



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

Proyecto Final

Sistema de Emprendimientos Juveniles

Bases de Datos

Alejandro Guanoluisa

Emily Galeas

2025-B

Ing. Lorena Chulde



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

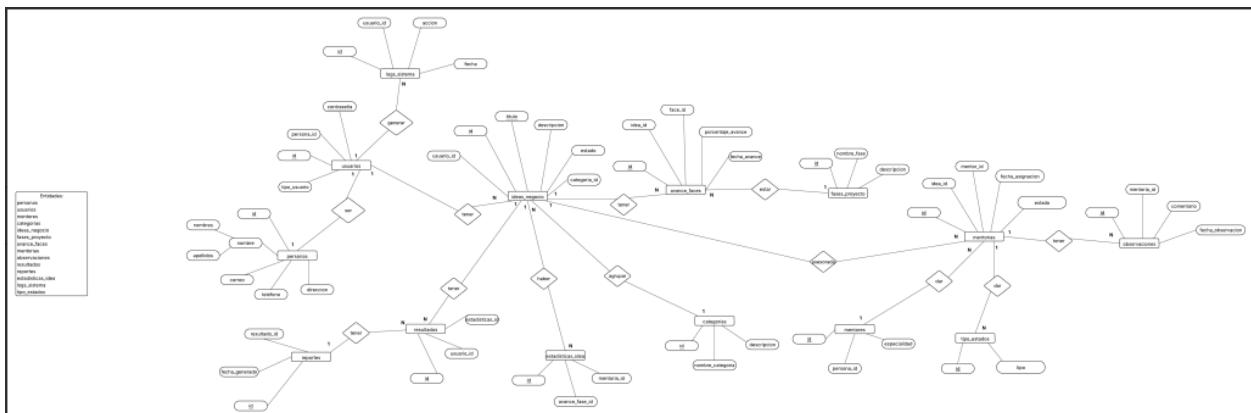
Link Git: https://github.com/emygaleas/ProyectoBimestral_BDD

1.- Creación del Modelo Entidad – Relación

1.1.- Definimos las entidades.



1.2.- Creación del modelo



Link del modelo: https://lucid.app/lucidchart/b6e5fb90-44c3-4fe9-92d7-d07951c5627a/edit?viewport_loc=-503%2C-338%2C3731%2C1621%2C0_0&invitationId=inv_691d99e6-8edf-44da-9c30-3f8de6d19be3



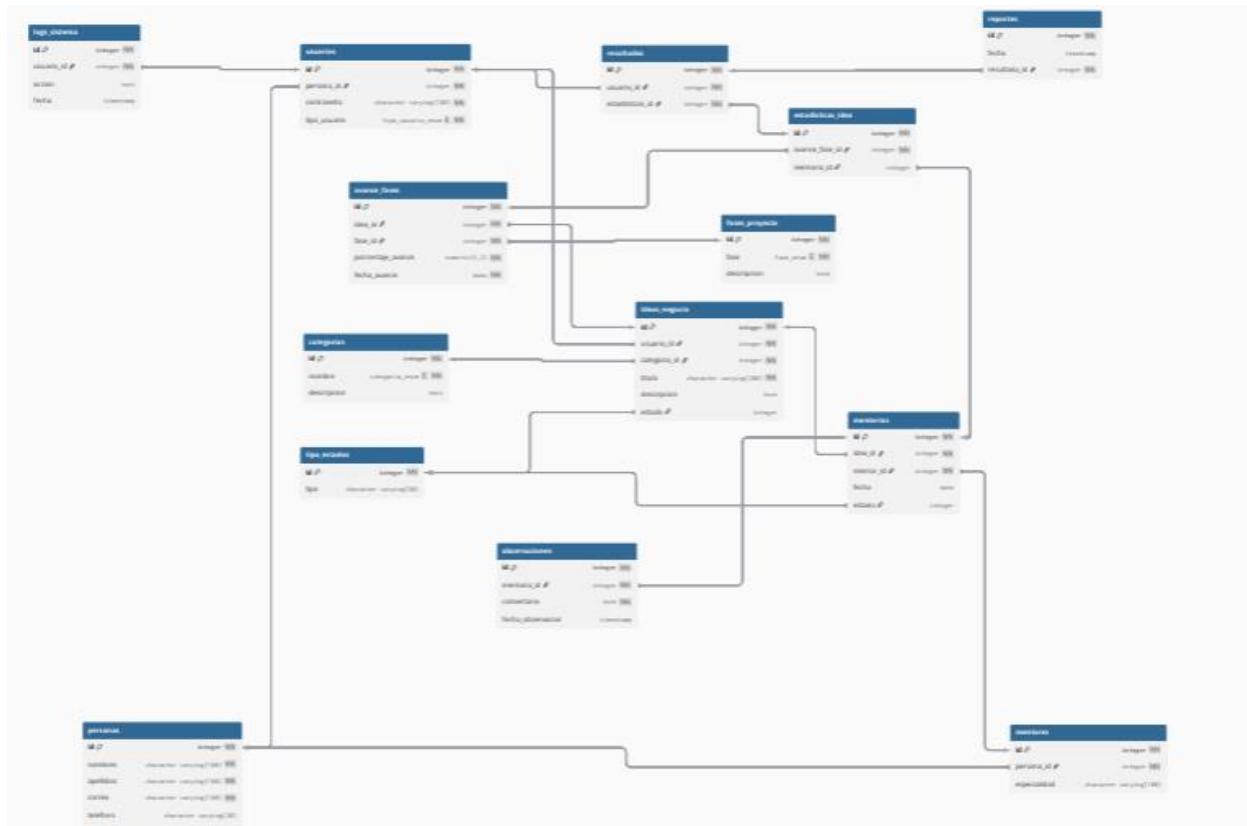
ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

2.- Creación del Modelo Lógico



Link del modelo: <https://dbdocs.io/guanoluisaalejandro5/Sistema-de-proyectos-juveniles>



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

3.- Creación del Modelo Físico

3.1 Modelo en PostgreSQL

```
CREATE TYPE "categoria_enum" AS ENUM (
    'Tecnología',
    'Educación',
    'Salud',
    'Alimentos',
    'Moda',
    'Turismo',
    'Finanzas'
);

CREATE TYPE "estado_enum" AS ENUM (
    'pendiente',
    'aprobado',
    'rechazado',
    'en_proceso',
    'finalizado'
);

CREATE TYPE "fase_enum" AS ENUM (
    'Ideación',
    'Planificación',
    'Desarrollo',
    'Validación',
    'Lanzamiento',
    'Seguimiento'
);

CREATE TYPE "tipo_usuario_enum" AS ENUM (
    'emprendedor',
    'administrador'
);

CREATE TABLE "avance_fases" (
    "id" integer PRIMARY KEY NOT NULL,
    "idea_id" integer NOT NULL,
    "fase_id" integer NOT NULL,
    "porcentaje_avance" numeric(5,2) NOT NULL,
    "fecha_avance" date NOT NULL DEFAULT (CURRENT_TIMESTAMP)
);
```



BASES DE DATOS

(TDSD353)

```
CREATE TABLE "categorias" (
    "id" integer PRIMARY KEY NOT NULL,
    "nombre" categoria_enum UNIQUE NOT NULL,
    "descripcion" text
);

CREATE TABLE "estadisticas_idea" (
    "id" integer PRIMARY KEY NOT NULL,
    "avance_fase_id" integer NOT NULL,
    "mentoría_id" integer
);

CREATE TABLE "fases_proyecto" (
    "id" integer PRIMARY KEY NOT NULL,
    "fase" fase_enum UNIQUE NOT NULL,
    "descripcion" text
);

CREATE TABLE "ideas_negocio" (
    "id" integer PRIMARY KEY NOT NULL,
    "usuario_id" integer NOT NULL,
    "categoria_id" integer NOT NULL,
    "titulo" "character varying(200)" NOT NULL,
    "descripcion" text,
    "estado" integer DEFAULT 1
);

CREATE TABLE "logs_sistema" (
    "id" integer PRIMARY KEY NOT NULL,
    "usuario_id" integer NOT NULL,
    "accion" text,
    "fecha" timestamp DEFAULT (CURRENT_TIMESTAMP)
);

CREATE TABLE "mentores" (
    "id" integer PRIMARY KEY NOT NULL,
    "persona_id" integer NOT NULL,
    "especialidad" "character varying(100)"
);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
CREATE TABLE "mentorias" (
    "id" integer PRIMARY KEY NOT NULL,
    "idea_id" integer NOT NULL,
    "mentor_id" integer NOT NULL,
    "fecha" date DEFAULT (CURRENT_TIMESTAMP),
    "estado" integer
);

CREATE TABLE "observaciones" (
    "id" integer PRIMARY KEY NOT NULL,
    "mentoria_id" integer NOT NULL,
    "comentario" text NOT NULL,
    "fecha_observacion" timestamp DEFAULT (CURRENT_TIMESTAMP)
);

CREATE TABLE "personas" (
    "id" integer PRIMARY KEY NOT NULL,
    "nombres" "character varying(100)" NOT NULL,
    "apellidos" "character varying(100)" NOT NULL,
    "correo" "character varying(100)" UNIQUE NOT NULL,
    "telefono" "character varying(20)",
    "direccion" text
);

CREATE TABLE "reportes" (
    "id" integer PRIMARY KEY NOT NULL,
    "fecha" timestamp DEFAULT (CURRENT_TIMESTAMP),
    "resultado_id" integer NOT NULL
);

CREATE TABLE "resultados" (
    "id" integer PRIMARY KEY NOT NULL,
    "usuario_id" integer NOT NULL,
    "estadisticas_id" integer NOT NULL
);

CREATE TABLE "tipo_estados" (
    "id" integer PRIMARY KEY NOT NULL,
    "tipo" "character varying(50)"
);

CREATE TABLE "usuarios" (
    "id" integer PRIMARY KEY NOT NULL,
    "persona_id" integer NOT NULL,
    "contraseña" "character varying(100)" NOT NULL,
    "tipo_usuario" tipo_usuario_enum NOT NULL
);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

- ▼ Tables (14)
 - > avance_fases
 - > categorias
 - > estadisticas_idea
 - > fases_proyecto
 - > ideas_negocio
 - > logs_sistema
 - > mentores
 - > mentorias
 - > observaciones
 - > personas
 - > reportes
 - > resultados
 - > tipo_estados
 - > usuarios

3.2 Integridad y restricciones

```
ALTER TABLE "avance_fases"
ADD CONSTRAINT "fk_avance_fase" FOREIGN KEY ("fase_id") REFERENCES "fases_proyecto" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "estadisticas_idea"
ADD CONSTRAINT "fk_avance_fase" FOREIGN KEY ("avance_fase_id") REFERENCES "avance_fases" ("id");

ALTER TABLE "avance_fases"
ADD CONSTRAINT "fk_avance_idea" FOREIGN KEY ("idea_id") REFERENCES "ideas_negocio" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "resultados"
ADD CONSTRAINT "fk_estadisticas_id" FOREIGN KEY ("estadisticas_id") REFERENCES "estadisticas_idea" ("id")
ON DELETE CASCADE;

ALTER TABLE "mentorias"
ADD CONSTRAINT "fk_estado" FOREIGN KEY ("estado") REFERENCES "tipo_estados" ("id");

ALTER TABLE "ideas_negocio"
ADD CONSTRAINT "fk_idea_categoria" FOREIGN KEY ("categoria_id") REFERENCES "categorias" ("id")
ON DELETE SET NULL ON UPDATE CASCADE;
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
ALTER TABLE "ideas_negocio"
ADD CONSTRAINT "fk_idea_usuario" FOREIGN KEY ("usuario_id") REFERENCES "usuarios" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "logs_sistema"
ADD CONSTRAINT "fk_log_usuario" FOREIGN KEY ("usuario_id") REFERENCES "usuarios" ("id")
ON DELETE SET NULL ON UPDATE CASCADE;

ALTER TABLE "mentores"
ADD CONSTRAINT "fk_mentor_persona" FOREIGN KEY ("persona_id") REFERENCES "personas" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "estadisticas_idea"
ADD CONSTRAINT "fk_mentoria_id" FOREIGN KEY ("mentor_id") REFERENCES "mentorias" ("id");

ALTER TABLE "mentorias"
ADD CONSTRAINT "fk_mentoria_idea" FOREIGN KEY ("idea_id") REFERENCES "ideas_negocio" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "mentorias"
ADD CONSTRAINT "fk_mentoria_mentor" FOREIGN KEY ("mentor_id") REFERENCES "mentores" ("id");

ALTER TABLE "observaciones"
ADD CONSTRAINT "fk_observacion_mentoria" FOREIGN KEY ("mentor_id") REFERENCES "mentorias" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "reportes"
ADD CONSTRAINT "fk_resultado_id" FOREIGN KEY ("resultado_id") REFERENCES "resultados" ("id")
ON DELETE CASCADE;

ALTER TABLE "resultados"
ADD CONSTRAINT "fk_usuario_id" FOREIGN KEY ("usuario_id") REFERENCES "usuarios" ("id")
ON DELETE CASCADE;

ALTER TABLE "usuarios"
ADD CONSTRAINT "fk_usuario_persona" FOREIGN KEY ("persona_id") REFERENCES "personas" ("id")
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE "ideas_negocio"
ADD CONSTRAINT "ideas_negocio_estado_fkey" FOREIGN KEY ("estado") REFERENCES "tipo_estados" ("id");
```

3.3 Modelo en dbdiagram.io

Generamos un dump de nuestra base creada

