****

Universidad Tecnológica Metropolitana

Base de Datos Avanzado

5to A

Unidad 1: Extracción de datos DQL

Actividad: Entregables Proyecto Integrador

Nombre del alumno(s):

Acosta Tzab José Bernardo

Ancona Alejo Javier Roberto

Cabrera Flota Martin Emanuel

Peña Herrera Bogarth Alejandro

Tun Cauich Brandon Andréi

Nombre del Profesor(a): Canche Caamal Mirian Magaly

Fecha de Entrega: miércoles 02 de febrero de 2026

**Proyecto Integrador Mubclean**

**Objetivo**

El objetivo de MubClean es desarrollar una plataforma digital que automatice la cotización, agendamiento y gestión de servicios de limpieza de muebles a domicilio, reduciendo la informalidad y los tiempos de atención mediante procesos claros y centralizados. Busca permitir a los clientes cotizar, agendar y pagar servicios de forma rápida y segura, mientras facilita a las empresas la administración de pedidos, agenda, técnicos y pagos desde un solo sistema. Asimismo, MubClean mejora la eficiencia operativa y la toma de decisiones a través de información y métricas en tiempo real, incrementando la confianza del usuario al ofrecer transparencia, seguimiento continuo y evidencia del servicio prestado.

Alcance (descripción detallada de los módulos o partes que conforman el proyecto, mencionar las tecnologías y herramientas a utilizar para el desarrollo de la aplicación web y/o móvil tanto en frontend como backend)

MubClean se estructura en tres módulos funcionales: Cotización, Agendamiento y Seguimiento/Cierre, los cuales cubren el ciclo completo del servicio.

En Cotización, el cliente registra una solicitud y los ítems a limpiar, apoyándose en las tablas solicitudes, items\_solicitud, fotos\_solicitud y el catálogo de servicios en servicios\_catalogo; aquí se concentran la evidencia inicial, el desglose y el cálculo del precio\_total.

En Agendamiento, se define la programación final del servicio y la asignación del técnico, gestionada principalmente mediante los campos de solicitudes (por ejemplo, fecha\_agendada\_final, hora\_agendada\_final, tecnico\_asignado\_id) y la relación de personal por negocio en empleados\_negocio, vinculada a los usuarios en perfiles para controlar disponibilidad y operación por empresa.

En Seguimiento/Cierre, se registra el avance y la conclusión del servicio, almacenando evidencia final en evidencia\_final, cambios de estado y fechas en solicitudes, y trazabilidad de automatizaciones en eventos\_n8n; adicionalmente, la administración SaaS se soporta en negocios, perfiles y el módulo de atención mediante soporte\_tickets.

En cuanto a tecnologías, el frontend móvil y administrativo se desarrolla con Flutter (Dart), el landing page con HTML y CSS, la base de datos y autenticación se gestionan con Supabase (PostgreSQL), y se integra un backend con Node.js y Express para exponer lógica adicional e incorporar la pasarela de pago de Mercado Pago (almacenando sus referencias en la solicitud cuando corresponde).

**Beneficios**

***Beneficios para empresas***

* Automatiza cotizaciones y reduce el tiempo invertido en atender mensajes y llamadas.
* Centraliza la operación (pedidos, agenda, técnicos, evidencias y pagos) en un solo sistema.
* Disminuye errores y cancelaciones por desorganización al manejar disponibilidad y estados del servicio.
* Mejora la toma de decisiones con historial, métricas e indicadores de servicios e ingresos.
* Aumenta la confianza del cliente al ofrecer evidencia del trabajo, trazabilidad y atención mediante soporte.

Beneficios para clientes

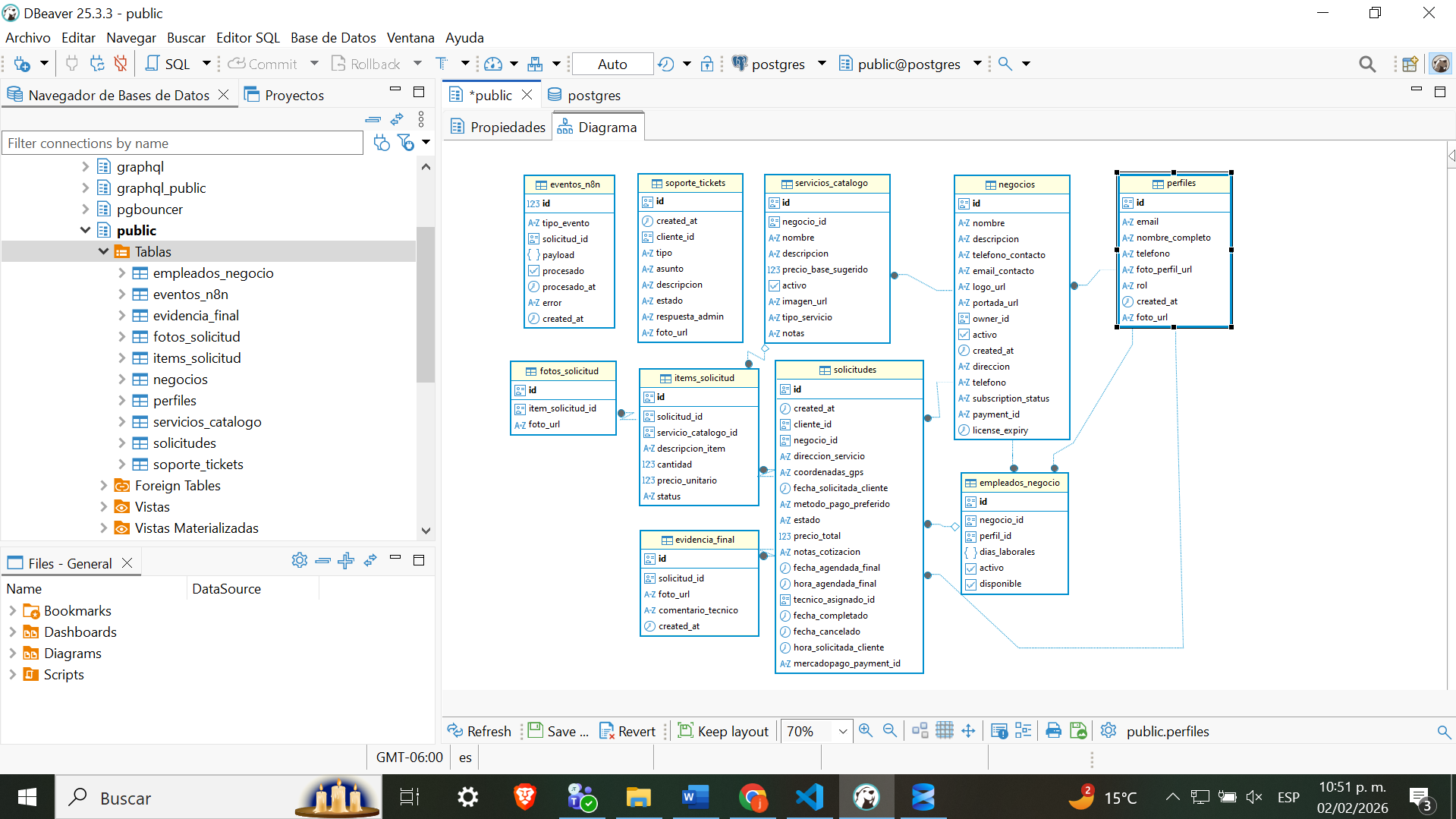
* Cotización rápida y clara sin depender de WhatsApp ni esperar respuesta manual.
* Agendamiento más sencillo con disponibilidad y horarios definidos.
* Mayor confianza al ver perfiles del negocio/técnicos, evidencias y opiniones/valoraciones.
* Pagos más accesibles (en línea o contra entrega según el caso) y comprobantes del servicio.
* Seguimiento del servicio con estados y evidencia final, reduciendo incertidumbre y conflictos.

**Modelo Entidad-Relacional**

Diagrama

El contenido generado por IA puede ser incorrecto.

