# SUBE

TP FINAL - GRUPO II - 2025

Lucas Golchtein, Marcos Achaval, Ludmila Cáceres, Micaela Floridia

# Contenidos

# Motor de BD


**¿Qué es Supabase?**
Plataforma web que ofrece base de datos lista para usar, autenticación de usuarios, APIs automáticas, almacenamiento de archivos, funciones serverless y más.

**¿Qué motor de base de datos usa?**
Supabase usa PostgreSQL, que es un motor de base de datos relacional muy potente, confiable y con muchas funcionalidades avanzadas.

**¿Qué planes tiene?**
Ofrece un plan gratuito pensado para proyectos personales y pequeñas empresas. Además, cuenta con planes de pago para equipos y opciones personalizadas, que permiten acceder a funciones adicionales como branching, aumentar capacidad de almacenamiento, mejorar el rendimiento y otras ventajas.

# Motor de BD

**¿Cómo me creo una cuenta?**
Para crearse una cuenta debemos ingresar a Supabase y registrarnos. Nos van a pedir email y contraseña, y luego algunas configuraciones que queramos.

**¿Cómo creo un proyecto?**
Para crear un proyecto primero debemos crear una organización. Luego podemos crear nuestro proyecto definiendo su nombre, la contraseña de la base de datos y la región en la cual queremos que esté nuestra base de datos.

**Próximos pasos**
Una vez creado nuestro proyecto, podemos acceder al editor SQL en donde podemos definir las queries que queramos, podemos ver las tablas que existen, funciones, triggers y mucho más. Y por supuesto podemos acceder a configuraciones de la base de datos.

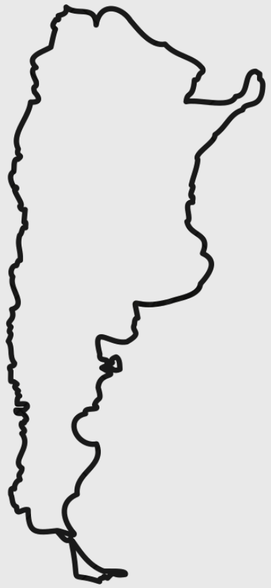
# Escenario

## Sistema SUBE

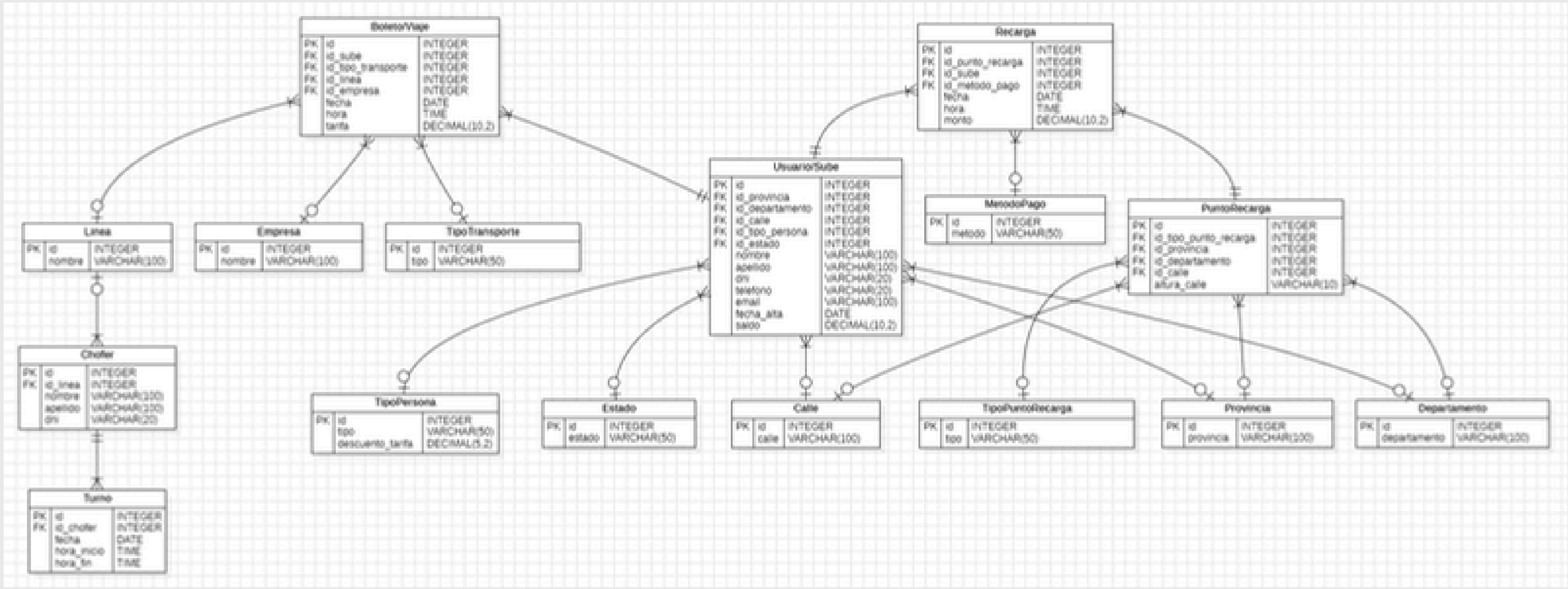
El organismo a cargo del sistema SUBE, solicita implementar la base de datos que almacenará la información del servicio. En la solicitud del sistema, se detallan los siguientes requerimientos:

1. Una persona puede hacerse usuario del sistema y adquirir una tarjeta SUBE (como máximo). Una tarjeta solo puede estar asociada a una persona.
2. Todos los colectivos, estaciones de subte y tren, permitirán viajar con el servicio SUBE a los ciudadanos.
3. Los kioscos, loterías, estaciones de tren y subte podrán ofrecer cargar el saldo de la tarjeta con todos los medios de pago.
4. La empresa de servicios y el gobierno puede consultar la cantidad de recargas y viajes realizados.
5. Los choferes deben registrar su ingreso y egreso al colectivo.
6. Las estaciones de trenes y subtes tienen boletería.

SUBE
SISTEMA ÚNICO DE BOLETO ELECTRÓNICO



# Diagrama de Entidad-Relación



# Creación de la tabla chofer

```
CREATE TABLE chofer (
 id SERIAL primary key,
 id_linea INTEGER NOT NULL references linea (id),
 nombre VARCHAR(100) NOT NULL,
 apellido VARCHAR(100) NOT NULL,
 dni VARCHAR(20) NOT NULL unique
);
```

# Eliminación de la tabla chofer

```
DROP TABLE Chofer;
```

# Inserción en la tabla tipo_persona

```
INSERT INTO tipo_persona (tipo, descuento_tarifa) VALUES
('General', 0),
('Estudiante', 50),
('Universitario', 20),
('Jubilado', 55),
('Discapacitado', 100);
```

# Actualización de la tabla usuario_sube

```
UPDATE usuario_sube
SET email = 'paulatorres01@gmail.com'
WHERE id = 7;
```

# Índices

```
CREATE INDEX IF NOT EXISTS idx_usuario_sube_nombre ON usuario_sube (nombre);

CREATE INDEX IF NOT EXISTS idx_usuario_sube_apellido ON usuario_sube (apellido);
```