```
c:\Windows\system32>pg_dump -h localhost -p 5432 -d Sistema_proyectos_juveniles -U postgres -s -F p -E UTF-8 -f "C:\Users\Flia Guanoluisa\Documents\pg_dump_juveniles.sql"
Contraseña:
```

Copiamos el dump y lo pegamos para que se genere el modelo



ESCUELA POLITÉCNICA NACIONAL



ESCUELA DE FORMACIÓN DE TECNÓLOGOS

BASES DE DATOS

(TDSD353)

Import from PostgreSQL

Instructions

Upload .sql

2. From MySQL
CC
 From PostgreSQL
 From SQL Server
 From Snowflake
 From Rails (schema.rb)

wing
need to add

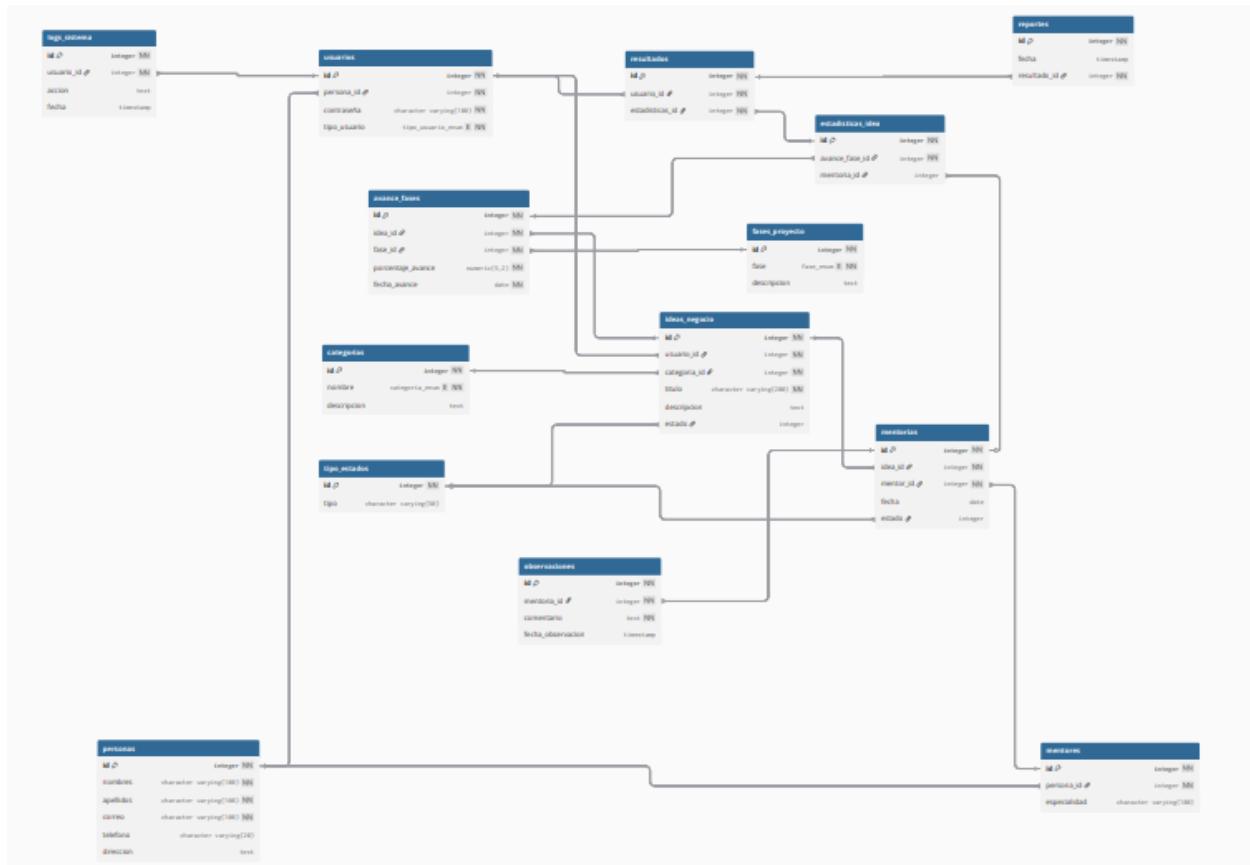
1 --
2 -- PostgreSQL database dump
3 --
4
5 -- Dumped from database version 16.9
6 -- Dumped by pg_dump version 16.9
7
8 SET statement_timeout = 0;
9 SET lock_timeout = 0;
10 SET idle_in_transaction_session_timeout = 0;
11 SET client_encoding = 'UTF8';
12 SET standard_conforming_strings = on;
13 SELECT pg_catalog.set_config('search_path', '', false);
14 SET check_function_bodies = false;
15 SET xmloption = content;
16 SET client_min_messages = warning;
17 SET row_security = off;
18
19 --

Example:

```
$ pg_dump -h localhost -p 5432 -d
ecommerce -U postgres
-s -F p -E UTF-8 -f
C:\pg_schema.sql
```

3. Copy the dumped SQL content and paste

Append converted DBML to the end





ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

Link de dbdiagram.io: <https://dbdocs.io/guanoluisaalejandro5/Sistema-de-proyectos-juveniles>

4. Procedimientos almacenados

4.1 Procedimientos

1.- Procedimiento para insertar idea validando que exista usuario y categoría.

```
create procedure insertar_idea_validada(
    in p_usuario_id int,
    in p_categoria_id int,
    in p_titulo varchar(200),
    in p_descripcion text,
    in p_estado int
)
language plpgsql
as $$

declare
    existe_usuario BOOLEAN;
    existe_categoria BOOLEAN;
begin
    select exists(select 1 from usuarios where id = p_usuario_id) into existe_usuario;
    select exists(select 1 from categorias where id = p_categoria_id) into existe_categoria;

    if not existe_usuario then
        raise exception 'Usuario no existe';
    elseif not existe_categoria then
        raise exception 'Categoría no existe';
    end if;

    insert into ideas_negocio(usuario_id, categoria_id, titulo, descripcion,estado)
    values (p_usuario_id, p_categoria_id, p_titulo, p_descripcion,p_estado);
end;
$$;

call insertar_idea_validada(10,1,'Super tablet', 'tablet con bateria auto recargable',1);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
ERROR: Usuario no existe
CONTEXT: función PL/pgSQL insertar_idea_validada(integer,integer,character varying,text,integer) en la línea 10 en RAISE
67  call insertar_idea_validada(4,10,'Super tablet', 'tablet con bateria auto recargable',1);
68

Data Output Messages Notifications

ERROR: Categoría no existe
CONTEXT: función PL/pgSQL insertar_idea_validada(integer,integer,character varying,text,integer) en la línea 12 en RAISE
SQL state: P0001
```

2.- Procedimiento para cambiar el estado de las ideas a todos los que iniciaron en 1 a otro estado.

	id [PK] integer	usuario_id integer	categoria_id integer	titulo character varying (200)	descripcion text	estado integer
1	3	4	1	SuperTV	Televisión con IA	1
2	4	5	5	AuntoBufanda	Bufanda que se cierra sola	1

```
69  create procedure cambio_estado_idea(
70    in nuevo_estado int
71  )
72  language plpgsql
73  as $$
74  begin
75    update ideas_negocio
76    set estado = nuevo_estado
77    where estado = 1;
78  end;
79  $$
80  call cambio_estado_idea(2);
81
82
```

Data Output Messages Notifications

CALL

Query returned successfully in 66 msec.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

	id [PK] integer	usuario_id integer	categoria_id integer	título character varying (200)	descripcion text	estado integer
1	3	4	1	SuperTV	Television con ia	2
2	4	5	5	AuntoBufanda	Bufanda que se cierra sola	2

3.- Procedimiento para eliminar mentor de forma segura (si no tiene mentorías a cargo) .

```
83 ✓ create procedure eliminar_mentor_sin_mentorias(in mentorId int)
84   language plpgsql
85   as $$
86   declare
87     existen_mentorias boolean;
88   begin
89     select exists(select 1 from mentorias where mentor_id = mentorId or estado = 2 )
90     into existen_mentorias;
91
92   if existen_mentorias then
93     raise exception 'El mentor tiene mentorías asignadas';
94   else
95     delete from mentores where id = mentorId;
96   end if;
97   end;
98   $$
99   call eliminar_mentor_sin_mentorias(1);
100
```

Data Output Messages Notifications

ERROR: El mentor tiene mentorías asignadas
CONTEXT: función PL/pgSQL eliminar_mentor_sin_mentorias(integer) en la línea 9 en RAISE

4.- Procedimiento que inserta un mentor solo si existe como persona.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
103 ✓ create procedure insertar_mentor(in personaId INT, in espe VARCHAR(100))
104 language plpgsql
105 as $$
106 declare
107     la_persona_existe BOOLEAN;
108 begin
109     select exists(select 1 from personas where id = personaId) into la_persona_existe;
110
111     if not la_persona_existe then
112         raise exception 'La persona no existe';
113     else
114         insert into mentores(persona_id, especialidad) values (personaId, espe);
115     end if;
116 end;
117 $$;
118 call insertar_mentor(10,'Salud');
119
120 |
```

Data Output Messages Notifications

ERROR: La persona no existe
CONTEXT: función PL/pgSQL insertar_mentor(integer,character varying) en la línea 8 en RAISE

5.- Procedimiento para registrar un resultado de manera segura, utilizando transacciones para garantizar la integridad de los datos.

```
122 ✓ create procedure registrar_resultado_con_transaccion(
123     IN usuarioId INT,
124     IN estadisticaId INT
125 )
126 language plpgsql
127 as $$
128 begin
129     START TRANSACTION;
130
131     SAVEPOINT antes_de_insertar;
132
133     INSERT INTO resultados(usuario_id, estadisticas_id)
134     VALUES (usuarioId, estadisticaId);
135
136     COMMIT;
137
138     EXCEPTION
139         WHEN OTHERS THEN
140             ROLLBACK;
141             RAISE NOTICE 'Error al registrar resultado. Transacción cancelada.';
142 END;
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
145     call registrar_resultado_con_transaccion(1,1);
146
147
```

Data Output Messages Notifications

NOTICE: Error al registrar resultado. Transacción cancelada.
CALL

4.2 Procedimientos

1.- Procedimiento para asignar mentoría con fecha futura.

```
148 ✓ create procedure asignar_mentoria(in ideaId INT,in mentorId INT, in p_fecha DATE, in nuevo_estado int)
149 language plpgsql
150 as $$
151 begin
152     if p_fecha < current_date then
153         raise exception 'La fecha debe ser futura.';
154     end if;
155
156 ✓     insert into mentorias(idea_id, mentor_id, fecha, estado)
157         values (ideaId, mentorId, p_fecha, nuevo_estado);
158     end;
159     $$;
160
161     call asignar_mentoria(3,1,'2024-07-30',2);
```

Data Output Messages Notifications

ERROR: La fecha debe ser futura.
CONTEXT: función PL/pgSQL asignar_mentoria(integer,integer,date,integer) en la línea 4 en RAISE

2.- Procedimiento para insertar una observación asegurándose de que antes primero exista una mentoría.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
166 ✓ create procedure insertar_observacion_si_existe_mentoria(in mentoriaId int,in nuevo_comentario text)
167   language plpgsql
168   as $$
169   declare
170     existe_mentoria boolean;
171   ✓ begin
172     select exists(select 1 from mentorias where id = mentoriaId) into existe_mentoria;
173   ✓ if not existe_mentoria then
174     raise exception 'No existe una mentoria previa';
175   ✓ else
176     insert into observaciones(mentor_id,comentario)values
177       (mentoriaId,nuevo_comentario);
178   end if;
179   end;
180   $$
181   call insertar_observacion_si_existe_mentoria(5,'Le falta documentacion');
182
183
```

Data Output Messages Notifications

ERROR: No existe una mentoria previa
CONTEXT: función PL/pgSQL insertar_observacion_si_existe_mentoria(integer,text) en la línea 7 en RAISE

```
182   call insertar_observacion_si_existe_mentoria(2,'Le falta documentacion');
```

183

Data Output Messages Notifications

Showing rows: 1 to 1

	id [PK] integer	idea_id integer	mentor_id integer	fecha date	estado integer
1	2	3	1	2025-07-30	2

3.- Procedimiento para insertar usuario si es que la persona existe.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
184 ✓ create procedure crear_usuario(in personaId int,in n_contraseña varchar(100),in n_tipo varchar(100))
185   language plpgsql
186   as $$
187   declare
188     existe_persona boolean;
189 ✓ begin
190   select exists(select 1 from personas where id = personaId) into existe_persona;
191 ✓ if existe_persona then
192   insert into usuarios(persona_id,contraseña,tipo_usuario)values
193   (personaId,n_contraseña,n_tipo);
194 ✓ else
195   raise exception 'La persona no existe';
196   end if;
197   end;
198   $$
199   call crear_usuario(10,'nuevo123','mentor');
200
```

Data Output Messages Notifications

ERROR: La persona no existe

CONTEXT: función PL/pgSQL crear_usuario(integer,character varying,character varying) en la línea 10 en RAISE

SQL state: P0001

4.- Procedimiento para cambiar todos los tipos de persona emprendedor a mentor

	id [PK] integer	persona_id integer	contraseña character varying (100)	tipo_usuario tipo_usuario_enum
1	1	1	alejo123	administrador
2	2	2	emi123	administrador
3	3	3	edu123	mentor
4	4	4	rosa123	emprendedor
5	5	5	ange123	emprendedor



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
201 ✓ create or replace procedure cambiar_tipo_usuario(in tipo_actual varchar(50), in tipo_nuevo varchar(50))
202 language plpgsql
203 as $$ 
204 begin
205 update usuarios
206 set tipo_usuario = tipo_nuevo::tipo_usuario_enum
207 where tipo_usuario = tipo_actual::tipo_usuario_enum;
208 end;
209 $$ 
210 call cambiar_tipo_usuario('emprendedor','mentor');
211
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1

	id [PK] integer	persona_id integer	contraseña character varying (100)	tipo_usuario tipo_usuario_enum
1	1	1	alejo123	administrador
2	2	2	emi123	administrador
3	3	3	edu123	mentor
4	4	4	rosa123	mentor
5	5	5	ange123	mentor

5.- Procedimiento para agregar nuevo avance (solo fecha actual).

```
212 ✓ create or replace procedure agregar_nuevo_avance(in ideaId int,in faseId int,in porcentaje numeric(5,2),in n_fecha_avance d
213 language plpgsql
214 as $$ 
215 begin
216 if n_fecha_avance < current_date or n_fecha_avance > current_date then
217     raise exception 'La fecha debe ser actual.';
218 end if;
219
220 ✓     insert into avance_fases(idea_id, fase_id, porcentaje_avance, fecha_avance)
221         values (ideaId, faseId, porcentaje, n_fecha_avance);
222 end;
223 $$ 
224 call agregar_nuevo_avance(3,3,50,'2025-08-01');
```

Data Output Messages Notifications

ERROR: La fecha debe ser actual.
CONTEXT: función PL/pgSQL agregar_nuevo_avance(integer,integer,numeric,date) en la línea 4 en RAISE

4.3 Funciones

1.- Función para ver un reporte de mentorías por mentor en un rango de fechas



BASES DE DATOS

(TDSD353)

```
165 ✓ create function reporte_mentorias_mentor(
166     in p_mentor_id INT,
167     in fecha_inicio DATE,
168     in fecha_fin DATE
169 )
170 returns table(out_idea_id INT, out_mentor_id INT, out_fecha DATE, out_estado INT)
171 language plpgsql
172 as $$
173 begin
174     return query
175     select m.idea_id, m.mentor_id, m.fecha, m.estado
176     from mentorias m
177     where m.mentor_id = p_mentor_id
178         and m.fecha::DATE between fecha_inicio and fecha_fin;
179 end;
180 $$;
181 |
182 select reporte_mentorias_mentor(1,'2025-07-01','2025-08-01');
```

Data Output	Messages	Notifications
Showing rows: 1 to 1 Page No: <input type="text"/>		
	reporte_mentorias_mentor record	
1	3,1,2025-07-30,2	

2.- Función para ver el porcentaje de ideas finalizadas por usuario.

```
247 ✓ create or replace function porcentaje_ideas_finalizadas(p_usuario_id int)
248     returns numeric as $$ 
249     declare
250         total int;
251         finalizadas int;
252     begin
253         select count(*) into total from ideas_negocio where usuario_id = p_usuario_id;
254         select count(*) into finalizadas from ideas_negocio where usuario_id = p_usuario_id and estado = 5;
255
256     if total = 0 then
257         return 0;
258     end if;
259
260     return (finalizadas * 100.0) / total;
261 end;
262 $$ language plpgsql;
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	porcentaje_ideas_finalizadas
1	0.0000000000000000000000000000



BASES DE DATOS (TDSD353)

3.- función para ver el tipo de usuario que es de acuerdo al numero de ideas que ha dado.