**Modelo Relacional**



**Identificación del uso de Restricciones PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, y DEFAULT. Incluir dos casos de prueba de cada uno.**

***Tabla de Identificación de Restricciones***

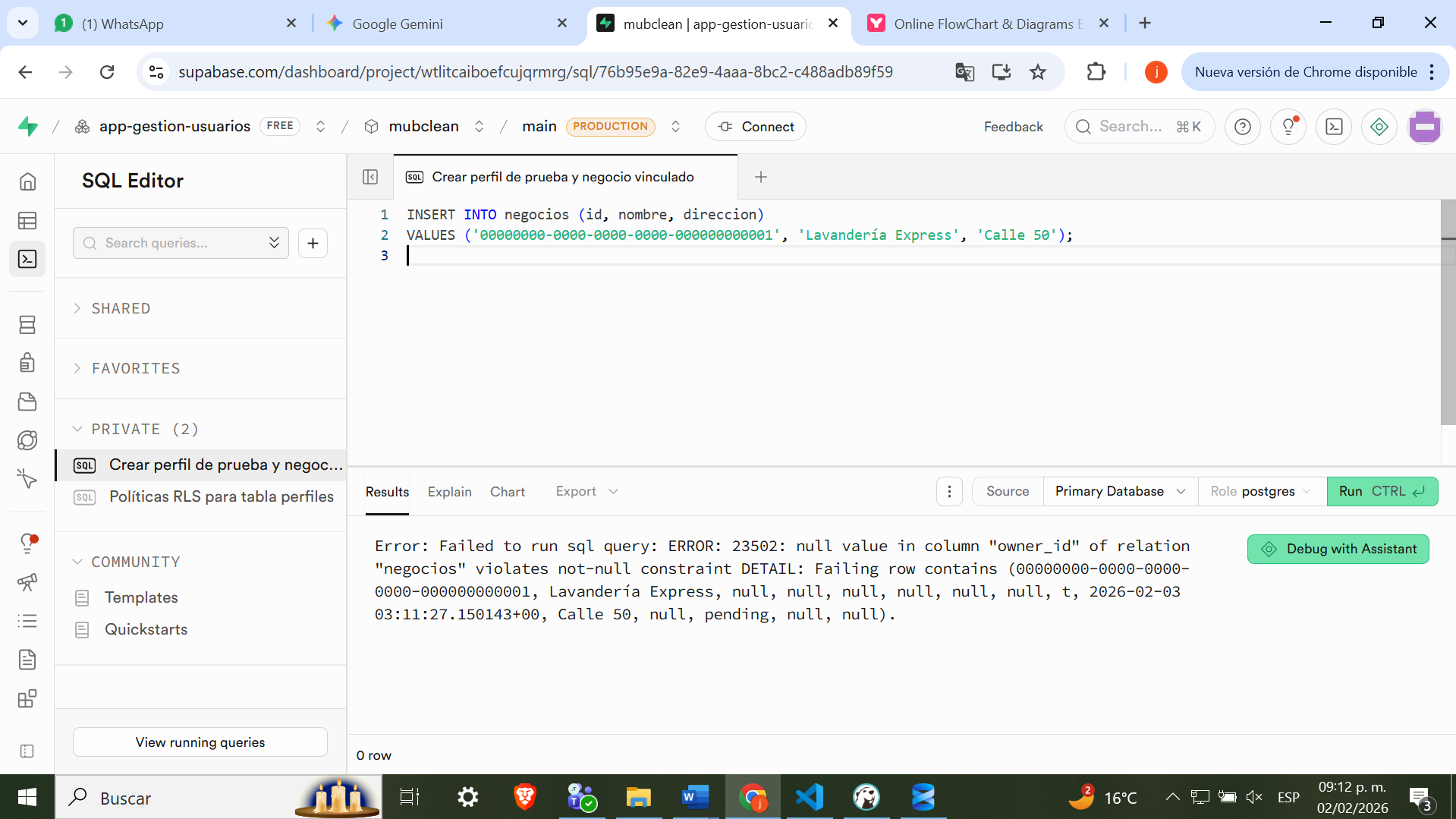
Esta tabla describe las reglas que configuraron en **PostgreSQL/Supabase**.

| **Tabla** | **Tipo de Restricción** | **Columna** | **Descripción / Regla** |
| --- | --- | --- | --- |
| negocios | **Primary Key (PK)** | id | Identificador único universal (UUID) del negocio. |
| perfiles | **Unique (UNQ)** | email | No pueden existir dos usuarios con el mismo correo. |
| servicios\_catalogo | **Check (CHK)** | precio\_base\_sugerido | El precio debe ser mayor o igual a 0 para evitar errores contables. |
| solicitudes | **Foreign Key (FK)** | negocio\_id | Vincula la solicitud a un negocio existente; si el negocio se borra, la solicitud falla. |
| solicitudes | **Default (DFT)** | estado | Toda solicitud nueva inicia automáticamente en estado 'pendiente'. |

***Restricción de Integridad Referencial (NOT NULL)***

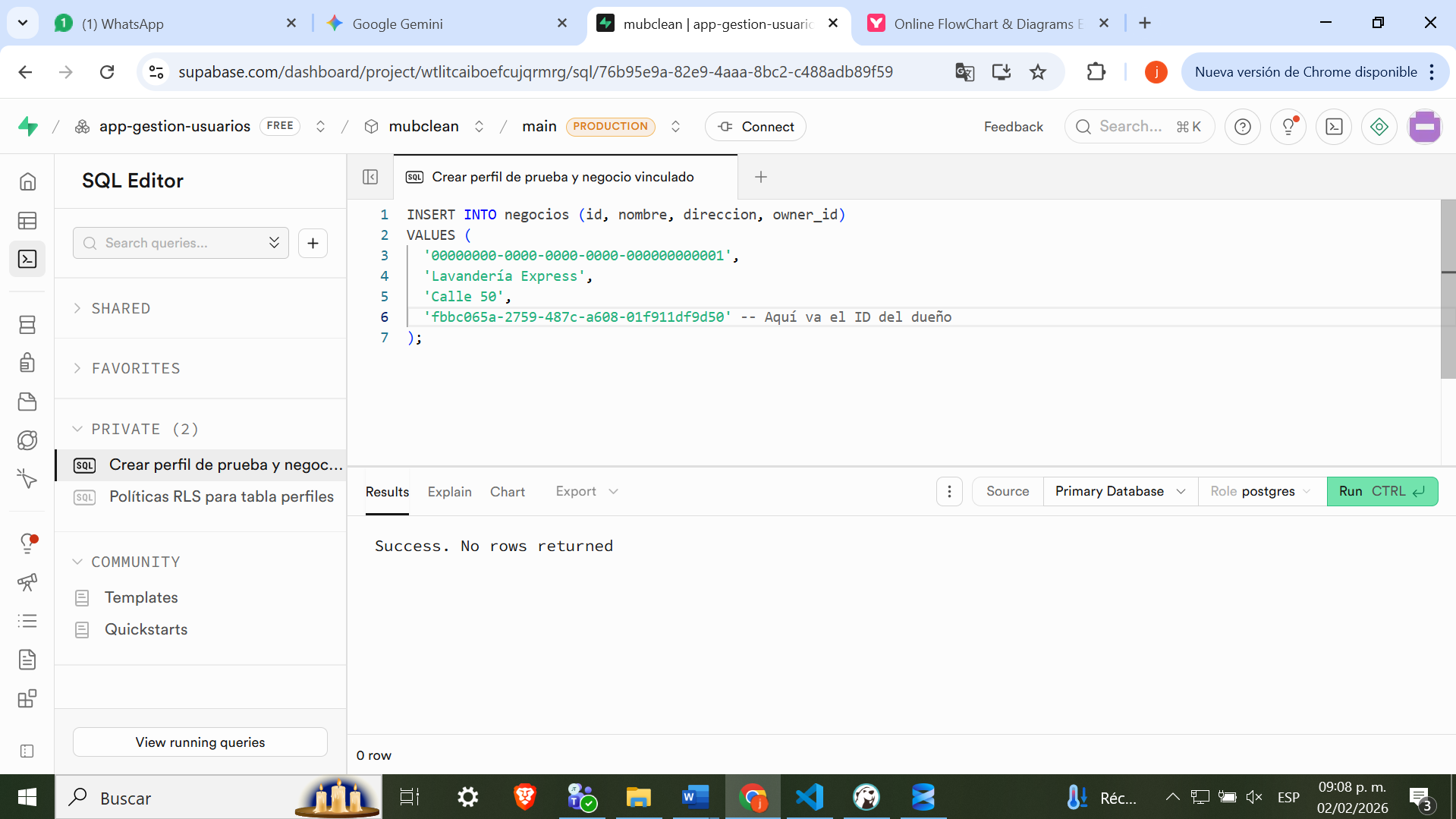
**Descripción:** Se valida que el sistema impida la creación de un negocio sin un dueño (perfil) asociado.

