p_emp1);
        RETURN;
    END IF;

    SELECT v_tipo, v_sucursal_clave_suc INTO emp2
    FROM empleado
    WHERE n_clave_emp = p_emp2;

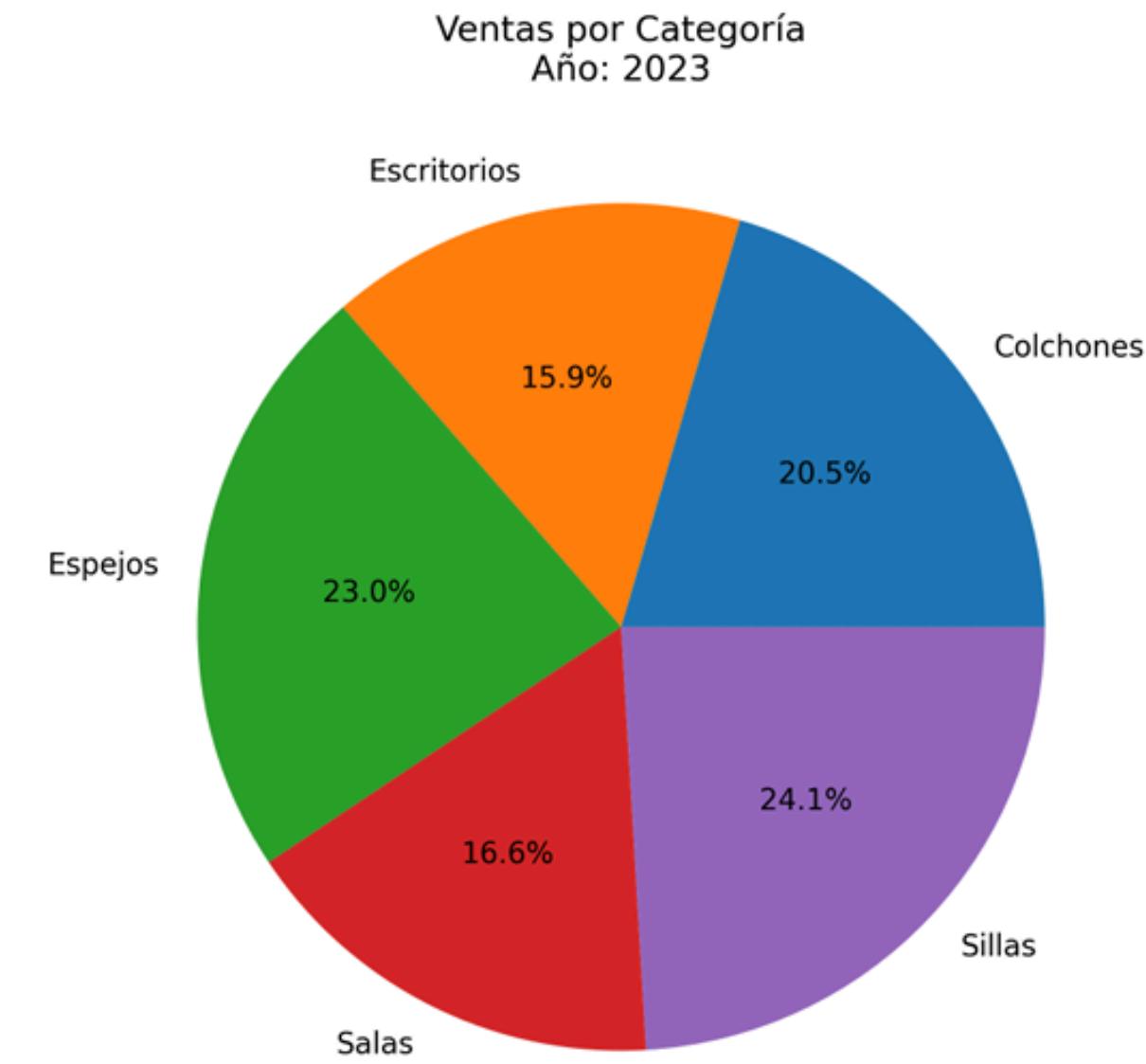
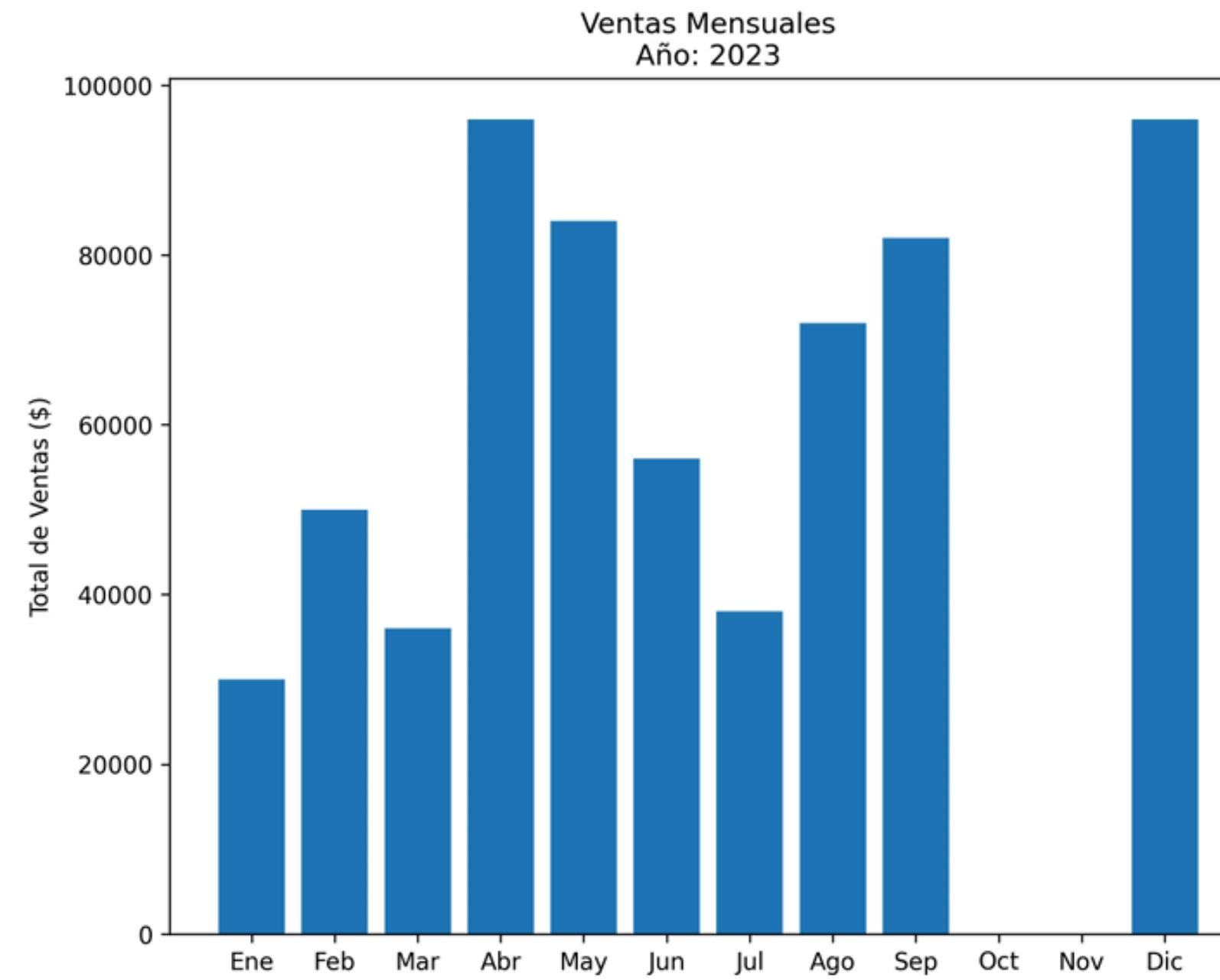
    IF NOT FOUND THEN
        RETURN QUERY SELECT 0, format('Empleado %s no
            existe', p_emp2);
        RETURN;
    END IF;

    IF (
        (LOWER(emp1.v_tipo) = 'vendedor' AND LOWER(emp2.
            v_tipo) = 'cajero') OR
```

# DISEÑO FÍSICO: TRIGGERS

```
CREATE OR REPLACE FUNCTION public.sp_set_razon_social()
RETURNS trigger
LANGUAGE 'plpgsql'
COST 100
VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    -- Solo si razon_social esta vacio o NULL
    IF NEW.v_razon_social IS NULL OR trim(NEW.
        v_razon_social) = '' THEN
        NEW.v_razon_social := trim(concat_ws(' ', NEW.
            v_nombre, NEW.v_ap_paterno, NEW.v_ap_materno))
    );
END IF;
RETURN NEW;
END;
$BODY$;
```

# DASHBOARD



# CONCLUSIONES

Todo el equipo en conjunto desarrolló exitosamente el proyecto: desde el análisis y diseño del MER, MR, hasta la implementación en PostgreSQL, validación mediante pruebas y documentación final. La solución presentada nos otorga una base de datos estructurada, funcional y confiable.

**¡GRACIAS!**