```
265 ✓ create function estado_usuario(p_usuario_id int)
266   returns text as $$ 
267   declare
268     ideas int;
269   begin
270     select count(*) into ideas from ideas_negocio where usuario_id = p_usuario_id;
271
272   if ideas = 0 then
273     return 'Inactivo';
274   elseif ideas < 3 then
275     return 'Activo';
276   else
277     return 'Emprendedor Destacado';
278   end if;
279 end;
280 $$ language plpgsql;
281 select estado_usuario(4);
```

Data Output Messages Notifications

	estado_usuario	text
1		Activo

Showing rows: 1 to 1

5. Triggers

1. Trigger que permite guardar el porcentaje anterior del avance de un proyecto, mantener un historial de los cambios realizados.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- tabla de respaldo
create table avances_fase_historial(
    id SERIAL PRIMARY KEY,
    avance_id integer NOT NULL,
    idea_id integer NOT NULL,
    fase_id integer NOT NULL,
    porcentaje_avance numeric(5,2) NOT NULL,
    porcentaje_anterior numeric (5,2) NOT NULL,
    fecha_avance DATE NOT NULL DEFAULT CURRENT_TIMESTAMP,
    fecha_registro DATE NOT NULL DEFAULT CURRENT_TIMESTAMP,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    usuario varchar(50),
    descripcion varchar(50)
);

-- función
CREATE OR REPLACE FUNCTION registrar_porcentaje_avance()
RETURNS trigger
LANGUAGE plpgsql
AS $$ 
BEGIN
    INSERT INTO avances_fase_historial(
        avance_id, idea_id, fase_id, porcentaje_avance, porcentaje_anterior, usuario, descripcion
    ) VALUES (
        OLD.id,
        OLD.idea_id,
        OLD.fase_id,
        NEW.porcentaje_avance,
        OLD.porcentaje_avance,
        current_user,
        'Actualización del porcentaje de avance.'
    );
    RETURN NEW;
END;
$$;

-- trigger
CREATE TRIGGER trg_Registrar_Porcentaje_Avance
BEFORE UPDATE ON avances_fase
FOR EACH ROW
EXECUTE FUNCTION registrar_porcentaje_avance();
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
45 -- Comprobación
46 select * from avances_fase_historial;
47 update avance_fases set porcentaje_avance = 30 where id=2;
```

Data Output Messages Notifications

	id [PK] integer	avance_id integer	idea_id integer	fase_id integer	porcentaje_avance numeric (5,2)	porcentaje_anterior numeric (5,2)	fecha_avance date	fecha_registro date	fecha timestamp without time zone
1	2	2	1	2	100.00	80.00	2025-08-03	2025-08-03	2025-08-03 18:19:50.9
2	1	2	1	2	40.00	80.00	2025-08-03	2025-08-03	2025-08-03 18:19:40.8
3	3	2	1	2	100.00	100.00	2025-08-03	2025-08-03	2025-08-03 18:20:36.6
4	4	2	1	2	30.00	100.00	2025-08-03	2025-08-03	2025-08-03 18:20:54.1

2. Trigger que permite guardar la información de una idea de negocio al actualizarlo o eliminarlo, para conservar un registro de las modificaciones y eliminaciones.

```
-- tabla de respaldo
CREATE TABLE ideas_negocio_historial (
    id SERIAL PRIMARY KEY,
    idea_id INTEGER NOT NULL,
    usuario_id INTEGER NOT NULL,
    usuario varchar(50),
    accion VARCHAR(20) NOT NULL,
    fecha TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    titulo VARCHAR(200),
    descripcion TEXT,
    estado INTEGER,
    descripcion_accion VARCHAR(255)
);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
-- función
CREATE OR REPLACE FUNCTION auditar_ideas_negocio()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO ideas_negocio_historial (
            idea_id, usuario_id, usuario, accion, titulo, descripcion, estado, descripcion_accion
        ) VALUES (
            OLD.id, OLD.usuario_id, current_user, TG_OP, OLD.titulo, OLD.descripcion, OLD.estado,
            'Eliminación de idea de negocio'
        );
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO ideas_negocio_historial (
            idea_id, usuario_id, usuario, accion, titulo, descripcion, estado, descripcion_accion
        ) VALUES (
            NEW.id, NEW.usuario_id, current_user, TG_OP, NEW.titulo, NEW.descripcion, NEW.estado,
            'Actualización de idea de negocio'
        );
        RETURN NEW;
    ELSE -- INSERT
        INSERT INTO ideas_negocio_historial (
            idea_id, usuario_id, usuario, usuario, accion, titulo, descripcion, estado, descripcion_accion
        ) VALUES (
            NEW.id, NEW.usuario_id, current_user, TG_OP, NEW.titulo, NEW.descripcion, NEW.estado,
            'Inserción de idea de negocio'
        );
        RETURN NEW;
    END IF;
END;
$$;
```

```
-- trigger
CREATE TRIGGER trg_auditar_ideas
AFTER INSERT OR UPDATE OR DELETE ON ideas_negocio
FOR EACH ROW
EXECUTE FUNCTION auditar_ideas_negocio();
```

```
107  -- Comprobación
108  select * from ideas_negocio_historial;
109  update ideas_negocio set titulo = 'Plataforma' where id = 2;
110
```

Data Output Messages Notifications

	id	idea_id	usuario	usuario	accion	fecha	titulo	descripcion
	[PK]	integer	integer	character varying	character varying	timestamp without time zone	character varying	text
1	1	2	5	postgres	UPDATE	2025-08-03 18:24:2...	Plataforma	Marketplace para que profesionales independientes ofrezcan cursos en dive

3. Trigger que permite guardar la información de una mentoría antes de eliminarla, además de eliminar los registros relacionados en estadísticas y observaciones para evitar inconsistencias.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- Tabla de respaldo
CREATE TABLE mentoria_historial (
    id SERIAL PRIMARY KEY,
    mentoria_id INTEGER NOT NULL,
    idea_id INTEGER,
    mentor_id INTEGER,
    fecha DATE,
    estado INTEGER,

    observacion text,
    fecha_observacion TIMESTAMP,

    accion varchar(30) NOT NULL,
    fecha_accion TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    usuario varchar(50) NOT NULL
);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- Función
CREATE OR REPLACE FUNCTION guardar_historial_mentoria()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$

BEGIN
    -- Insertar una fila por cada observación asociada
    INSERT INTO mentoría_historial (
        mentoría_id, idea_id, mentor_id, fecha, estado,
        observación, fecha_observación,
        acción, usuario
    )
    SELECT
        OLD.id, OLD.idea_id, OLD.mentor_id, OLD.fecha, OLD.estado,
        o.comentario, o.fecha_observacion,
        'DELETE', current_user
    FROM observaciones o
    WHERE o.mentoría_id = OLD.id;

    -- Si no hay observaciones, guardar los datos de la mentoría
    IF NOT EXISTS (SELECT 1 FROM observaciones WHERE mentoría_id = OLD.id) THEN
        INSERT INTO mentoría_historial (
            mentoría_id, idea_id, mentor_id, fecha, estado,
            acción, usuario
        )
        VALUES (
            OLD.id, OLD.idea_id, OLD.mentor_id, OLD.fecha, OLD.estado,
            'DELETE', current_user
        );
    END IF;

    -- Eliminar observaciones relacionadas
    DELETE FROM observaciones WHERE mentoría_id = OLD.id;

    -- Eliminar estadísticas relacionadas
    DELETE FROM estadísticas_idea WHERE mentoría_id = OLD.id;

    RETURN OLD;
END;
$$;
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
-- Trigger
CREATE TRIGGER trg_guardar_mentoria_y_eliminar_estadisticas
BEFORE DELETE ON mentorias
FOR EACH ROW
EXECUTE FUNCTION guardar_historial_mentoria();
```

```
178  -- Comprobación
179  delete from mentorias where id = 100;
180  select * from mentoria_historial;
```

1.8.1 Data Output Messages Notifications

	id	[PK] integer	mentoría_id	integer	idea_id	integer	mentor_id	integer	fecha	date	estado	integer	observacion	text	fecha_observacion	timestamp with time zone
1	1		100		100		38		2025-04-02		1		Observación para la mentoría 100: El diseño de UX necesita mejoras		2025-07-15 00:00:00+00	

6. Índices y Optimización

Los índices en PostgreSQL son cruciales para optimizar el rendimiento de las consultas ya que permiten que el servidor busque datos de manera más eficiente. Por ende, se han creado los siguientes índices simples para mejorar búsquedas y filtados:

```
-- Simples
-- Ideas de negocio
CREATE INDEX idx_ideas_estado ON ideas_negocio(estado);
CREATE INDEX idx_ideas_usuario_id ON ideas_negocio(usuario_id);

-- Avances de fases
CREATE INDEX idx_avance_idea_id ON avance_fases(idea_id);

-- Usuarios
CREATE INDEX idx_usuarios_persona_id ON usuarios(persona_id);

-- Estadísticas
CREATE INDEX idx_estadisticas_avance_fase ON estadisticas_idea(avance_fase_id);

-- Mentorías
CREATE INDEX idx_mentorias_id ON mentorias(id);
CREATE INDEX idx_mentorias_estado ON mentorias(estado);

-- Tipos de estados
CREATE INDEX idx_tipo_estados_id ON tipo_estados(id);
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

También se definieron índices compuestos para optimizar consultas que involucran múltiples columnas:

```
-- Compuestos
-- Ideas de negocio
CREATE INDEX idx_ideas_estado_usuario ON ideas_negocio(estado, usuario_id);

-- Estadísticas
CREATE INDEX idx_estadisticas_compuesto ON estadisticas_idea(avance_fase_id, mentoria_id);

-- Mentorías
CREATE INDEX idx_mentorias_compuesto ON mentorias(id, estado);
CREATE INDEX idx_ideas_negocio_estado_id ON ideas_negocio(estado, id);
CREATE INDEX idx_avance_fases_idea_fase ON avance_fases(idea_id, fase_id);
```

El rendimiento de las consultas se ha medido con la siguiente instrucción:

```
-- Análisis de rendimiento con EXPLAIN
EXPLAIN ANALYZE
SELECT i.id, i.titulo, p.nombres, p.apellidos, ef.fase_id, ef.porcentaje_avance
FROM ideas_negocio i
JOIN usuarios u ON i.usuario_id = u.id
JOIN personas p ON u.persona_id = p.id
JOIN avance_fases ef ON i.id = ef.idea_id
WHERE i.estado = 2;
```

1. Rendimiento antes de implementar índices



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

Data Output Messages Notifications

Showing rows: 1 to 22 Page No: 1 of 1

QUERY PLAN	
1	Nested Loop (cost=4000000000.85..40000000361.12 rows=311 width=62) (actual time=0.055..1.831 rows=300 loops=1)
2	-> Nested Loop (cost=3000000000.57..30000000262.91 rows=311 width=38) (actual time=0.049..1.396 rows=300 loops=1)
3	-> Nested Loop (cost=2000000000.28..20000000177.67 rows=311 width=38) (actual time=0.037..1.119 rows=300 loops=1)
4	-> Seq Scan on avance_fases ef (cost=1000000000.00..10000000019.95 rows=1195 width=13) (actual time=0.013..0.098 rows=...)
5	-> Memoize (cost=0.29..0.33 rows=1 width=29) (actual time=0.001..0.001 rows=0 loops=1195)
6	Cache Key: ef.idea_id
7	Cache Mode: logical
8	Hits: 797 Misses: 398 Evictions: 0 Overflows: 0 Memory Usage: 33kB
9	-> Index Scan using ideas_negocio_pkey on ideas_negocio i (cost=0.28..0.32 rows=1 width=29) (actual time=0.001..0.001 rows=...)
10	Index Cond: (id = ef.idea_id)
11	Filter: (estado = 2)
12	Rows Removed by Filter: 1
13	-> Memoize (cost=0.29..0.38 rows=1 width=8) (actual time=0.001..0.001 rows=1 loops=300)
14	Cache Key: i.usuario_id
15	Cache Mode: logical
16	Hits: 201 Misses: 99 Evictions: 0 Overflows: 0 Memory Usage: 11kB
17	-> Index Scan using usuarios_pkey on usuarios u (cost=0.28..0.37 rows=1 width=8) (actual time=0.001..0.001 rows=1 loops=99)
18	Index Cond: (id = i.usuario_id)
19	-> Index Scan using personas_pkey on personas p (cost=0.28..0.32 rows=1 width=32) (actual time=0.001..0.001 rows=1 loops=300)
20	Index Cond: (id = u.persona_id)
21	Planning Time: 0.721 ms
22	Execution Time: 1.883 ms

Total rows: 22 Query complete 00:00:00.053 CRLF Ln 50, Col 39

2. Rendimiento con índices



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

Data Output Messages Notifications

Showing rows: 1 to 13 | Page No: 1 of 1 | Back | Forward | Last | First |

QUERY PLAN	
	text
1	Nested Loop (cost=30000000001.10..30000000260.99 rows=311 width=62) (actual time=0.034..0.723 rows=300 loops=1)
2	-> Nested Loop (cost=20000000000.82..20000000171.81 rows=208 width=53) (actual time=0.030..0.503 rows=208 loops=1)
3	-> Nested Loop (cost=10000000000.55..10000000106.12 rows=208 width=29) (actual time=0.025..0.289 rows=208 loops=1)
4	-> Index Scan using idx_ideas_negocio_estado_id on ideas_negocio i (cost=0.28..28.78 rows=208 width=29) (actual time=0.019..0.064 rows=20...)
5	Index Cond: (estado = 2)
6	-> Index Scan using usuarios_pkey on usuarios u (cost=0.28..0.37 rows=1 width=8) (actual time=0.001..0.001 rows=1 loops=208)
7	Index Cond: (id = i.usuario_id)
8	-> Index Scan using personas_pkey on personas p (cost=0.28..0.32 rows=1 width=32) (actual time=0.001..0.001 rows=1 loops=208)
9	Index Cond: (id = u.persona_id)
10	-> Index Scan using idx_avance_idea_id on avance_fases ef (cost=0.28..0.40 rows=3 width=13) (actual time=0.001..0.001 rows=1 loops=208)
11	Index Cond: (idea_id = i.id)
12	Planning Time: 0.251 ms
13	Execution Time: 0.753 ms

Total rows: 13 | Query complete 00:00:00.063 | CRLF | Ln 58, Col 20

Conclusión: Antes de crear los índices, las consultas recorrían toda la tabla para encontrar los datos, lo que hacía que se demore la ejecución, consuma mucha memoria y se repitan varias operaciones. Después de aplicar los índices, como idx_ideas_estado e idx_avance_idea, se puede llegar más rápido a la información requerida.