**Caso de Falla:** Se intentó insertar un negocio dejando el campo owner\_id vacío.



**Resultado:** El SGBD arrojó el error 23502 (null value violates not-null constraint), bloqueando la operación.

**Caso de Éxito:** Se insertó el registro incluyendo un UUID de perfil válido.

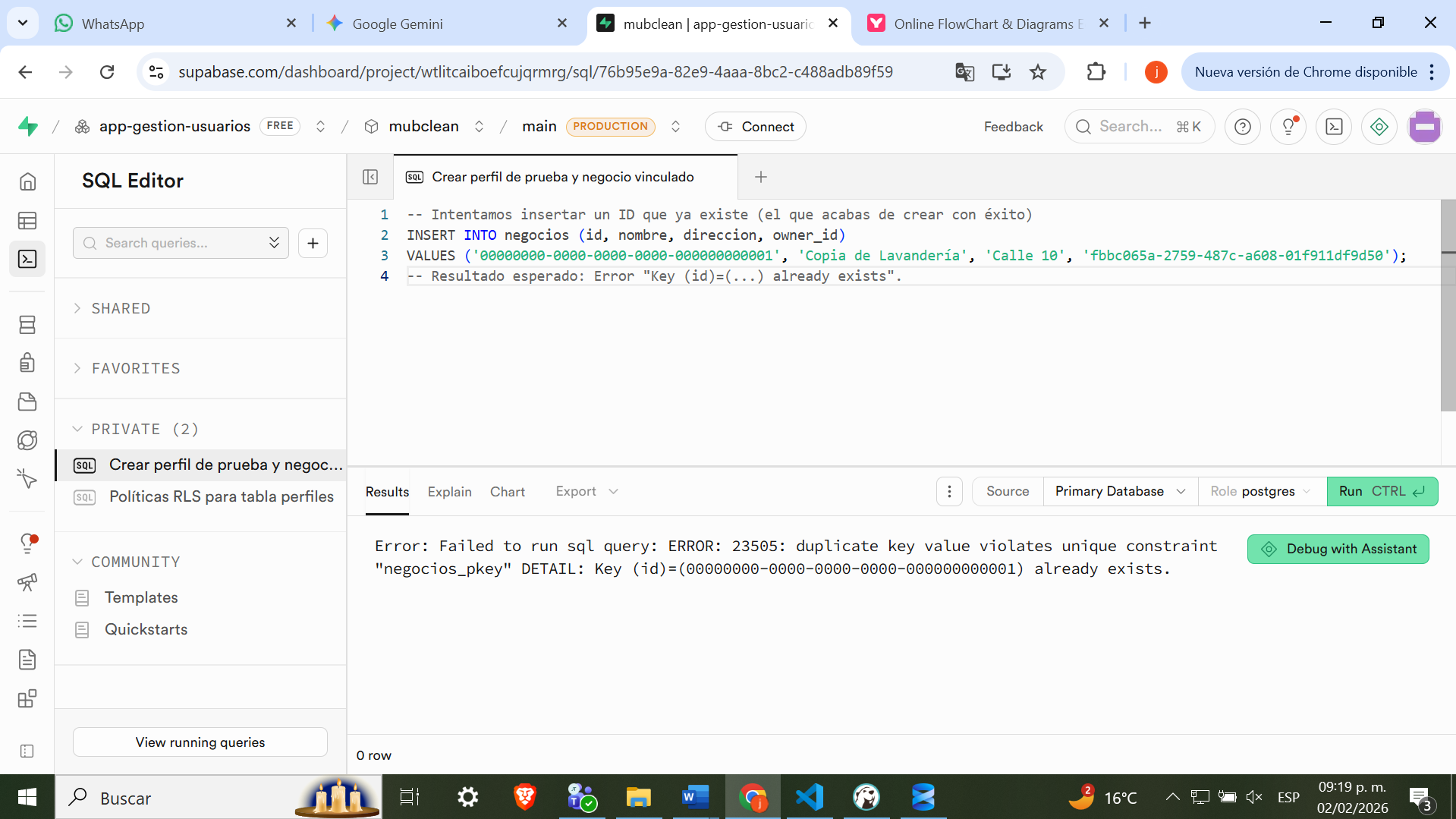


**Resultado:** El registro se creó correctamente, vinculando el negocio al usuario fbbc065a....

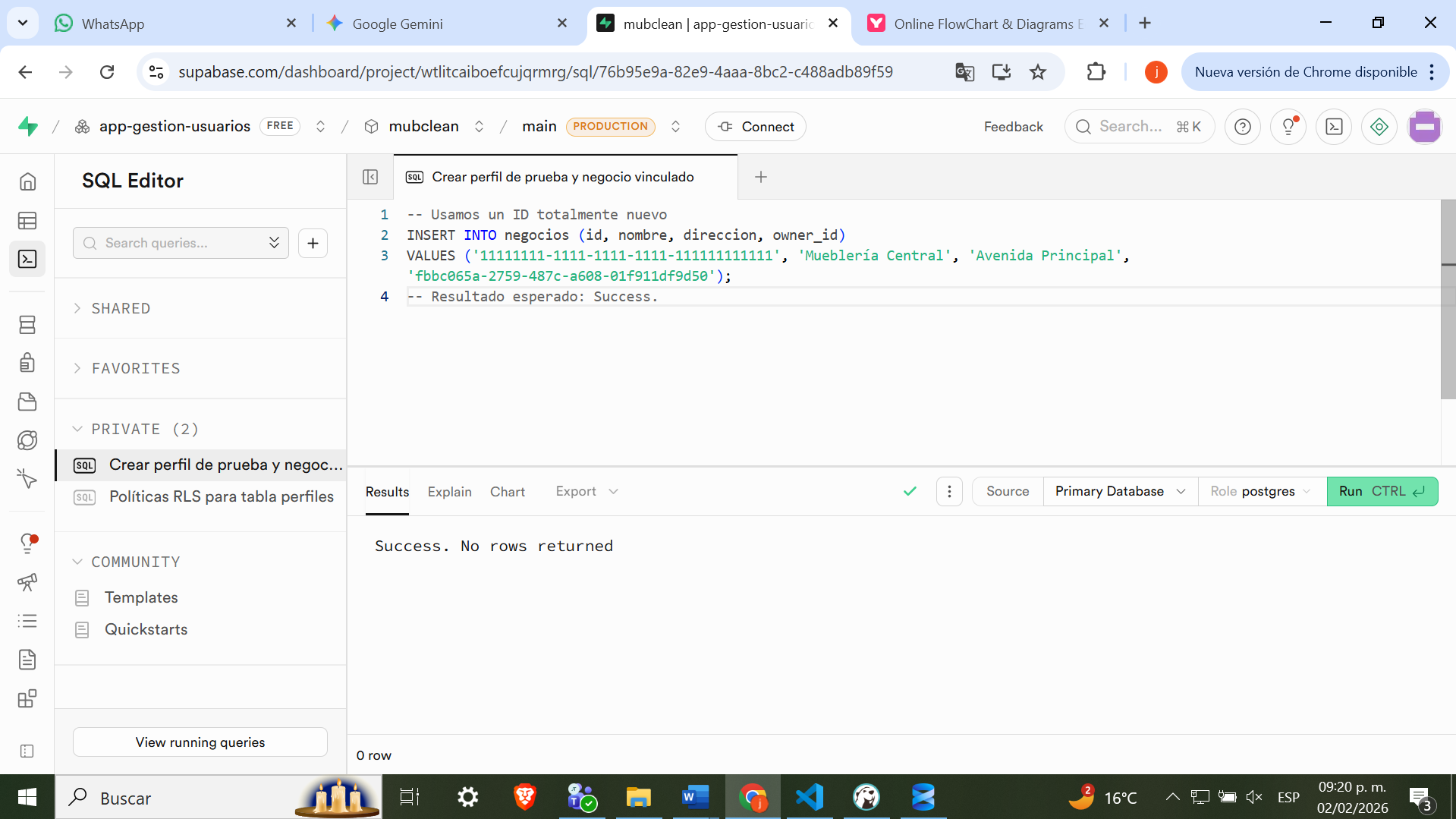