# Búsqueda

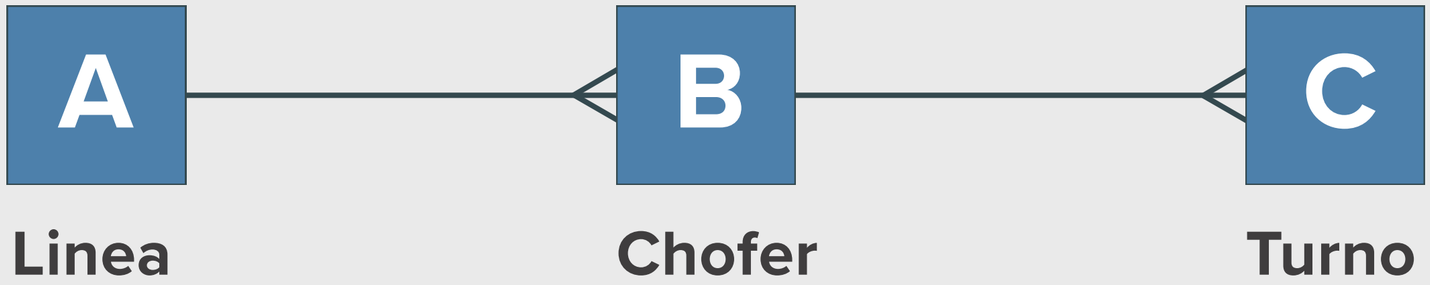
- **Usando una clave**

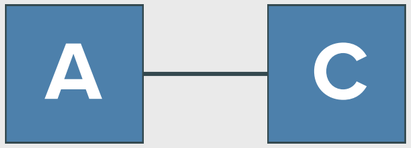
```
SELECT *
FROM usuario_sube
WHERE nombre = 'Juan';
```

- **Usando dos claves**

```
SELECT *
FROM usuario_sube
WHERE nombre = 'Matias'
AND apellido = 'Gómez';
```

# Relación entre tablas

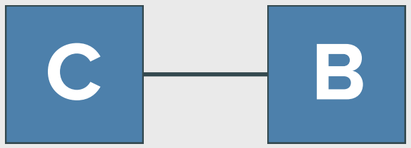






**Obtengo la fecha de turnos de choferes que pertenezcan a la linea ‘107’**

```
SELECT C.fecha   FROM   turno C
JOIN chofer B ON B.id = C.id_chofer
JOIN linea A ON A.id = B.id_linea WHERE  A.id = 3;
```

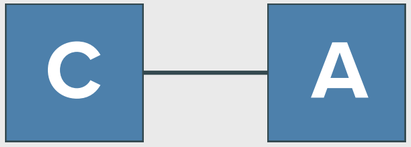
| fecha |
|---|
| 2025-09-11 |
| 2025-10-30 |
| 2025-09-11 |




**Obtengo el DNI de choferes donde hora_inicio = 8:00:00**

```
SELECT B.dni FROM chofer B
JOIN turno C ON C.id_chofer = B.id
WHERE  C.hora_inicio = '08:00:00';
```

| dni |
|---|
| 40000111 |
| 40000333 |
| 40000666 |




**Obtengo líneas de colectivo con turnos hasta 23:00:00 hs**

```
SELECT A.nombre FROM linea A
JOIN chofer B ON B.id_linea = A.id
JOIN turno C ON C.id_chofer = B.id
WHERE  C.hora_fin = '23:00:00';
```

| nombre |
|---|
| 107 |

# Creación de la función actualizar_saldo_sube_viaje

```
CREATE OR REPLACE FUNCTION actualizar_saldo_sube_viaje()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE usuario_sube
  SET saldo = saldo - NEW.tarifa
  WHERE id = NEW.id_sube;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

# Creación del trigger ActualizarSaldoSubeViaje

```
CREATE OR REPLACE TRIGGER ActualizarSaldoSubeViaje
AFTER INSERT ON boleto_viaje
FOR EACH ROW
EXECUTE FUNCTION actualizar_saldo_sube_viaje();
```

# Creación de la función aplicar_descuento_tarifa (Parte 1/2)

```
CREATE OR REPLACE FUNCTION aplicar_descuento_tarifa()
RETURNS TRIGGER AS $$
DECLARE
 v_descuento DECIMAL(5, 2);
BEGIN
 SELECT TP.descuento_tarifa
 INTO v_descuento
 FROM usuario_sube US
 JOIN tipo_persona TP ON US.id_tipo_persona = TP.id
 WHERE US.id = NEW.id_sube;D;
```

# Creación de la función aplicar_descuento_tarifa (Parte 2/2)

```
    IF v_descuento IS NOT NULL THEN
    NEW.tarifa := NEW.tarifa * (1 - v_descuento / 100);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

# Creación del trigger ActualizarDescuentoTarifa

```
CREATE OR REPLACE TRIGGER AplicarDescuentoTarifa
BEFORE INSERT ON boleto_viaje
FOR EACH ROW
EXECUTE FUNCTION aplicar_descuento_tarifa();
```

# ¡MUCHAS GRACIAS!

¿Preguntas?