7. Seguridad y Roles

1. Creación de Roles

Se crearon cinco roles diferentes para: administrador, auditor, operador, mentor y emprendedor.

```
CREATE ROLE administrador NOLOGIN SUPERUSER;
CREATE ROLE auditor NOLOGIN NOSUPERUSER;
CREATE ROLE operador NOLOGIN NOSUPERUSER;
CREATE ROLE mentor NOLOGIN NOSUPERUSER;
CREATE ROLE emprendedor NOLOGIN NOSUPERUSER;
```

SUPERUSER: El usuario tiene el control total sobre la base de datos.

NOSUPERUSER: El usuario no tiene el control total sobre la base de datos, solo tiene los permisos asignados con GRANT.



BASES DE DATOS (TDSD353)

2. Gestión de privilegios

- El administrador puede acceder, leer, escribir, borrar y alterar todo en la base de datos.

```
GRANT ALL PRIVILEGES ON DATABASE bdd_juveniles TO administrador;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO administrador;
```

- El auditor solo puede leer los datos de todas las tablas.

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO auditor;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO auditor;
```

- El operador puede consultar, insertar y modificar datos en personas, usuarios e ideas_negocio. Por otro lado, puede usar las secuencias para insertar nuevos usuarios o ideas_negocio.

```
GRANT SELECT, INSERT, UPDATE ON personas, usuarios, ideas_negocio TO operador;
GRANT SELECT ON categorias, fases_proyecto, tipo_estados TO operador;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO operador;
```

- El mentor puede crear y editar mentorías y observaciones, puede consultar ideas y avances pero no editarlas. Puede usar secuencias necesarias para agregar mentorías y observaciones.

```
GRANT SELECT, INSERT, UPDATE ON mentorias, observaciones TO mentor;
GRANT SELECT ON ideas_negocio, avance_fases TO mentor;
GRANT USAGE ON SEQUENCE mentorias_id_seq, observaciones_id_seq TO mentor;
```

- El emprendedor puede crear y modificar sus ideas de negocio, puede ver mentorías, observaciones y avances, pero no editarlos. Puede usar secuencias para insertar nuevas ideas.

```
GRANT SELECT, INSERT, UPDATE ON ideas_negocio TO emprendedor;
GRANT SELECT ON mentorias, observaciones, avance_fases TO emprendedor;
GRANT USAGE ON SEQUENCE ideas_negocio_id_seq TO emprendedor;
```

Nota: En PostgreSQL, aunque un rol tenga permiso para insertar datos en una tabla, no puede crear registros nuevos si no tiene



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

acceso a la secuencia que genera el ID automático, es decir, aunque pueda crear ideas, observaciones u otros datos, si no tiene el permiso USAGE sobre la secuencia correspondiente, el sistema no le permite insertar porque no puede generar el ID. Por eso es necesario dar ese permiso aparte.

3. Encriptación con pgcrypto

1. Activar extensión pgcrypto, para usar funciones de encriptación como crypt() y gen_salt().

```
-- Extensión para funciones criptográficas
CREATE EXTENSION IF NOT EXISTS pgcrypto;
```

2. Modificar el tipo de la columna contraseña a text para almacenar contraseñas encriptadas, ya que varchar no almacenaría las contraseñas completas.

```
ALTER TABLE public.usuarios
ALTER COLUMN "contraseña" type text;
```

3. Actualizar contraseñas actuales, con crypt() y gen_salt().

```
UPDATE public.usuarios
SET "contraseña" = crypt("contraseña", gen_salt('bf', 8));
```

4. Crear la función encriptar_contraseña() para que automáticamente la contraseña se encripte cada que se inserte o actualice.

```
CREATE OR REPLACE FUNCTION encriptar_contraseña()
RETURNS TRIGGER AS $$ 
BEGIN
    NEW.contraseña = crypt(NEW.contraseña, gen_salt('bf'));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

5. Crear el trigger tr_encriptar_contraseña que ejecuta la función antes de cada INSERT o UPDATE sobre la columna contraseña.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
CREATE TRIGGER tr_encriptar_contrasenia
BEFORE INSERT OR UPDATE OF "contraseña" ON usuarios
FOR EACH ROW EXECUTE FUNCTION encriptar_contrasenia();
```

- Antes de encriptar las contraseñas:

```
42  -- Antes de encriptar
43  select * from usuarios;
44
```

Data Output Messages Notifications

Showing row

	id [PK] integer	persona_id integer	contraseña text	tipo_usuario tipo_usuario_enum
1	102	101	user101pass	mentor
2	103	102	user102pass	emprendedor
3	104	103	user103pass	mentor
4	105	104	user104pass	mentor
5	106	105	user105pass	emprendedor

- Despues de encriptar las contraseñas:

```
42  -- Despues de encriptar
43  select * from usuarios;
44
```

Data Output Messages Notifications

Showing rows: 1 to 802 Page No: 1 of 1

	id [PK] integer	persona_id integer	contraseña text	tipo_usuario tipo_usuario_enum
1	102	101	\$2a\$08\$wC7PO/lfr1WvrHLithZNGu91nvFnC17jnu240NYiPN3lxL.F2qa3q	mentor
2	103	102	\$2a\$08\$NYVeVTlxjhST57nAUGTvKOT5BX.3Aj2nKBMAbnTwEmdvkCWto...	emprendedor
3	104	103	\$2a\$08\$js8FSYXVJcvQW7iHYb7v.IWEoJ1csR58m3kLTnsty5h5nRUQM5u	mentor
4	105	104	\$2a\$08\$0pGrP04/WgnMCyP0SeuM70W4frNmTThB8GLnMy4Z5BW2Lh...	mentor
5	106	105	\$2a\$08\$rdgW7GAy5jBXJTZOTWhU.urhVr3KJZr00UvHab2XM3exSbQgDb...	emprendedor

4. Simulación de conexión SSL/TLS para asegurar el tráfico entre cliente y servidor.



BASES DE DATOS (TDSD353)

Para simular una conexión segura entre el cliente y el servidor PostgreSQL en un entorno local, se utilizó el protocolo SSL/TLS. Este proceso cifra la comunicación y evita que los datos viajen en texto plano.

- **Instalación de OpenSSL:** Se descargó e instaló el paquete Win64 OpenSSL v3.5.1 Light esde el siguiente link:

[Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions](https://slproweb.com/products/Win32OpenSSL.html)

The screenshot shows a web browser window with the URL <https://slproweb.com/products/Win32OpenSSL.html>. The page title is "Win32/Win64 OpenSSL Screenshot". It features a screenshot of the installer's welcome screen and a download section. The download section includes a file list table:

File	Type	Description
Win64 OpenSSL v3.5.1 Light EXE MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.5.1 (Recommended for users by the creators of OpenSSL). Only installs an Ed25519 variant of

- **Aceptar la licencia:** Durante la instalación, se aceptaron los términos de la licencia de software.

The screenshot shows the "Setup - OpenSSL 3.5.1 Light (64-bit)" window. The title bar says "License Agreement". The main content area displays the license text and a note about donations. At the bottom, there are two radio buttons for accepting or declining the agreement, and buttons for "Next >" and "Cancel".

- **Ruta de instalación:** Se escogió la ruta de instalación por defecto.



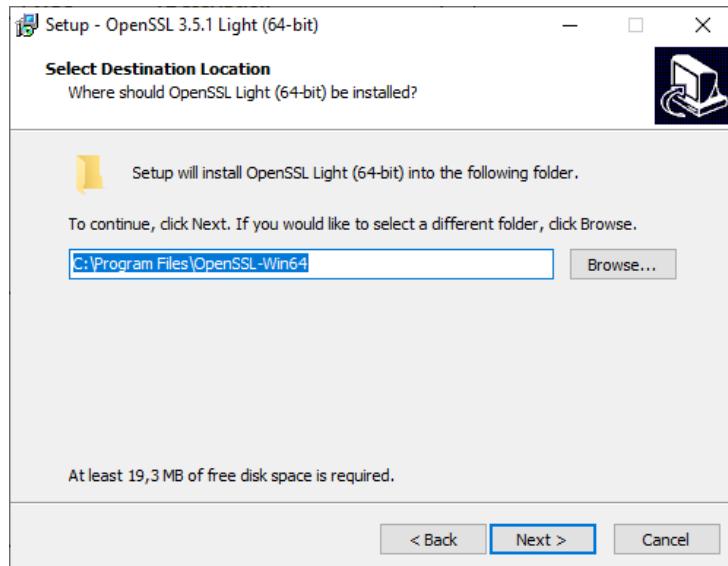
ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

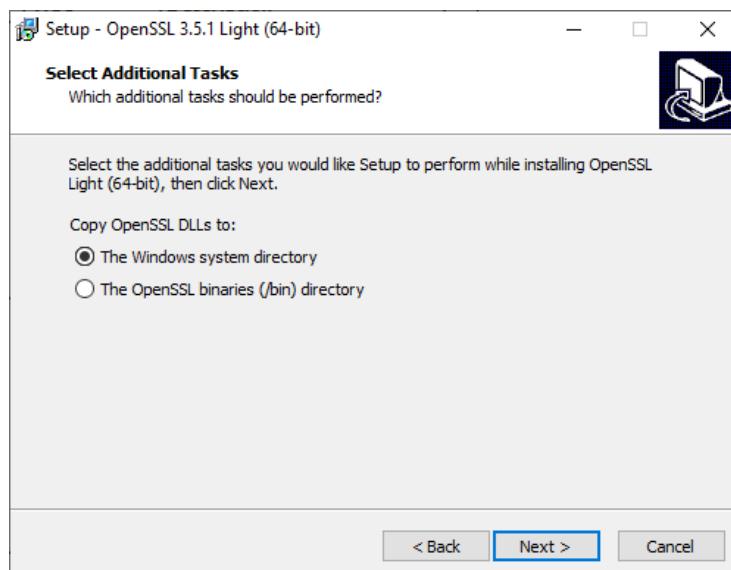


BASES DE DATOS

(TDSD353)



- Agregar al PATH del sistema:** Se seleccionó para agregar automáticamente OpenSSL al PATH de Windows, facilitando su ejecución desde la terminal.



- Confirmar la configuración e instalar:** Una vez configuradas las opciones, se procedió con la instalación.

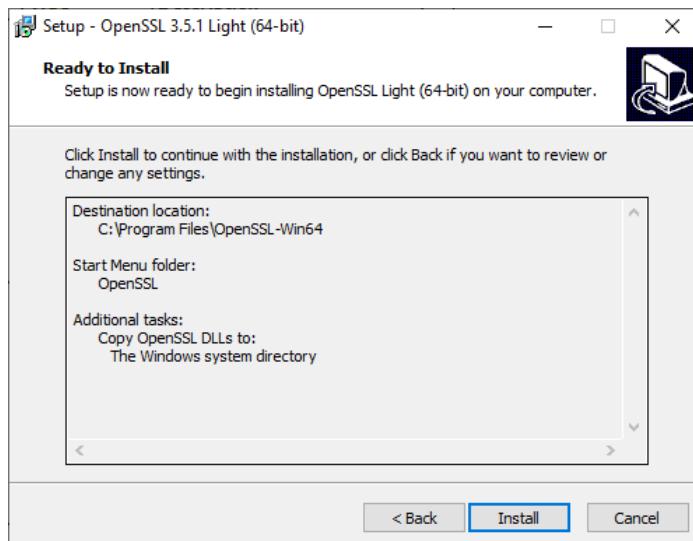


ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)



- **Confirmar la instalación:** Para comprobar que OpenSSL se haya instalado correctamente, se ejecutó el comando: openssl versión.