***Restricción: Primary Key (PK)***

**Tabla:** negocios | Columna: id

**Caso de Falla (Duplicidad):**



**Caso de éxito:**

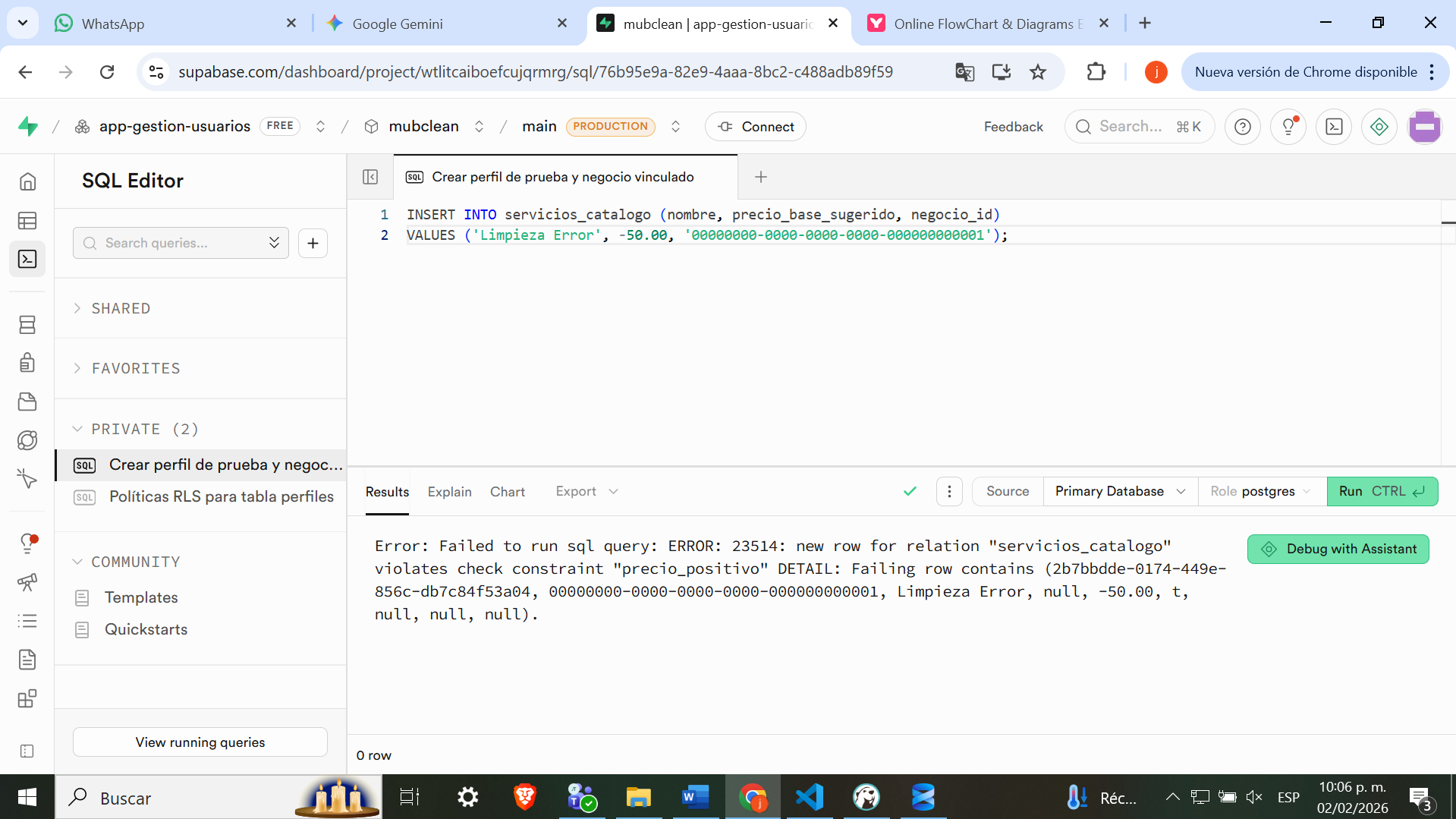


**Descripción para el reporte:** Se valida la restricción de Llave Primaria, la cual garantiza que no existan dos negocios con el mismo identificador único, asegurando la integridad de la entidad principal del sistema.

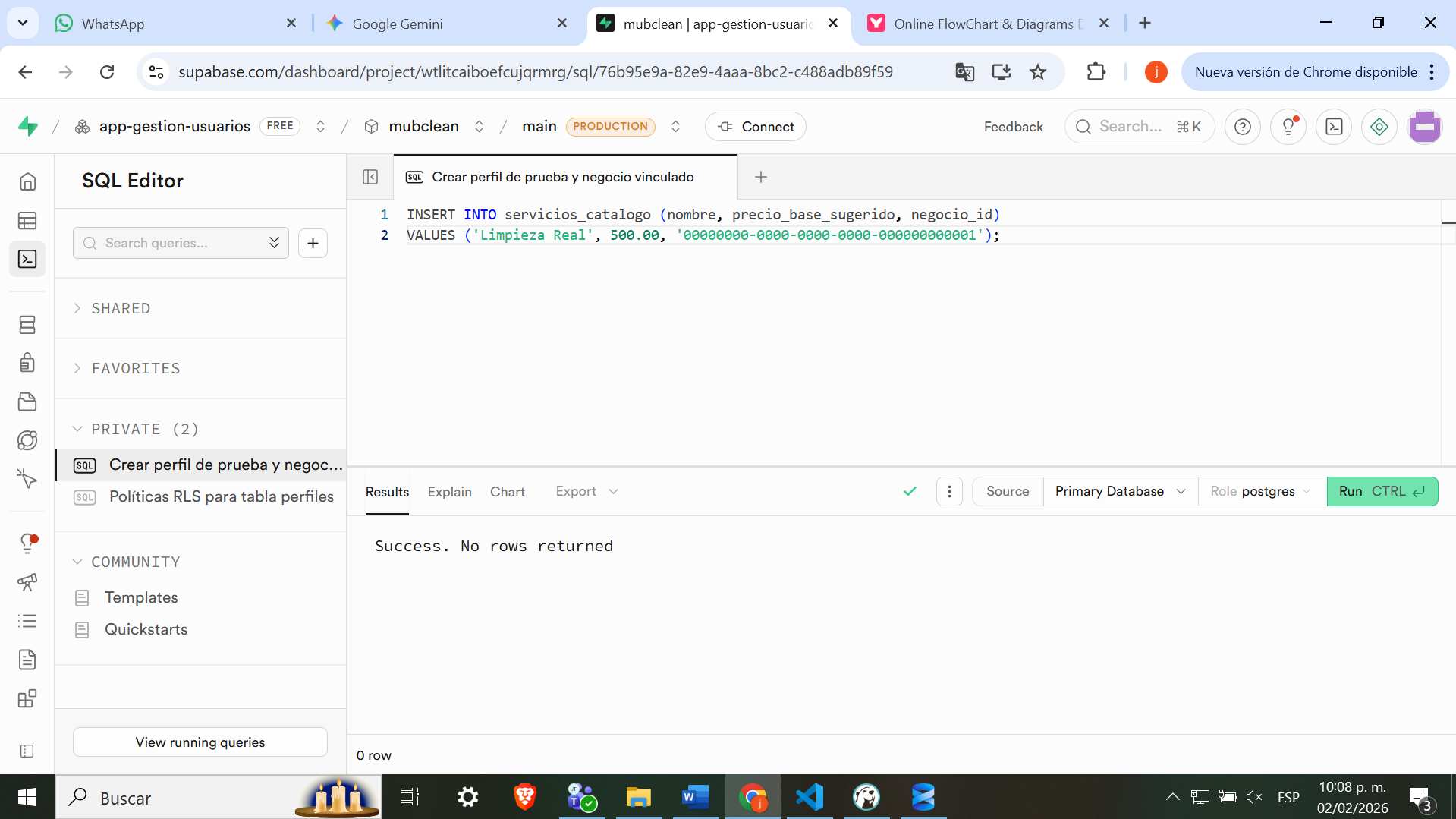
***Restricción: CHECK (Precios Negativos)***

Esta prueba valida que no se puedan cobrar cantidades negativas. Es ideal para documentar la Integridad de Dominio.