# + Funciones y Triggers

- **Función : actualizar_saldo_sube_recarga()**

```
CREATE OR REPLACE FUNCTION actualizar_saldo_sube_recarga()
RETURNS TRIGGER AS $$
BEGIN
 UPDATE usuario_sube
 SET saldo = saldo + NEW.monto
 WHERE id = NEW.id_sube;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

- **Trigger : ActualizarSaldoSubeRecarga**

```
CREATE OR REPLACE TRIGGER ActualizarSaldoSubeRecarga
AFTER INSERT ON recarga
FOR EACH ROW
EXECUTE FUNCTION actualizar_saldo_sube_recarga();
```

- **Función : impedir_carga_sube_inactiva()**

```
CREATE OR REPLACE FUNCTION impedir_carga_sube_inactiva()
RETURNS TRIGGER AS $$
DECLARE
  v_estado_usuario INTEGER;
  v_nombre_estado VARCHAR(50);
BEGIN
  SELECT us.id_estado, e.estado
  INTO
    v_estado_usuario, v_nombre_estado
  FROM   usuario_sube us
  JOIN   estado e ON us.id_estado = e.id
  WHERE    us.id = NEW.id_sube;
```

```
  -- ID 2 corresponde a Suspendido
  IF v_estado_usuario = 2 THEN
    RAISE EXCEPTION 'Error: No se puede recargar
una tarjeta con estado (%). ID de usuario: %',
v_nombre_estado, NEW.id_sube;
    RETURN NULL;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

- **Trigger : ImpedirCargaSubeInactiva**

```
CREATE OR REPLACE TRIGGER ImpedirCargaSubeInactiva
BEFORE INSERT ON recarga
FOR EACH ROW
EXECUTE FUNCTION impedir_carga_sube_inactiva();
```

- **Función : obtener_choferes_excedidos()**

```
CREATE OR REPLACE FUNCTION obtener_choferes_excedidos()
RETURNS TABLE(
 nombre VARCHAR,
 apellido VARCHAR,
 fecha DATE,
 horas_trabajadas NUMERIC
)
AS $$

BEGIN
 RETURN QUERY
 SELECT
  ch.nombre,
  ch.apellido,
  t.fecha,
  ROUND(EXTRACT(EPOCH FROM (t.hora_fin - t.hora_inicio)) / 3600, 2) AS horas_trabajadas
 FROM turno t
 JOIN chofer ch ON ch.id = t.id_chofer
 WHERE (t.hora_fin - t.hora_inicio) > INTERVAL '8 hours'; -- Solo turnos de más de 8 horas
END;
$$ LANGUAGE plpgsql;
```

- **Función : obtener_usuarios_inactivos()**

```
CREATE OR REPLACE FUNCTION obtener_usuarios_inactivos()
RETURNS TABLE(
 id INTEGER,
 nombre VARCHAR,
 apellido VARCHAR,
 dni VARCHAR,
 fecha_alta DATE,
 saldo DECIMAL(10,2)
)

AS $$
BEGIN

 RETURN QUERY
 SELECT
  u.id,
  u.nombre,
  u.apellido,
  u.dni,
  u.fecha_alta,
  u.saldo
 FROM usuario_sube u

 LEFT JOIN boleto_viaje b
  ON u.id = b.id_sube
  AND b.fecha >= CURRENT_DATE - INTERVAL '90 days'

 LEFT JOIN recarga r
  ON u.id = r.id_sube
  AND r.fecha >= CURRENT_DATE - INTERVAL '90 days'

 -- Obtengo solo con los que no tienen viajes ni recargas
 WHERE b.id IS NULL
  AND r.id IS NULL;
END;
$$ LANGUAGE plpgsql;
```