```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.6093]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Usuario>openssl version
OpenSSL 3.5.1 1 Jul 2025 (Library: OpenSSL 3.5.1 1 Jul 2025)

C:\Users\Usuario>
```

- **Generación de certificados:** Se generaron los archivos server.key (clave privada) y server.crt (certificado público) mediante los siguientes comandos:

```
openssl req -new -x509 -days 365 -nodes -text -out server.crt -keyout
server.key
```



ESCUELA POLITÉCNICA NACIONAL



ESCUELA DE FORMACIÓN DE TECNÓLOGOS

BASES DE DATOS

(TDSD353)

- **Ubicación de certificados:** Los certificados fueron copiados a la carpeta “data” del servidor PostgreSQL, para que puedan ser utilizados en la configuración del servicio.

Nombre	Fecha de modificación	Tipo
base	02/08/2025 10:38	Carpetas de archivos
global	02/08/2025 23:28	Carpetas de archivos
log	02/08/2025 23:27	Carpetas de archivos
pg_commit_ts	02/05/2025 17:39	Carpetas de archivos
pg_dynshmem	02/05/2025 17:39	Carpetas de archivos
pg_logical	02/08/2025 23:28	Carpetas de archivos
pg_multixact	02/05/2025 17:39	Carpetas de archivos
pg_notify	02/05/2025 17:39	Carpetas de archivos
pg_replslot	02/05/2025 17:39	Carpetas de archivos
pg_serial	02/05/2025 17:39	Carpetas de archivos
pg_snapshots	02/05/2025 17:39	Carpetas de archivos
pg_stat	26/07/2025 18:40	Carpetas de archivos
pg_stat_tmp	02/05/2025 17:39	Carpetas de archivos
pg_subtrans	02/05/2025 17:39	Carpetas de archivos
pg_tblspc	02/05/2025 17:39	Carpetas de archivos
pg_twophase	02/05/2025 17:39	Carpetas de archivos
pg_wal	02/08/2025 15:52	Carpetas de archivos
pg_xact	02/05/2025 17:39	Carpetas de archivos
current_logfiles	02/08/2025 23:27	Archivos
pg_hba	02/05/2025 17:39	Archivo CONF
pg_ident	02/05/2025 17:39	Archivo CONF
PG_VERSION	02/05/2025 17:39	Archivo
postgresql.auto	02/05/2025 17:39	Archivo CONF
postgresql	02/08/2025 23:21	Archivo CONF
postmaster.opts	02/08/2025 23:27	Archivo OPTS
postmaster.pid	02/08/2025 23:28	Archivo PID
server	02/08/2025 23:08	Certificado de seg...
server.key	02/08/2025 23:08	Archivo KEY

- **Configuración en postgresql.conf:** Se habilitó el uso de SSL modificando el archivo postgresql.conf con las siguientes líneas:
ssl = on
ssl cert file = 'server.crt'



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

ssl_key_file = 'server.key'

```
# - SSL -  
  
#ssl = on  
#ssl_ca_file = ''  
#ssl_cert_file = 'server.crt'  
#ssl_crl_file = ''  
#ssl_crl_dir = ''  
#ssl_key_file = 'server.key'  
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'  
#ssl_prefer_server_ciphers = on
```

Nota: Debemos asegurarnos de que las líneas agregadas o modificadas en el documento, no se encuentren comentadas o con # al inicio, si no, no se reconocerán los cambios.

- **Archivo pg_hba.conf:** Se agregaron las siguientes líneas al archivo para forzar el uso de SSL en conexiones locales:

```
# Permitir conexiones SSL desde cualquier IP con md5  
(contraseña)  
hostssl all all 0.0.0.0/0 md5  
  
# Permitir conexiones sin SSL solo desde localhost con md5  
host all all 127.0.0.1/32 md5
```

- **Reinicio el servidor:** Después de realizar los cambios, se reinició el servicio de PostgreSQL.



BASES DE DATOS (TDSD353)

```
pgAdmin 4
File Object Tools Edit View Window Help
Welcome postgres/postgres@PostgreSQL 17 X
Microsoft Windows [Versión 10.0.19045.6093]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Documentos\Emy\EPN\3er Semestre\Bases de Datos\PostgreSQL\pgAdmin 4\runtime\psql.exe=postgres user=postgres sslmode=prefer connect_timeout=10" 2>>&1
psql (17.4)
WARNING: Console code page (850) differs from Windows code page (932)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# SELECT pg_reload_conf();
pg_reload_conf
-----
t
(1 row)
```

- **Verificación del uso de SSL/TLS:** Finalmente, se comprobó que la conexión estaba utilizando SSL ejecutando la siguiente instrucción:

```
D:\Documentos\Emy\EPN\3er Semestre\Bases de Datos\PostgreSQL>psql "host=127.0.0.1 dbname=postgres user=postgres sslmode=require"
Contraseña para usuario postgres:
psql (17.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado, ALPN: postgresql)
Digite «help» para obtener ayuda.

postgres=#
```

También podemos comprobarlo ejecutando la siguiente consulta desde un script en pgAdmin:

	pid	ssl	version	cipher	bits	client_dn	client_serial	issuer_dn
	integer	boolean	text	text	integer	text	numeric	text
1	20892	true	TLSv1.3	TLS_AES_256_GCM_SHA384	256	[null]	[null]	[null]

5. Registro de intentos fallidos

- Se crea una tabla que registre el usuario y contraseña que fueron ingresadas pero erróneamente, también guarda la hora en que ocurrió el



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

evento y por último la razón de porqué no se pudo ingresar correctamente con las credenciales.

```
-- Tabla de respaldo para el registro de intentos fallidos
CREATE TABLE intentos_fallidos (
    id SERIAL PRIMARY KEY,
    usuario int,
    contrasenia_ingresada TEXT,
    fecha_intento TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    motivo TEXT
);

CREATE OR REPLACE FUNCTION validar_login(usuario_input int, contrasena_input TEXT)
RETURNS BOOLEAN
LANGUAGE plpgsql
AS $$
DECLARE
    hash_guardado TEXT;
BEGIN
    -- Buscar la contraseña encriptada del usuario por su id
    SELECT "contraseña" INTO hash_guardado
    FROM usuarios
    WHERE id = usuario_input;

    IF NOT FOUND THEN
        INSERT INTO intentos_fallidos(usuario, contrasenia_ingresada, motivo)
        VALUES (usuario_input, contrasena_input, 'Usuario no encontrado');
        RETURN FALSE;
    END IF;

    -- Comparar contraseñas encriptadas
    IF NOT (crypt(contrasena_input, hash_guardado) = hash_guardado) THEN
        INSERT INTO intentos_fallidos(usuario, contrasenia_ingresada, motivo)
        VALUES (usuario_input, contrasena_input, 'Contraseña incorrecta');
        RETURN FALSE;
    END IF;

    RETURN TRUE;
END;
$$;
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
104  SELECT validar_login('1000', 'user123');
105  select * from intentos_fallidos;
```

Data Output Messages Notifications

	id [PK] integer	usuario integer	contrasenia_ingresada text	fecha_intento timestamp without time zone	motivo text
1	1	2	emi1234	2025-08-03 01:22:34.565618	Contraseña incorrecta
2	2	2	emi132	2025-08-03 01:22:40.534856	Contraseña incorrecta
3	3	1	user123	2025-08-03 01:22:51.804987	Contraseña incorrecta
4	4	1000	user123	2025-08-03 01:23:00.471092	Usuario no encontrado

6. Validaciones de entrada usando expresiones regulares

- Trigger que valida el correo en la tabla personas

```
CREATE OR REPLACE FUNCTION validar_correo()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.correo !~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' THEN
        RAISE EXCEPTION 'Correo inválido: %', NEW.correo;
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER trg_validar_correo
BEFORE INSERT OR UPDATE ON personas
FOR EACH ROW EXECUTE FUNCTION validar_correo();
```

Ejemplo de uso:

```
126  -- Ejemplo de uso
127  INSERT INTO personas (correo) VALUES ('email_invalido');      -- Error
```

Data Output Messages Notifications

```
ERROR: Correo inválido: email_invalido
CONTEXT: función PL/pgSQL validar_correo() en la línea 4 en RAISE

SQL state: P0001
```

- Trigger que valida la contraseña tenga solo letras, números y algunos símbolos. Debe ser mínimo de 6 caracteres.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
CREATE OR REPLACE FUNCTION validar_contrasena()
RETURNS TRIGGER
LANGUAGE plpgsql;
AS $$
BEGIN
    IF NEW."contraseña" !~ '^[A-Za-z0-9@#$%^&+=]{6,}' THEN
        RAISE EXCEPTION USING MESSAGE = 'Contraseña inválida: debe tener al menos 6 caracteres';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER trg_validar_contrasena
BEFORE INSERT OR UPDATE ON usuarios
FOR EACH ROW EXECUTE FUNCTION validar_contrasena();
```

Ejemplo de uso:

```
145 -- Ejemplo de uso
146 INSERT INTO usuarios ("contraseña") VALUES ('abc');
147 |

Data Output Messages Notifications
ERROR: Contraseña inválida: debe tener al menos 6 caracteres y solo letras, números o @#$%^&+=
CONTEXT: función PL/pgSQL validar_contrasena() en la línea 4 en RAISE
SQL state: P0001

148 INSERT INTO usuarios ("contraseña") VALUES ('abc***#');
149

Data Output Messages Notifications
ERROR: Contraseña inválida: debe tener al menos 6 caracteres y solo letras, números o @#$%^&+=
CONTEXT: función PL/pgSQL validar_contrasena() en la línea 4 en RAISE
SQL state: P0001
```

7. Revisión del historial de roles asignados y auditoría de privilegios activos

Para controlar quién tiene qué permisos, se hizo lo siguiente:

- Se creó una tabla llamada `historial_roles` para guardar cuándo y quién asignó o quitó roles a los usuarios.



BASES DE DATOS (TDSD353)

```
CREATE TABLE historial_roles (
    id SERIAL PRIMARY KEY,
    usuario_rol varchar(50) NOT NULL,
    rol varchar(50) NOT NULL,
    accion varchar(50) NOT NULL,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    usuario_encargado varchar(50) NOT NULL
);
```

- Se hicieron funciones que facilitan dar o quitar roles a los usuarios y que además registran esos cambios en la tabla de historial.

```
CREATE OR REPLACE FUNCTION asignar_rol(usuario TEXT, rol TEXT) RETURNS VOID
LANGUAGE plpgsql
AS $$
BEGIN
    EXECUTE FORMAT('GRANT %I TO %I', rol, usuario);
    INSERT INTO historial_roles(usuario_rol, rol, accion, usuario_encargado)
    VALUES (usuario, rol, 'ASIGNADO', current_user);
END;
$$;

CREATE OR REPLACE FUNCTION revocar_rol(usuario TEXT, rol TEXT) RETURNS VOID
LANGUAGE plpgsql
AS $$
BEGIN
    EXECUTE FORMAT('REVOKE %I FROM %I', rol, usuario);
    INSERT INTO historial_roles(usuario_rol, rol, accion, usuario_encargado)
    VALUES (usuario, rol, 'REVOCADO', current_user);
END;
$$;
```

- Se crearon usuarios (de ejemplo) con permiso para iniciar sesión.

```
CREATE ROLE alejo WITH LOGIN PASSWORD 'alejo123';
CREATE ROLE emy WITH LOGIN PASSWORD 'emy123';
CREATE ROLE pedro WITH LOGIN PASSWORD 'pedro123';
```

- Se usaron las funciones para asignar o revocar roles a esos usuarios, dejando un registro automático.

```
SELECT asignar_rol('alejo', 'administrador');
SELECT asignar_rol('emy', 'operador');
SELECT revocar_rol('pedro', 'empreendedor');
```

- Se verificó que la tabla de historial guarda correctamente los cambios.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
select * from historial_roles;
```

Output Messages Notifications

id [PK] integer	usuario_rol character varying (50)	rol character varying (50)	accion character varying (50)	fecha timestamp without time zone	usuario_encargado character varying (50)
1	alejo	administrador	ASIGNADO	2025-08-03 02:13:12.671175	postgres
2	emy	operador	ASIGNADO	2025-08-03 02:13:12.671175	postgres
3	pedro	emprendedor	ASIGNADO	2025-08-03 02:13:12.671175	postgres
4	pedro	emprendedor	REVOCADO	2025-08-03 02:16:56.642672	postgres

- Se revisaron los roles activos que tiene cada usuario.

```
203 v SELECT
204      pg_get_userbyid(member) AS usuario,
205      pg_get_userbyid(roleid) AS rol
206  FROM pg_auth_members
207 ORDER BY usuario;
208
```

Data Output Messages Notifications

	usuario name	rol name
1	alejo	administrador
2	emy	operador
3	pg_monitor	pg_read_all_settings
4	pg_monitor	pg_read_all_stats
5	pg_monitor	pg_stat_scan_tables

8. Auditoría

1. Tabla log_acciones completa: usuario, IP, acción, tabla, ID afectado.

```
-- Tabla de respaldo completa
CREATE TABLE log_acciones(
    id SERIAL PRIMARY KEY,
    usuario varchar(50) NOT NULL,
    ip text,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    accion text,
    tabla text,
    id_afectado int
)
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

2. Función para insertar cambios en log_acciones:

```
-- Función
CREATE OR REPLACE FUNCTION auditoria_tablas()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$

DECLARE
    ID_afectado int;
BEGIN
    ID_afectado := COALESCE(NEW.id, OLD.id);
    INSERT INTO log_acciones(usuario, ip, accion, tabla, id_afectado)
    VALUES (
        current_user,
        inet_client_addr()::TEXT,
        TG_OP,
        TG_TABLE_NAME,
        ID_afectado
    );
    IF TG_OP = 'DELETE' THEN
        RETURN OLD;
    ELSE
        RETURN NEW;
    END IF;
END;
$$;
```

3. Triggers con todas las operaciones (Insert, Delete, Update)

- Tabla "resultados"

```
-- Trigger para la tabla resultados
CREATE TRIGGER trg_aud_resultados
AFTER INSERT OR UPDATE OR DELETE ON resultados
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "mentores"

```
-- Trigger para la tabla mentores
CREATE TRIGGER trg_aud_mentores
AFTER INSERT OR UPDATE OR DELETE ON mentores
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "reportes"



BASES DE DATOS (TDSD353)

```
-- Trigger para la tabla reportes
CREATE TRIGGER trg_aud_reportes
AFTER INSERT OR UPDATE OR DELETE ON reportes
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "estadisticas_idea"

```
-- Trigger para la tabla estadisticas_idea
CREATE TRIGGER trg_aud_estadisticas_idea
AFTER INSERT OR UPDATE OR DELETE ON estadisticas_idea
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "personas"

```
-- Trigger para la tabla personas
CREATE TRIGGER trg_aud_personas
AFTER INSERT OR UPDATE OR DELETE ON personas
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "usuarios"

```
-- Trigger para la tabla usuarios
CREATE TRIGGER trg_aud_usuarios
AFTER INSERT OR UPDATE OR DELETE ON usuarios
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "categorias"

```
-- Trigger para la tabla categorias
CREATE TRIGGER trg_aud_categorias
AFTER INSERT OR UPDATE OR DELETE ON categorias
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "fases_proyecto"