**Caso de Falla:**



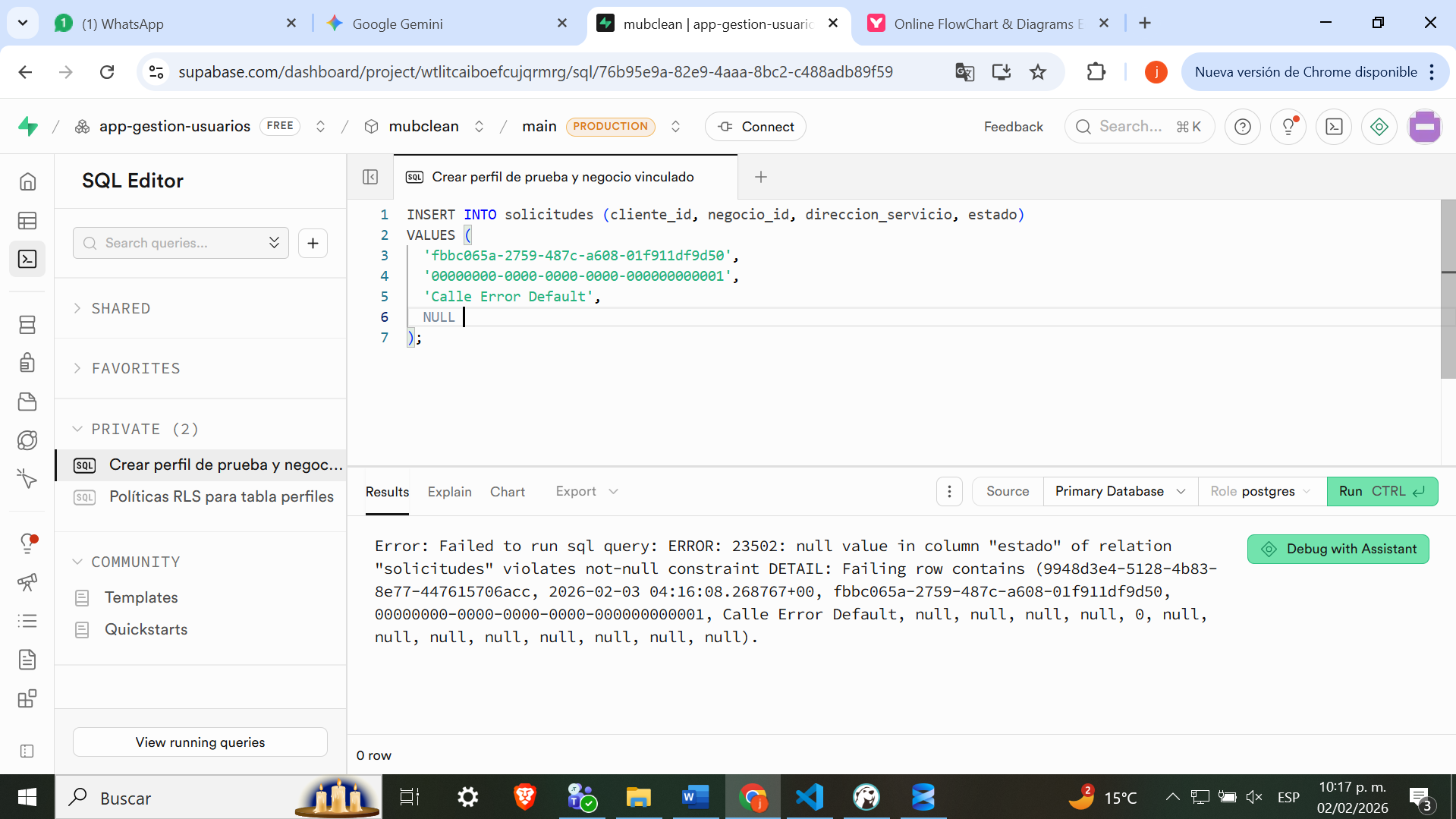
**Caso de éxito**



***Restricción: DEFAULT (Valor por Defecto)***

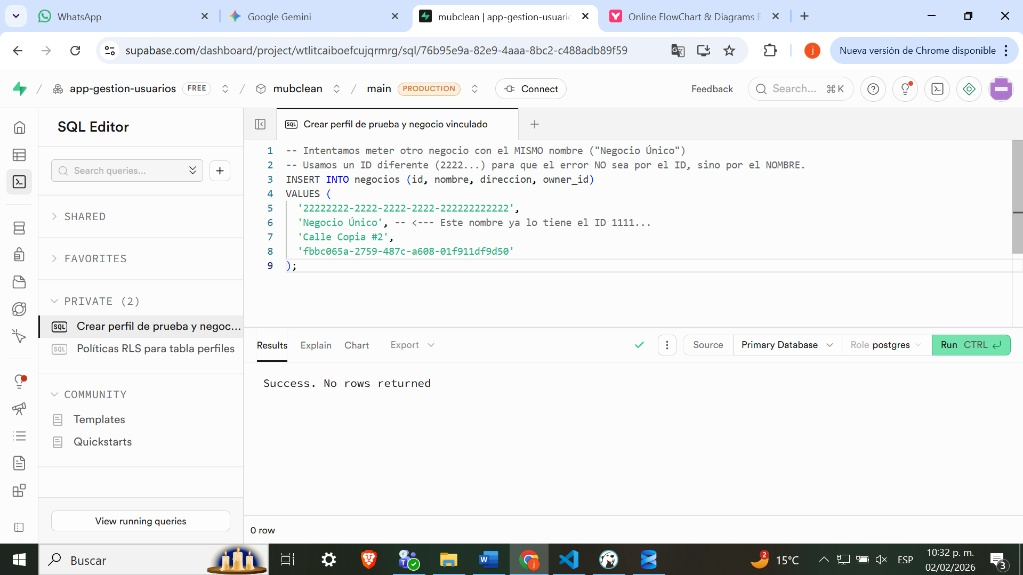
Descripción: "Se configuró la columna 'estado' como obligatoria (NOT NULL) pero con un valor predeterminado ('pendiente'). Esto asegura que ninguna solicitud quede en el limbo, asignándole automáticamente el estado inicial del flujo de negocio si el usuario no lo especifica."