```
-- Trigger para la tabla fases_proyecto
CREATE TRIGGER trg_aud_fases_proyecto
AFTER INSERT OR UPDATE OR DELETE ON fases_proyecto
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "tipo_estados"



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- Trigger para la tabla tipo_estados  
CREATE TRIGGER trg_aud_tipo_estados  
AFTER INSERT OR UPDATE OR DELETE ON tipo_estados  
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "ideas_negocio"

```
-- Trigger para la tabla ideas_negocio  
CREATE TRIGGER trg_aud_ideas_negocio  
AFTER INSERT OR UPDATE OR DELETE ON ideas_negocio  
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "mentorias"

```
-- Trigger para la tabla mentorias  
CREATE TRIGGER trg_aud_mentorias  
AFTER INSERT OR UPDATE OR DELETE ON mentorias  
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "observaciones"

```
-- Trigger para la tabla observaciones  
CREATE TRIGGER trg_aud_observaciones  
AFTER INSERT OR UPDATE OR DELETE ON observaciones  
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Tabla "avance_fases"

```
-- Trigger para la tabla avance_fases  
CREATE TRIGGER trg_aud_avance_fases  
AFTER INSERT OR UPDATE OR DELETE ON avance_fases  
FOR EACH ROW EXECUTE FUNCTION auditoria_tablas();
```

- Comprobación



BASES DE DATOS

(TDSD353)

```
102 -- Comprobación
103 select * from personas;
104 insert into personas(nombres, apellidos, correo, telefono, direccion)
105 values
106 ('María José', 'Peñafiel Torres', 'majo@gmail.com', '0995471823', 'La Luz');
107
108 select * from log_acciones;
```

Data Output Messages Notifications

	id [PK] integer	usuario character varying (50)	ip text	fecha timestamp without time zone	accion text	tabla text	id_afectado integer
1	1	postgres	::1/128	2025-08-03 10:31:26.7615	INSERT	personas	804

4. Reportes por usuario, acción, módulo o fecha.

- Vista general de la auditoría

```
-- vista general
CREATE OR REPLACE VIEW vista_log_acciones_general AS
SELECT
    id,
    usuario,
    rol,
    ip,
    fecha,
    accion,
    tabla,
    id_afectado
FROM log_acciones;
-- Comprobación
select * from vista_log_acciones_general;
```

- Vista por usuario

```
-- vista por usuario
CREATE OR REPLACE VIEW vista_log_acciones_usuario AS
SELECT * FROM log_acciones
ORDER BY usuario, fecha DESC;
-- Comprobación
SELECT * FROM vista_auditioria_por_usuario WHERE usuario = 'admin';
```

- Vista por tabla



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- vista por tabla
CREATE OR REPLACE VIEW vista_log_acciones_tabla AS
SELECT * FROM log_acciones
ORDER BY tabla, fecha DESC;
-- Comprobación
SELECT * FROM vista_log_acciones_tabla WHERE tabla = 'personas';

```

utput Messages Notifications

id	usuario	ip	fecha	accion	tabla	id_afectado	rol
10	postgres	::1/128	2025-08-03 11:00:32.876089	DELETE	personas	804	none
3	postgres	::1/128	2025-08-03 10:44:33.801283	INSERT	personas	804	none

- Vista por fecha

```
-- vista por fecha (últimos 7 días)
CREATE OR REPLACE VIEW vista_log_acciones_fecha AS
SELECT * FROM log_acciones
WHERE fecha >= current_date - INTERVAL '7 days'
ORDER BY fecha DESC;
-- Comprobación
select * from vista_log_acciones_fecha;
```

9. Respaldo y Restauración

1. **Backup en caliente:** Se realiza mientras la base de datos está en uso o activa, sin necesidad de detener el servicio.

- Abrimos la terminal
- Escribimos el comando:

pg_dump -U postgres -d bdd_juveniles -F c -f backup_juveniles1.backup

Donde:

- U Usuario con permisos de backup
- d Nombre de la base de datos
- F c Formato personalizado
- f Nombre del archivo de salida



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
D:\>pg_dump -U postgres -d bdd_juveniles -F c -f backup_juveniles1.backup
Contraseña:

D:\>dir
   El volumen de la unidad D es Almacenamiento
   El número de serie del volumen es: D6C0-DCFB

   Directorio de D:\

   03/08/2025  11:28           184.031 backup_juveniles1.backup
```

- **Adicionales:**

Backup Comprimido:

```
pg_dump -U postgres -d bdd_juveniles -F c -Z 9 -f backup_juveniles_compressed.backup
```

Donde -Z 9 es la máxima compresión

Backup de una sola tabla:

```
pg_dump -U postgres -d bdd_juveniles -t ideas_negocio -f backup_ideas_negocio.backup
```

Comprobación:

```
D:\>pg_dump -U postgres -d bdd_juveniles -F c -Z 9 -f backup_juveniles_compressed.backup
Contraseña:

D:\>pg_dump -U postgres -d bdd_juveniles -t ideas_negocio -f backup_ideas_negocio.backup
Contraseña:

D:\>dir
   El volumen de la unidad D es Almacenamiento
   El número de serie del volumen es: D6C0-DCFB

   Directorio de D:\

   03/08/2025  11:36           134.967 backup_ideas_negocio.backup✓
   03/08/2025  11:28           184.031 backup_juveniles1.backup
   03/08/2025  11:36           182.856 backup_juveniles_compressed.backup✓
```

2. Backup en frío: La base de datos debe estar detenida. Se copia

directamente el directorio de datos.

- **Detener PostgreSQL:** Buscamos servicios en Windows y paramos el servicio:

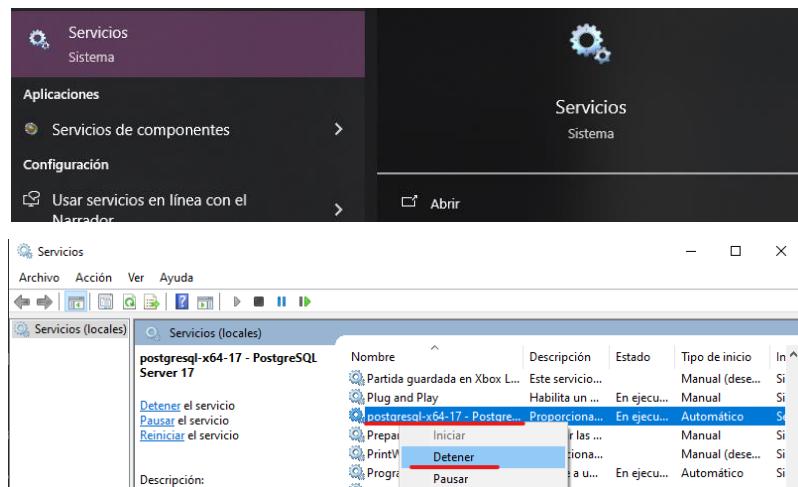


ESCUELA POLITÉCNICA NACIONAL

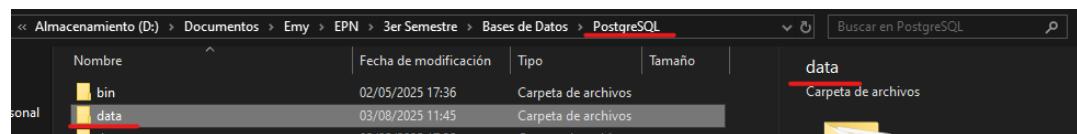
ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)



- Copiamos la carpeta “data”



"D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data" Semestre\Bases de

Datos\PostgreSQL\data" "D:\backup_frio" /E /H /C /I /K /Y

Dónde:

/E: Copia subdirectorios (incluyendo vacíos).

/H: Copia archivos ocultos

/C: Continúa aunque haya errores.

/I: Asume que el destino es una carpeta.

/K: Conserva atributos (solo lectura, etc.).

/Y: Sobrescribe sin preguntar.

```
C:\WINDOWS\system32>xcopy "D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data"
"D:\backup_frio" /E /H /C /I /K /Y
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\current_logfiles
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\pg_hba.conf
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\pg_ident.conf
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\PG_VERSION
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\postgresql.auto.conf
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\postgresql.conf
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\postmaster.opts
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\server.crt
D:\Documentos\Em y\EPN\3er Semestre\Bases de Datos\PostgreSQL\data\server.key
Acceso denegado.
```

- **Reiniciar PostgreSQL:** Volvemos a iniciar el servicio de PostgreSQL desde Servicios



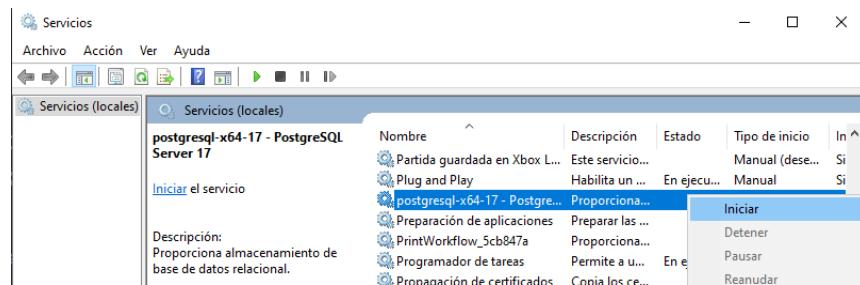
ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)



3. Restauración desde Consola:

- **Restauración en una base de datos nueva:**

1. Ingresar a la consola CMD y crear una nueva base de datos con:
createdb -U postgres bdd_respaldo. Luego ingresamos la contraseña.

```
C:\WINDOWS\system32>createdb -U postgres bdd_respaldo_juveniles  
Contraseña:
```

2. Restauramos todo el .backup:

```
pg_restore -U postgres -d bdd_respaldo -F c "D:\backup_juveniles1.backup".  
E ingresamos la contraseña
```

```
C:\WINDOWS\system32>pg_restore -U postgres -d bdd_respaldo_juveniles -F c  
"D:\backup_juveniles1.backup"  
Contraseña:
```

3. Comprobamos que se hayan pasado las tablas a la base de datos de respaldo:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
C:\WINDOWS\system32>psql -U postgres -d bdd_respaldo_juveniles
Contraseña para usuario postgres:
psql (17.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.

Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado, ALPN: postgresql)
Digite «help» para obtener ayuda.

bdd_respaldo_juveniles=# \dt
           Listado de relaciones
   Esquema |     Nombre      | Tipo | Dueño
-----+-----+-----+-----+
 public | avance_fases | tabla | postgres
 public | categorias   | tabla | postgres
 public | estadisticas_idea | tabla | postgres
 public | fases_proyecto | tabla | postgres
 public | historial_roles | tabla | postgres
 public | ideas_negocio | tabla | postgres
 public | intentos_fallidos | tabla | postgres
 public | log_acciones | tabla | postgres
 public | logs_sistema | tabla | postgres
 public | mentores | tabla | postgres
 public | mentorias | tabla | postgres
 public | observaciones | tabla | postgres
 public | personas | tabla | postgres
 public | reportes | tabla | postgres
 public | resultados | tabla | postgres
 public | tipo_estados | tabla | postgres
 public | usuarios | tabla | postgres
(17 filas)
```

10. Seguridad ante SQL Injection

- Simulación de formulario vulnerable (' OR '1'='1)



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
1 -- Ejemplo de login vulnerable
2 v SELECT p.*, u.* FROM personas p
3 JOIN usuarios u ON p.id = u.persona_id
4 WHERE u.contraseña = '' OR '1'='1';
```

Data Output Messages Notifications

	dirección	id	persona_id	contraseña	tipo_usuario
	text	integer	integer	text	tipo_usuario_enum
1	San Sebastian	102	101	\$2a\$08\$wC7PO/lfR1WvrHLithZNGu91nvFnc17jnu240NYiPN3lxL.F2qa3q	mentor
2	San Blas	103	102	\$2a\$08\$NYVeVTlxjhST57nAUGTvKOT5BX.3Aj2nKBMAbnTwEmdvkCWto...	emprendedor
3	San Marcos	104	103	\$2a\$08\$JS8FSYXVJcvQW7itHyb7v.lWEojL1csR58m3kLTnsty5h5nRUQM5u	mentor
4	San Roque	105	104	\$2a\$08\$0pGrPO4/WgnMcP0SeumT0W4frNmTThB8GLnMy4Z5BW22L...	mentor
5	La Kennedy	106	105	\$2a\$08\$rdgW7GAY5jBXJTZOTWhU.urhVr3KJzr00vHab2XM3exSbQgDb...	emprendedor
6	La Ecuatoriana	107	106	\$2a\$08\$D8Teijv6Md2dDoiBU/m8ceA3PxzbzPbCOVER1s7la4lLygho1QKe	emprendedor
7	La Ferroviaria	108	107	\$2a\$08\$ybPjLd43n08ToxkInHJLuleM5sc6qS1n.x750GPluT50rZ/w026	administrador
8	La Magdalena	109	108	\$2a\$08\$DKLi2GS.85mCxvCkZtYj00Ttg6sjw4I02Fr0bFmpq/KiqxsAqhUe	emprendedor
9	La Mena	110	109	\$2a\$08\$BV5vRXiZXIX6dHSABuHiD.YK66ofkv.7SXnEFpjpro05z.pHVs...	mentor
10	La Colmena	111	110	\$2a\$08\$DMfPBeb0bld481wRUW9iNUuUnQFcmbtKnJD9hwCV9zeRak/Iu...	administrador
11	La Vicentina	112	111	\$2a\$08\$NItpGoHfNJZ9RphoQSXlt.GPnQZSAT2HR/forh4dqPhzlisDCoSLu	mentor
12	La Forestal	113	112	\$2a\$08\$h5BtzXtfFMbscrr3E7kEte6Xo5A049Xv/h8la0qRff.2yS4fZzsS	administrador
13	La Granja	114	113	\$2a\$08\$3Xz3lpFh5SEcw3.pQv4xuqq0VtkLMmjISLN3ZN.XH2xFryCk9...	emprendedor
14	La Armenia	115	114	\$2a\$08\$GKTGpLiviNGHSC24tXluuqtYESsWsIGU9m1U2LXn293NppoW2...	mentor
15	La Concepcion	116	115	\$2a\$08\$wl0MIuy4z1TeZmln7msC2eTf2DNXu7gncWeZ1GJcKn2HtmmXT...	emprendedor
16	La Carolina	117	116	\$2a\$08\$7MxhZqiMTDxJmw5GQnD040uLTPjwH7Rim2ljYrNxhUYGfgTVF...	administrador
17	La Pradera	118	117	\$2a\$08\$YBF9clUsj.GrRoD2R6lqOpMFlrtEH3QHevA2u69woeDwj.O1DH2	mentor
18	La Pradera	119	118	\$2a\$08\$9o3bVL95wa9SznL6gAds.Y13K9yII9BYNEKvD0CGj//FNuEWyP02	administrador
19	La Mariscal	120	119	\$2a\$08\$08tfz/SWkCRPAIW1uqYRw4kOvh6jE57Bs4qGb59Ko2puK6CE167T...	emprendedor
20	La Floresta	121	120	\$2a\$08\$76bnNszoJx/TR1qgOr92mu20Dvl5WOW2mhJpZkWa8HULM94T...	administrador
21	Conocoto	122	121	\$2a\$08\$mrccim6bSc5yqarxSSB3cuRETNuZTRml1.LMjwlvjJDv/vWTCcA...	emprendedor
22	Quitumbe	123	122	\$2a\$08\$FH.H4VId1hL7.U5gUIJpeFKhrUqnlgOJplf7SA32hBBmCxjlHKhy	emprendedor
23	Conocoto	124	123	\$2a\$08\$qNZpe/T5l0WduvqduCL5Z.sjbItataEaBDuryg7ALBiPLRsITzBLK	administrador

- **Protección con consultas preparadas.**