**Caso de falla:**



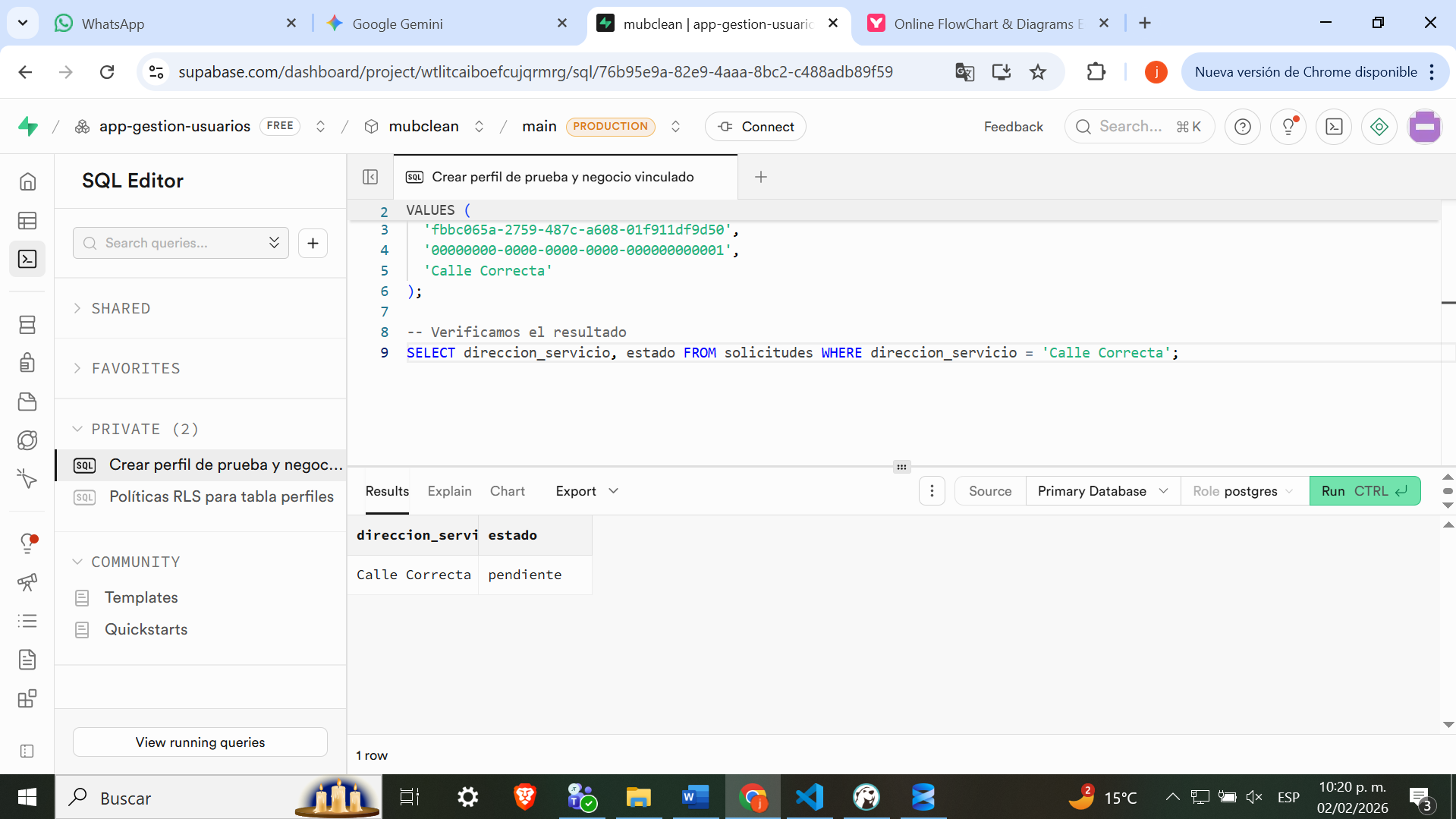
**Acción:** Se intentó forzar la inserción de un valor NULL (vacío) en el campo estado.

**Resultado:** El SGBD bloqueó la operación (Error 23502), confirmando que el campo no admite valores nulos explícitos.

**Caso de éxito:**

**Acción:** Se registró un nuevo negocio llamado "Negocio Único".

**Resultado:** El sistema aceptó la operación exitosamente al verificar que el nombre no existía previamente.



**Acción:** Se registró una nueva solicitud omitiendo la columna estado en la sentencia SQL.

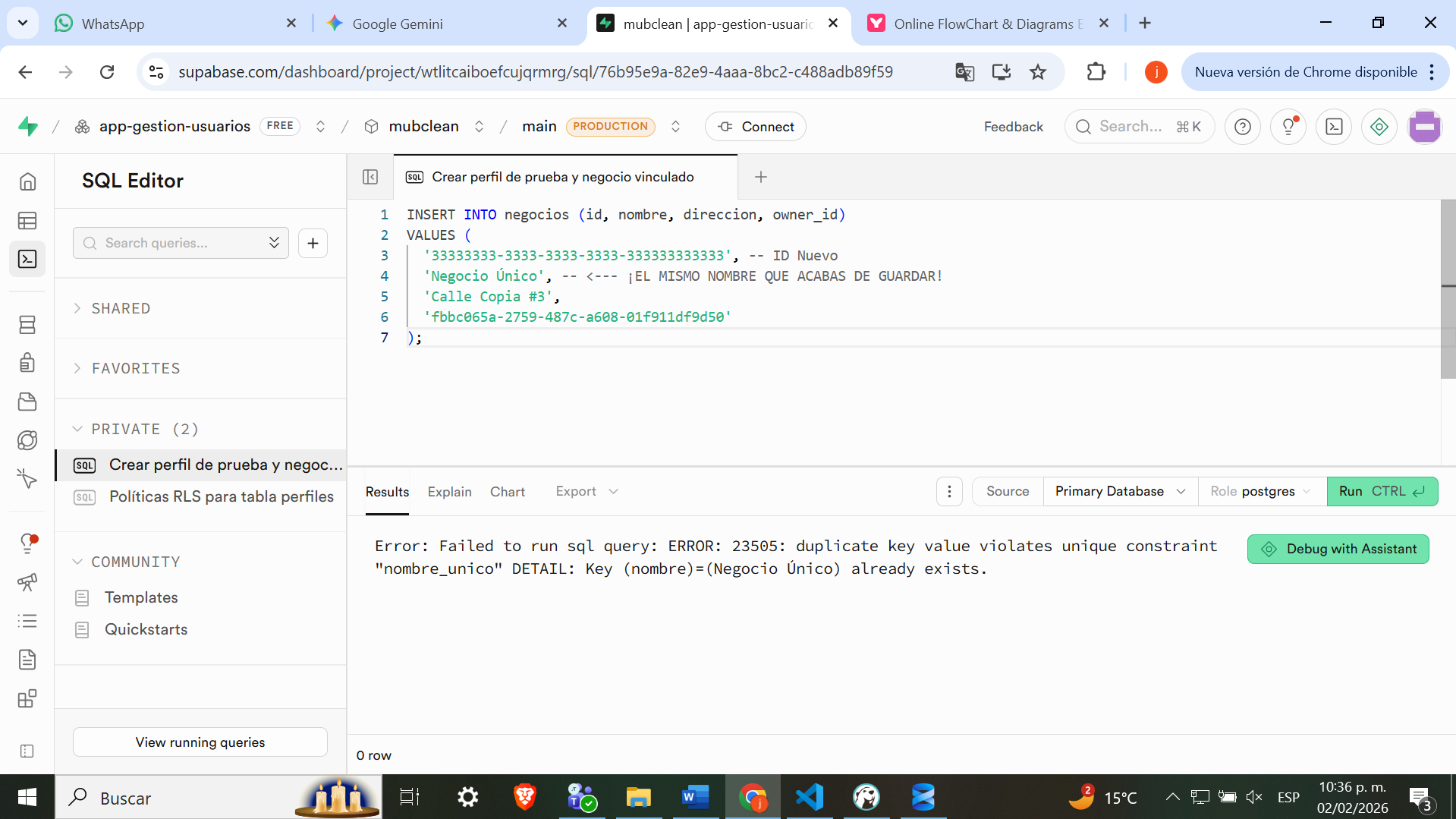
**Resultado:** El sistema aplicó la regla DEFAULT y asignó el valor 'pendiente' automáticamente.

***Restricción: UNIQUE (Unicidad)***

**Descripción:** "Se configuró el campo nombre de la tabla negocios con la restricción UNIQUE. Esto impide que se registren dos establecimientos con la misma razón social, evitando duplicidad y confusión en la plataforma.

**Caso de éxito:**

**Caso de falla:**



**Acción:** Se intentó registrar un segundo negocio con el mismo nombre ("Negocio Único") pero con diferente ID.

**Resultado:** El SGBD bloqueó la transacción (Error 23505), indicando que el valor ya existe y violaría la restricción de unicidad.