```
6 -- Consulta segura
7 v PREPARE consulta_login_seguro(text, text) AS
8 SELECT p.*, u.*
9 FROM personas p
10 JOIN usuarios u ON p.id = u.persona_id
11 WHERE p.correo = $1 AND u.contraseña = crypt($2, u.contraseña);
12
13 EXECUTE consulta_login_seguro('emy@gmail.com', 'emi123');
```

Data Output Messages Notifications

	nombr... character varying (100)	apellid... character varying (100)	correo character varying (100)	telefono character varying (20)	direccion text	id integer	persona_id integer	contraseña text
1	Emily Alejandra	Galeas Tingo	emy@gmail.com	0987458521	Carcelen	2	2	\$2a\$08\$RMcS...

- **Validaciones previas en procedimientos y vistas.**

1. Función Segura para login



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
18  -- Función segura para login
19  DROP FUNCTION login_seguro;
20  CREATE OR REPLACE FUNCTION login_seguro(p_correo text, p_password text)
21    RETURNS TABLE (
22      id_persona integer,
23      correo character varying(100),
24      tipo_usuario tipo_usuario_enum
25    )
26  AS $$
27  BEGIN
28    RETURN QUERY
29    SELECT p.id, p.correo, u.tipo_usuario
30    FROM personas p
31    JOIN usuarios u ON p.id = u.persona_id
32    WHERE p.correo = p_correo
33      AND u.contraseña = crypt(p_password, u.contraseña);
34  END;
35  $$ LANGUAGE plpgsql;
36
37  -- Comprobación
38  SELECT * FROM login_seguro('alejo@gmail.com', 'alejo123');
```

Data Output Messages Notifications

Showing rows: 1 to 1

	id_persona	correo	tipo_usuario
	integer	character varying	tipo_usuario_enum
1	1	alejo@gmail.com	administrador

2. Vista que muestra solo la información general sin exponer contraseñas

```
40  -- Vista que solo muestra información general sin exponer contraseñas
41  CREATE OR REPLACE VIEW vista_usuarios_activos AS
42  SELECT p.id AS persona_id, p.correo, u.tipo_usuario
43  FROM personas p
44  JOIN usuarios u ON p.id = u.persona_id;
45
46  -- Comprobación
47  select * from vista_usuarios_activos;
```

Data Output Messages Notifications

Showing rows: 1 to 802

	persona_id	correo	tipo_usuario
	integer	character varying (100)	tipo_usuario_enum
1	101	rafael.zambrano@mail.com	mentor
2	102	mariana.miranda@mail.com	emprendedor
3	103	esteban.espinoza@mail.com	mentor
4	104	patricia.vera@mail.com	mentor
5	105	roberto.pena@mail.com	emprendedor
6	106	monica.figueroa@mail.com	emprendedor
7	107	eduardo.rivera@mail.com	administrador
8	108	lucia.navarro@mail.com	emprendedor

3. Trigger que previene la creación de personas con correos duplicados



BASES DE DATOS (TDSD353)

```
49 -- Trigger que previene la creación de personas con correos duplicados
50 ✓ CREATE OR REPLACE FUNCTION validar_correo_unico()
51 RETURNS TRIGGER AS $$ 
52 BEGIN
53   IF EXISTS (SELECT 1 FROM personas WHERE correo = NEW.correo) THEN
54     RAISE EXCEPTION 'El correo ya está registrado.';
55   END IF;
56   RETURN NEW;
57 END;
58 $$ LANGUAGE plpgsql;
59
60 -- Asocia el trigger a la tabla personas
61 ✓ CREATE TRIGGER trg_correo_unico
62 BEFORE INSERT ON personas
63 FOR EACH ROW
64 EXECUTE FUNCTION validar_correo_unico();
65
66 -- Comprobación
67 ✓ INSERT INTO personas (nombres, correo)
68 VALUES ('Emily A', 'emy@gmail.com');
```

Data Output Messages Notifications

ERROR: El correo ya está registrado.
CONTEXT: función PL/pgSQL validar_correo_unico() en la línea 4 en RAISE

SQL state: P0001

- **Documentación del ataque y su prevención.**

El SQL injection es un tipo de ataque cibernético donde se manipulan consultas a bases de datos SQL mediante la inserción de código SQL malicioso en campos de entrada de una aplicación web o servicio. Este ataque puede permitir a los atacantes acceder, modificar o eliminar datos sensibles, o incluso tomar el control total del sistema.

Este ataque ocurre cuando una aplicación web no valida correctamente los campos de entrada en un formulario, por ejemplo, de inicio de sesión, permitiendo que código SQL se incorpore en consultas a la base de datos. Por lo que los atacantes pueden usar esta vulnerabilidad para:

- **Robar datos:** Obtener información confidencial como contraseñas, datos de tarjetas de crédito, información personal de usuarios, etc.
- **Modificar datos:** Cambiar datos en la base de datos, alterar transacciones, etc.
- **Eliminar datos:** Borrar información de la base de datos.
- **Tomar control del sistema:** En casos más graves, la inyección SQL puede permitir a un atacante obtener acceso root al servidor, dando control completo sobre el sistema



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

Para evitar este tipo de ataques, se pueden implementar las siguientes medidas:

- **Validación estricta de datos de entrada:** Uso de expresiones regulares y validaciones previas para filtrar y controlar los datos que recibe el sistema.
- **Consultas preparadas (Prepared Statements):** Separar el código SQL de los datos ingresados por el usuario, evitando la ejecución de comandos maliciosos dentro de la consulta.
- **Restricción de permisos:** Asignar a cada rol o usuario solo los permisos necesarios para minimizar daños en caso de explotación.
- **Auditoría y monitoreo:** Registrar intentos fallidos de acceso y actividad sospechosa para detectar y actuar frente a posibles ataques a tiempo.
- **Políticas de validación de datos entrantes.**

Las políticas para validar los datos que llegan al sistema son muy importantes porque ayudan a asegurar que solo se procese información correcta, segura y que realmente se espera recibir. Estas políticas incluyen varios aspectos clave:

1. Verificar que los datos tengan el formato adecuado, por ejemplo, que un correo tenga un dominio válido, que números y letras estén en los campos correctos y que no se ingresen caracteres extraños.
2. Usar expresiones regulares que ayuden a filtrar y bloquear caracteres peligrosos que puedan injectar código malicioso.
3. Definir límites en la cantidad de caracteres que se pueden ingresar, para evitar que se pueda enviar información demasiado larga ya causar problemas en el sistema.
4. Limpiar y transformar datos para eliminar cualquier carácter especial que pueda ser interpretado como código.

Por lo que, si los datos no cumplen con estas reglas, el sistema rechaza la entrada y envía un mensaje claro al usuario para que pueda corregirlo.

11. Monitoreo y Rendimiento

Para asegurar que la base de datos funcione bien y evitar problemas de rendimiento, es importante hacer monitoreos regulares y analizar ciertos aspectos clave:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

1. Consulta de tamaño de tablas

```
1 -- Consulta de tamaño de tablas
2 SELECT
3     tablename,
4     pg_size_pretty(pg_total_relation_size(tablename::regclass)) AS tamaño_total
5 FROM
6     pg_tables
7 WHERE
8     schemaname = 'public'
9 ORDER BY
10    pg_total_relation_size(tablename::regclass) DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 17 | | Page No

	tablename name	tamaño_total text
1	ideas_negocio	400 kB
2	usuarios	328 kB
3	avance_fases	280 kB
4	personas	248 kB
5	mentorias	232 kB
6	estadisticas_id...	200 kB
7	observaciones	176 kB
8	resultados	144 kB
9	reportes	120 kB
10	mentores	56 kB
11	categorias	48 kB
12	fases_proyecto	48 kB
13	tipo_estados	40 kB
14	intentos_fallidos	32 kB
15	log_acciones	32 kB
16	historial_roles	24 kB
17	logs_sistema	16 kB

2. Consulta de tamaño de índices



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
12  -- Consulta de tamaño de índices
13  SELECT
14      indexname,
15      tablename,
16      pg_size.pretty(pg_relation_size(indexname::regclass)) AS tamaño
17  FROM pg_indexes
18  WHERE schemaname = 'public'
19  ORDER BY pg_relation_size(indexname::regclass) DESC;
```

Data Output Messages Notifications

The screenshot shows a PostgreSQL query tool interface. At the top, there is a SQL editor with the following code:

```
-- Consulta de tamaño de índices
SELECT
    indexname,
    tablename,
    pg_size.pretty(pg_relation_size(indexname::regclass)) AS tamaño
FROM pg_indexes
WHERE schemaname = 'public'
ORDER BY pg_relation_size(indexname::regclass) DESC;
```

Below the editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is selected, showing the results of the query in a grid format:

	indexname name	tablename name	tamaño text
15	idx_estadisticas_completo	estadisticas_id...	40 kB
16	idx_mentorias_completo	mentorias	40 kB
17	idx_ideas_negocio_estado_id	ideas_negocio	40 kB
18	idx_estadisticas_avance_fase	estadisticas_id...	40 kB
19	personas_pkey	personas	40 kB
20	reportes_pkey	reportes	40 kB
21	idx_ideas_usuario_id	ideas_negocio	40 kB
22	log_acciones_pkey	log_acciones	16 kB
23	categorias_nombre_key	categorias	16 kB
24	categorias_pkey	categorias	16 kB
25	fases_proyecto_fase_key	fases_proyecto	16 kB
26	fases_proyecto_pkey	fases_proyecto	16 kB
27	mentores_pkey	mentores	16 kB
28	tipo_estados_pkey	tipo_estados	16 kB
29	idx_ideas_estado	ideas_negocio	16 kB
30	idx_mentorias_estado	mentorias	16 kB
31	idx_tipo_estados_id	tipo_estados	16 kB

Showing rows: 1 to 34

3. Consulta de uso de disco

```
21  -- Consulta de uso de disco
22  SELECT
23      pg_size.pretty(pg_database_size(current_database())) AS tamaño_base_de_datos,
```

Data Output Messages Notifications

The screenshot shows a PostgreSQL query tool interface. At the top, there is a SQL editor with the following code:

```
-- Consulta de uso de disco
SELECT
    pg_size.pretty(pg_database_size(current_database())) AS tamaño_base_de_datos,
```

Below the editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is selected, showing the results of the query in a grid format:

	tamaño_base_de_datos text
1	11 MB

Showing rows: 1 to 1

Page No: 1



BASES DE DATOS (TDSD353)

4. Evaluación de consultas más lentas

- Necesitamos activar la extensión pg_stat_statements

```
-- Activar la extensión
CREATE EXTENSION pg_stat_statements;
```

- Luego editamos en el archivo postgres.conf ubicado en data:

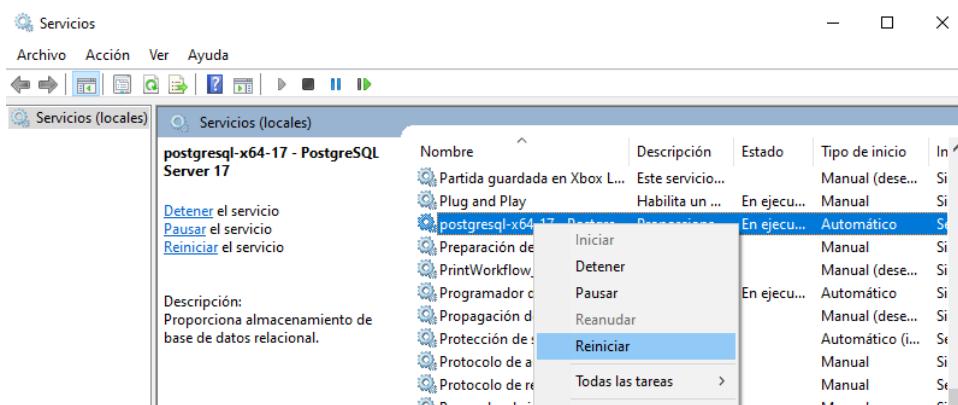
```
postgresql: Bloc de notas
Archivo Edición Formato Ver Ayuda
#local_preload_libraries = ''
#session_preload_libraries = ''
shared_preload_libraries = 'pg_stat_statements'          #
(change requires restart)
#jit_provider = 'llvmjit'                                # JIT library to
use

# - Other Defaults -

#dynamic_library_path = '$libdir'
#gin_fuzzy_search_limit = 0

...
```