**Normalización del Modelo de Datos**

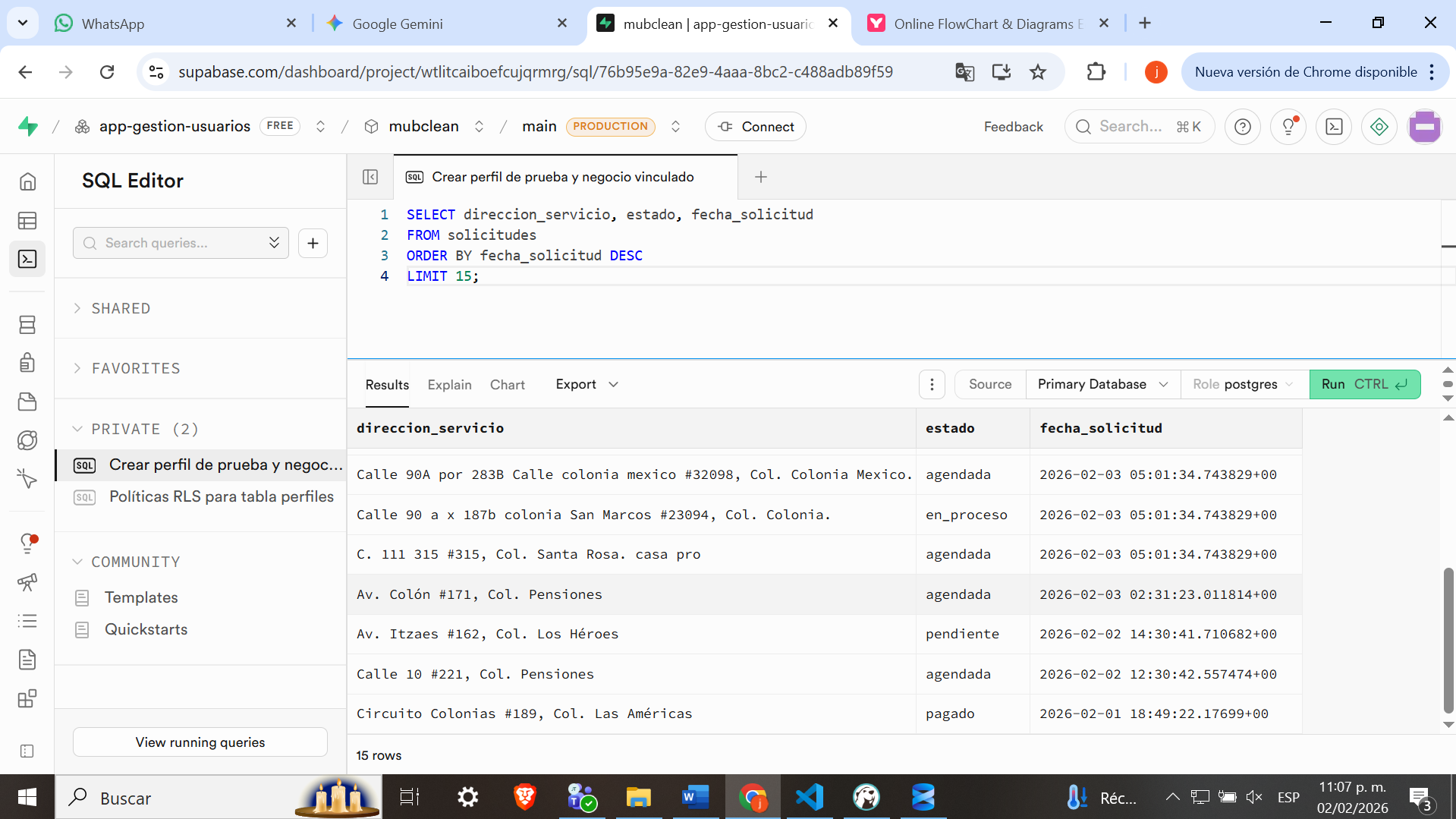
Con el objetivo de garantizar la integridad de la información y optimizar la estructura de la base de datos en Supabase, el esquema relacional de MubClean ha sido validado para cumplir con las Tres Formas Normales (3FN), realizando los siguientes ajustes al diseño original:

**Cumplimiento de la 1FN (Atomicidad):** Se eliminaron estructuras de datos no atómicas (tipo JSONB) en entidades operativas. Específicamente, en la relación de empleados\_negocio, la gestión de disponibilidad (dias\_laborales) se extrajo a una tabla independiente de horarios, asegurando que cada atributo contenga un único valor indivisible.

**Cumplimiento de la 2FN (Dependencia Total):** Todas las entidades cuentan con una clave primaria única (UUID), asegurando que todos los atributos no clave dependan totalmente de la llave primaria y no de una parte de ella.

* **Cumplimiento de la 3FN (Eliminación de Dependencias Transitivas y Datos Calculados):** Se depuraron los campos redundantes cuyo valor dependía de atributos no clave o de cálculos aritméticos:
  1. En el módulo de **Cotización**, se eliminó el campo precio\_total de la tabla solicitudes, estableciendo que el costo final se obtendrá mediante consultas de agregación (SUM) de los ítems relacionados.
  2. En la tabla items\_solicitud, se retiraron los campos precio\_unitario y descripción, ya que estos datos dependen transitivamente del servicio\_catalogo\_id. Esto asegura que la información del servicio se mantenga centralizada en el catálogo servicios\_catalogo, evitando inconsistencias por duplicidad de datos.

**Registros:**



Generación de Datos Masivos (Mock Data)

**Descripción:** Se poblaron las tablas principales con un set de datos de prueba realista utilizando scripts de generación aleatoria. Como se observa en la captura, se simularon direcciones geolocalizadas en Mérida (colonias y avenidas reales), variedad de estados del flujo de servicio (agendada, pagado, etc.) y dispersión temporal en las fechas de solicitud. Esto permite validar el rendimiento de consultas y filtros en escenarios cercanos a producción.

**Anexo: Script SQL de la Base de Datos**

-- public.eventos\_n8n definition

-- Drop table

-- DROP TABLE public.eventos\_n8n;

CREATE TABLE public.eventos\_n8n (

id bigserial NOT NULL,

tipo\_evento text NOT NULL,

solicitud\_id uuid NULL,

payload jsonb DEFAULT '{}'::jsonb NOT NULL,

procesado bool DEFAULT false NOT NULL,

procesado\_at timestamptz NULL,

"error" text NULL,

created\_at timestamptz DEFAULT now() NOT NULL,

CONSTRAINT eventos\_n8n\_pkey PRIMARY KEY (id)

);

CREATE INDEX idx\_eventos\_n8n\_pendientes ON public.eventos\_n8n USING btree (procesado, created\_at);

-- public.empleados\_negocio definition

-- Drop table

-- DROP TABLE public.empleados\_negocio;

CREATE TABLE public.empleados\_negocio (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

negocio\_id uuid NOT NULL,

perfil\_id uuid NOT NULL,

dias\_laborales jsonb DEFAULT '["lun", "mar", "mie", "jue", "vie"]'::jsonb NULL,

activo bool DEFAULT true NULL,

disponible bool DEFAULT false NULL,

CONSTRAINT empleados\_negocio\_negocio\_id\_perfil\_id\_key UNIQUE (negocio\_id, perfil\_id),

CONSTRAINT empleados\_negocio\_pkey PRIMARY KEY (id)

);

-- public.evidencia\_final definition

-- Drop table

-- DROP TABLE public.evidencia\_final;

CREATE TABLE public.evidencia\_final (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

solicitud\_id uuid NOT NULL,

foto\_url text NOT NULL,

comentario\_tecnico text NULL,

created\_at timestamptz DEFAULT now() NULL,

CONSTRAINT evidencia\_final\_pkey PRIMARY KEY (id)

);

-- public.fotos\_solicitud definition

-- Drop table

-- DROP TABLE public.fotos\_solicitud;

CREATE TABLE public.fotos\_solicitud (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

item\_solicitud\_id uuid NOT NULL,

foto\_url text NOT NULL,

CONSTRAINT fotos\_solicitud\_pkey PRIMARY KEY (id)

);

-- public.items\_solicitud definition

-- Drop table

-- DROP TABLE public.items\_solicitud;

CREATE TABLE public.items\_solicitud (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

solicitud\_id uuid NOT NULL,

servicio\_catalogo\_id uuid NULL,

descripcion\_item text NULL,

cantidad int4 DEFAULT 1 NULL,

precio\_unitario numeric DEFAULT 0 NULL,

status text DEFAULT '''pending\_acceptance'''::text NOT NULL,

CONSTRAINT items\_solicitud\_pkey PRIMARY KEY (id)

);

-- public.negocios definition

-- Drop table

-- DROP TABLE public.negocios;

CREATE TABLE public.negocios (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

nombre text NOT NULL,

descripcion text NULL,

telefono\_contacto text NULL,

email\_contacto text NULL,

logo\_url text NULL,

portada\_url text NULL,

owner\_id uuid NOT NULL,

activo bool DEFAULT true NULL,

created\_at timestamptz DEFAULT now() NULL,

direccion text NULL,

telefono text NULL,

subscription\_status text DEFAULT 'pending'::text NULL,

payment\_id text NULL,

license\_expiry timestamptz NULL,

CONSTRAINT negocios\_pkey PRIMARY KEY (id),

CONSTRAINT nombre\_unico UNIQUE (nombre)

);

-- Table Triggers

create trigger on\_negocio\_created after

insert

on

public.negocios for each row execute function ascender\_a\_negocio();

-- public.perfiles definition

-- Drop table

-- DROP TABLE public.perfiles;

CREATE TABLE public.perfiles (

id uuid NOT NULL,

email text NULL,

nombre\_completo text NULL,

telefono text NULL,

foto\_perfil\_url text NULL,

rol public."rol\_sistema" DEFAULT 'cliente'::rol\_sistema NULL,

created\_at timestamptz DEFAULT now() NULL,

foto\_url text NULL,

CONSTRAINT perfiles\_pkey PRIMARY KEY (id)

);

-- public.servicios\_catalogo definition

-- Drop table

-- DROP TABLE public.servicios\_catalogo;

CREATE TABLE public.servicios\_catalogo (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

negocio\_id uuid NOT NULL,

nombre text NOT NULL,

descripcion text NULL,

precio\_base\_sugerido numeric DEFAULT 0 NULL,

activo bool DEFAULT true NULL,

imagen\_url text NULL,

tipo\_servicio text NULL,

notas text NULL,

CONSTRAINT precio\_positivo CHECK ((precio\_base\_sugerido >= (0)::numeric)),

CONSTRAINT servicios\_catalogo\_pkey PRIMARY KEY (id)

);

-- public.solicitudes definition

-- Drop table

-- DROP TABLE public.solicitudes;

CREATE TABLE public.solicitudes (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

created\_at timestamptz DEFAULT now() NULL,

cliente\_id uuid NOT NULL,

negocio\_id uuid NOT NULL,

direccion\_servicio text NOT NULL,

coordenadas\_gps text NULL,

fecha\_solicitada\_cliente date NULL,

metodo\_pago\_preferido text NULL,

estado public."estado\_solicitud" DEFAULT 'pendiente'::estado\_solicitud NOT NULL,

precio\_total numeric DEFAULT 0 NULL,

notas\_cotizacion text NULL,

fecha\_agendada\_final date NULL,

hora\_agendada\_final time NULL,

tecnico\_asignado\_id uuid NULL,

fecha\_completado timestamptz NULL,

fecha\_cancelado timestamptz NULL,

hora\_solicitada\_cliente time NULL,

mercadopago\_payment\_id text NULL,

CONSTRAINT solicitudes\_pkey PRIMARY KEY (id)

);

-- Table Triggers

create trigger tr\_emit\_evento\_n8n\_solicitud\_agendada after

update

of estado on

public.solicitudes for each row execute function fn\_emit\_evento\_n8n\_solicitud\_agendada();

create trigger trg\_emit\_evento\_solicitud\_agendada after

update

of estado on

public.solicitudes for each row execute function fn\_emit\_evento\_solicitud\_agendada();

create trigger trg\_emitir\_evento\_n8n\_por\_estado after

update

of estado on

public.solicitudes for each row execute function fn\_emitir\_evento\_n8n\_por\_estado();

create trigger trg\_emitir\_evento\_solicitud\_agendada after

update

of estado on

public.solicitudes for each row execute function fn\_emitir\_evento\_solicitud\_agendada();

create trigger trg\_evento\_solicitud\_agendada after

update

of estado on

public.solicitudes for each row execute function fn\_evento\_solicitud\_agendada();

create trigger trg\_solicitud\_agendada after

update

on

public.solicitudes for each row execute function fn\_evento\_solicitud\_agendada();

-- public.soporte\_tickets definition

-- Drop table

-- DROP TABLE public.soporte\_tickets;

CREATE TABLE public.soporte\_tickets (

id uuid DEFAULT gen\_random\_uuid() NOT NULL,

created\_at timestamptz DEFAULT timezone('utc'::text, now()) NOT NULL,

cliente\_id uuid NOT NULL,

tipo text NOT NULL,

asunto text NOT NULL,

descripcion text NOT NULL,

estado text DEFAULT 'abierto'::text NULL,

respuesta\_admin text NULL,

foto\_url text NULL,

CONSTRAINT soporte\_tickets\_pkey PRIMARY KEY (id)

);

-- public.empleados\_negocio foreign keys

ALTER TABLE public.empleados\_negocio ADD CONSTRAINT empleados\_negocio\_negocio\_id\_fkey FOREIGN KEY (negocio\_id) REFERENCES public.negocios(id) ON DELETE CASCADE;

ALTER TABLE public.empleados\_negocio ADD CONSTRAINT empleados\_negocio\_perfil\_id\_fkey FOREIGN KEY (perfil\_id) REFERENCES public.perfiles(id);

-- public.evidencia\_final foreign keys

ALTER TABLE public.evidencia\_final ADD CONSTRAINT evidencia\_final\_solicitud\_id\_fkey FOREIGN KEY (solicitud\_id) REFERENCES public.solicitudes(id);

-- public.fotos\_solicitud foreign keys

ALTER TABLE public.fotos\_solicitud ADD CONSTRAINT fotos\_solicitud\_item\_solicitud\_id\_fkey FOREIGN KEY (item\_solicitud\_id) REFERENCES public.items\_solicitud(id) ON DELETE CASCADE;

-- public.items\_solicitud foreign keys

ALTER TABLE public.items\_solicitud ADD CONSTRAINT items\_solicitud\_servicio\_catalogo\_id\_fkey FOREIGN KEY (servicio\_catalogo\_id) REFERENCES public.servicios\_catalogo(id);

ALTER TABLE public.items\_solicitud ADD CONSTRAINT items\_solicitud\_solicitud\_id\_fkey FOREIGN KEY (solicitud\_id) REFERENCES public.solicitudes(id) ON DELETE CASCADE;

-- public.negocios foreign keys

ALTER TABLE public.negocios ADD CONSTRAINT negocios\_owner\_id\_fkey FOREIGN KEY (owner\_id) REFERENCES public.perfiles(id);

-- public.perfiles foreign keys

ALTER TABLE public.perfiles ADD CONSTRAINT perfiles\_id\_fkey FOREIGN KEY (id) REFERENCES auth.users(id) ON DELETE CASCADE;

-- public.servicios\_catalogo foreign keys

ALTER TABLE public.servicios\_catalogo ADD CONSTRAINT servicios\_catalogo\_negocio\_id\_fkey FOREIGN KEY (negocio\_id) REFERENCES public.negocios(id) ON DELETE CASCADE;

-- public.solicitudes foreign keys

ALTER TABLE public.solicitudes ADD CONSTRAINT solicitudes\_cliente\_id\_fkey FOREIGN KEY (cliente\_id) REFERENCES public.perfiles(id);

ALTER TABLE public.solicitudes ADD CONSTRAINT solicitudes\_negocio\_id\_fkey FOREIGN KEY (negocio\_id) REFERENCES public.negocios(id);

ALTER TABLE public.solicitudes ADD CONSTRAINT solicitudes\_tecnico\_asignado\_id\_fkey FOREIGN KEY (tecnico\_asignado\_id) REFERENCES public.empleados\_negocio(id);

-- public.soporte\_tickets foreign keys

ALTER TABLE public.soporte\_tickets ADD CONSTRAINT soporte\_tickets\_cliente\_id\_fkey FOREIGN KEY (cliente\_id) REFERENCES auth.users(id);