- Reiniciamos el servidor



- Ejecutamos la consulta:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
30  -- Consulta
31  < SELECT
32    query,
33    calls,
34    total_exec_time,
35    mean_exec_time,
36    rows
37  FROM pg_stat_statements
38  ORDER BY mean_exec_time DESC;
39
```

Data Output Messages Notifications

	query text	calls bigint	total_exec_time double precision	mean_exec_time double precision	rows bigint
1	SELECT	1	30.486800000000002	30.486800000000002	10
2	SELECT set_config(\$1,\$2,\$3) FROM pg_show_all...	1	6.2371	6.2371	1
3	SELECT set_config(\$1,\$2,\$3) FROM pg_show_all...	1	6.2242	6.2242	1
4	SELECT DISTINCT att.attname as name, att.attn...	1	0.6401	0.6401	49
5	SELECT at.attname, ty.typname, at.attnum	1	0.2422	0.2422	49
6	SELECT at.attname, at.attnum, ty.typname	1	0.146	0.146	0
7	SELECT	1	0.079	0.079	1
8	SELECT	1	0.0776	0.0776	1
9	SELECT	1	0.0694	0.0694	1
10	SELECT	1	0.0649999999999999	0.0649999999999999	1
11	SELECT CASE	1	0.0518	0.0518	1
12	SELECT	1	0.0276	0.0276	1
13	SELECT	1	0.0271000000000003	0.0271000000000003	1
14	SELECT oid, pg_catalog.format_type(oid, \$2) AS ...	1	0.0149	0.0149	3
15	SELECT CASE WHEN usesuper	1	0.0127999999999999	0.0127999999999999	1
16	SET client_min_messages=notice	1	0.006	0.006	0
17	SET DateStyle=ISO	1	0.0058	0.0058	0
18	SET LOCAL lc_c... ...o	1	0.0057	0.0057	0

5. Registro del uso de funciones, procedimientos, recursos

```
40  -- Registro del uso de funciones, procedimientos y recursos.
41  < SELECT
42    funcid::regprocedure AS funcion,
43    calls,
44    total_time,
45    self_time
46  FROM pg_stat_user_functions
47  ORDER BY total_time DESC;
```

Data Output Messages Notifications

	funcion regprocedure	calls bigint	total_time double precision	self_time double precision
1	login_seguro(text,text)	27	368.391	7.846
2	crypt(text,text)	27	360.543	360.543
3	validar_login(integer,text)	12	87.478	87.478
4	validar_correo_unico()	17	2.569	2.569
5	validar_correo()	0	0	0



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

12. Protección de Datos y Gestión Crítica

- Cifrado de datos sensibles: emails, contraseñas.

-- 1. Renombrar la columna original

```
ALTER TABLE personas RENAME COLUMN correo TO correo_old;
```

-- 2. Crear la nueva columna con tipo correcto

```
ALTER TABLE personas ADD COLUMN correo text;
```

-- 3. Copiar y cifrar los valores antiguos

```
UPDATE personas SET correo = encode(pgsql_sym_encrypt(correo_old, 'clave_segura'), 'base64');
```

-- 4. (Opcional) Eliminar la columna antigua A

```
ALTER TABLE personas DROP COLUMN correo_old;
```

	id [PK] integer	nombres character varying (100)	apellidos character varying (100)	correo_old character varying (100)	telefono character varying (20)	direccion text	correo text
1	1	Xavier Alejandro	Guanoluisa Quevedo	alejo@gmail.com	0993630096	Reino de Quito	ww0EBwMCBAxTKayAVHd30t
2	2	Emily Alejandra	Galeas Tingo	emy@gmail.com	0987458521	Carcelen	ww0EBwMCMDCnhqawF25r0j
3	3	Eduardo Steven	Chacha Guzman	edu@mail.com	0987458578	Conocoto	ww0EBwMCNP3Er4kNwjJg0j0
4	4	Rosa Maria	Ganchala Chulide	rosa@gmail.com	0963212541	Calderon	ww0EBwMCqKMAshonSAZ50j
5	5	Angelica Fernanda	Garcia Torres	ange@gmail.com	0912541236	La Carolina	ww0EBwMCav8FtBH0jJR+0j8E
6	6	Jose Manuel	Rojas Cruz	jose@gmail.com	0987485421	Chillogallo	ww0EBwMCwAYt4Bf471d0j0j8E
7	11	Pepe Manuel	Espin Vasquez	pepe@gmail.com	0987412102	Mena 2	ww0EBwMCTKrU80NguTz0j0j8E

```
51 ALTER TABLE usuarios RENAME COLUMN contraseña TO contrasenia_old;
52 ALTER TABLE usuarios ADD COLUMN contrasenia text;
53 UPDATE usuarios SET contrasenia = encode(pgsql_sym_encrypt(contrasenia_old, 'contraseña_seguro'), 'base64');
54 ALTER TABLE usuarios DROP COLUMN contrasenia_old;
55
```

Data Output Messages Notifications

	id [PK] integer	persona_id integer	contrasenia_old character varying (100)	tipo_usuario tipo_usuario_enum	contrasenia text
1	1	1	alejo123	administrador	ww0EBwMCKQ9S5zest5hx0jkBEjglvbcEfMZcf+wUrk2PGnltqS8DIMYV6WD53laRfASISNF1
2	2	2	emi123	administrador	ww0EBwMCSDdC7akfNHrt0jcBfQmf0vBMWaKfxMmaABXpRvYV/uQHTpp5TFXL1hwjDy
3	4	4	rosa123	emprendedor	ww0EBwMCkrhq2y4KkFJl0jgBKyo5/CH4YBLjQlx8xltveRfoATMUrzucJXXS/X8ZLwCxGqf
4	5	5	ange123	emprendedor	ww0EBwMCs/Zj4JE108p/OjgBsrJGcktf0y78IPjGEV+AZ3YpF0JLaZ2GbBeWqb+vioCSqtorr
5	3	3	edu123	mentor	ww0EBwMCSumRSLgr4/dy0jcBhNkmnVmz9dWRwr6r9009E0DWFeVdP/yvLqVLnyktj1vqV

Si queremos crear una nueva persona con correo encriptado:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- Inserta |
INSERT INTO personas(nombres,apellidos,correo_old,telefono,direccion,correo)VALUES
(
    'Laura Sofia',
    'Gomez Franco',
    'laura@gmail.com',
    '0987417802',
    'Mena 2',
    encode(pgpg_sym_encrypt('laura@gmail.com', 'correo_seguro'), 'base64')
);

-- Verifica que el correo fue insertado bien
SELECT nombres, correo FROM personas WHERE nombres = 'Laura Sofia';
```

Y para consultar el valor descifrado:

```
45 -- Intenta descifrar
46 SELECT nombres, apellidos,
47     pgpg_sym_decrypt(decode(correo, 'base64'), 'correo_seguro') AS correo_descifrado
48 FROM personas
49 WHERE nombres = 'Laura Sofia';
50
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No:

	nombres character varying (100)	apellidos character varying (100)	correo_descifrado text
1	Laura Sofia	Gomez Franco	laura@gmail.com

Todos los correos desencriptados:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
46 ▾ SELECT nombres, apellidos,  
47     pgp_sym_decrypt(decode(correo, 'base64'), 'correo_seguro') AS correo_descifrado  
48 FROM personas;  
49 |
```

Data Output Messages Notifications

	nombres character varying (100)	apellidos character varying (100)	correo_descifrado text
1	Xavier Alejandro	Guanoluisa Quevedo	alejo@gmail.com
2	Emily Alejandra	Galeas Tingo	emy@gmail.com
3	Eduardo Steven	Chacha Guzman	edu@mail.com
4	Rosa Maria	Ganchala Chulde	rosa@gmail.com
5	Angelica Fernanda	Garcia Torres	ange@gmail.com
6	Jose Manuel	Rojas Cruz	jose@gmail.com
7	Pepe Manuel	Espin Vasquez	pepe@gmail.com
8	Laura Sofia	Gomez Franco	laura@gmail.com

- Simulación de anonimización y enmascaramiento.

Anonimización: reemplazar datos por ficticios o genéricos.

```
59 ▾ UPDATE personas  
60   SET nombres = 'Usuario', apellidos = 'Anónimo', correo = NULL  
61   WHERE id = 2;  
62   select * from personas where id = 2;
```

Data Output Messages Notifications

	id [PK] integer	nombres character varying (100)	apellidos character varying (100)	correo_old character varying (100)	telefono character varying (20)	direccion text	correo text
1	2	Usuario	Anónimo	emy@gmail.com	0987458521	Carcelen	ww0EBwMCAMZUID1yTYF/0j4B

Enmascaramiento: mostrar parcialmente una parte(por ejemplo correo).



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
-- Enmascaramiento simple
CREATE TABLE personas_cifrado (
    id SERIAL PRIMARY KEY,
    nombres TEXT,
    apellidos TEXT,
    correo BYTEA -- importante: tipo bytea
);
INSERT INTO personas_cifrado(nombres, apellidos, correo)
VALUES (
    'Pepe',
    'Espin',
    pgp_sym_encrypt('pepe@gmail.com', 'clave_segura')
);

83 ▾ SELECT nombres,
84         apellidos,
85         -- Primero se descifra el correo, luego se enmascara
86         regexp_replace(pgp_sym_decrypt(correo, 'clave_segura')::TEXT,
87                         '^(.).*(@.*',
88                         '\1***\2') AS correo_enmascarado
89 FROM personas_cifrado;
90
```

Data Output Messages Notifications

	nombres	apellidos	correo_enmascarado
1	Pepe	Espin	p***@gmail.com
2	Pepe	Espin	p***@gmail.com
3	Pepe	Espin	p***@gmail.com

- Implementación de integridad lógica: reglas de negocio con procedimientos y funciones

1.- Procedimiento para insertar una persona con encriptación de correo.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
create or replace procedure insertar_persona_encriptada(
    in p_nombres varchar(100),
    in p_apellidos varchar(100),
    in p_correoOld varchar(100),
    in p_telefono varchar(20),
    in p_direccion text,
    in p_correo text
)
language plpgsql
as $$
begin
    insert into personas(nombres,apellidos,correo_old,telefono,direccion,correo)values
    (p_nombres,p_apellidos,p_correoOld,p_telefono,p_direccion,pgp_sym_encrypt(p_correo, 'correo_seguro'));
end;
```

124 ✓ call insertar_persona_encriptada(
125 'Isaac Steven','Pozo Achig','isac@gmail.com','0987410210','Reino de Quito','isac@gmail.com'
126);
127

Data Output Messages Notifications

Showing rows: 1 to 9 | | Page No: 1 of 1 |

	id [PK] integer	nombres character varying (100)	apellidos character varying (100)	correo_old character varying (100)	telefono character varying (20)	direccion text	correo text
7	1	Usuario	Anónimo	alejo@gmail.com	0993630096	Reino de Quito	ww0EBwMCPaUbtgMCbjBs
8	2	Usuario	Anónimo	emy@gmail.com	0987458521	Carcelen	ww0EBwMCAMZUID1yTYF/
9	15	Isaac Steven	Pozo Achig	isac@gmail.com	0987410210	Reino de Quito	\xc30d040703029c4f0014d

2.- Función para mostrar datos con correo desencriptado y enmascarado.



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
CREATE OR REPLACE FUNCTION obtener_personas_desencriptadas()
RETURNS TABLE (
    nombres TEXT,
    apellidos TEXT,
    correo_original TEXT,
    correo_enmascarado TEXT
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        p.nombres,
        p.apellidos,
        pgp_sym_decrypt(p.correo, 'clave_segura')::text AS correo_original,
        -- Enmascarar el correo, mostrando solo la parte antes de '@' como '***'
        regexp_replace(pgp_sym_decrypt(p.correo, 'clave_segura')::text, '^[@]+', '***') || '@' ||
        SPLIT_PART(pgp_sym_decrypt(p.correo, 'clave_segura')::text, '@', 2) AS correo_enmascarado
    FROM personas_cifrado p;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM obtener_personas_desencriptadas();
```

	nombres text	apellidos text	correo_original text	correo_enmascarado text
1	Pepe	Espin	pepe@gmail.com	***@gmail.com@gmail.com

13. Simulación de Perfiles Profesionales

- **Administrador de BD:** mantenimiento de índices, respaldo, programación de tareas automatizadas, monitoreo del sistema.

Mantenimiento de índices:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
4  -- mantenimiento de indices
5  -- Mejorar rendimiento de búsqueda por correo
6  CREATE INDEX idx_personas_correo ON personas(correo_old);
7  EXPLAIN SELECT * FROM personas WHERE correo_old = 'alejo@gmail.com';
8
```

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	QUERY PLAN	
1	Seq Scan on personas (cost=0.00..1.11 rows=1 ...)	<input type="button"/>
2	Filter: ((correo_old)::text = 'alejo@gmail.com'::te...	

Respaldo manual:

```
C:\Users\Flia Guanoluisa>pg_dump -U postgres -d Sistema_proyectos_juveniles > respaldo_sistema.sql
Contraseña:
```

respaldo_sistema.sql 2/8/2025 22:34

Monitoreo simple:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
SELECT relname AS tabla, pg_size.pretty(pg_total_relation_size(relid)) AS tamano
FROM pg_catalog.pg_statio_user_tables
ORDER BY pg_total_relation_size(relid) DESC;
```

	tabla name	lock	tamano text	lock
1	personas		64 kB	
2	ideas_negocio		48 kB	
3	fases_proyecto		48 kB	
4	categorias		48 kB	
5	personas_cifrado		32 kB	
6	usuarios		32 kB	
7	observaciones		32 kB	
8	tipo_estados		24 kB	
9	mentorias		24 kB	
10	mentores		24 kB	
11	logs_sistema		16 kB	
12	avance_fases		8192 bytes	
13	resultados		8192 bytes	
14	reportes		8192 bytes	
15	estadisticas_idea		8192 bytes	

- **Arquitecto de BD:** diseño lógico y físico, definición de estándares, proyección de escalabilidad y modularidad, revisión de integridad. Verificación de integridad referencial



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
-- Listar claves foráneas
SELECT conname AS restriccion, relname AS tabla
FROM pg_constraint
JOIN pg_class ON conrelid = pg_class.oid
WHERE contype = 'f';
```

	restriccion name	tabla name
1	fk_usuario_persona	usuarios
2	fk_mentor_persona	mentores
3	fk_idea_usuario	ideas_negocio
4	fk_idea_categoria	ideas_negocio
5	fk_avance_idea	avance_fases
6	fk_avance_fase	avance_fases
7	fk_mentoria_idea	mentorias
8	fk_observacion_mentoria	observaciones
9	fk_log_usuario	logs_sistema
10	fk_usuario_id	resultados
11	fk_estadisticas_id	resultados
12	fk_resultado_id	reportes
13	fk_estado	mentorias
14	fk_mentoria_id	estadisticas_idea
15	fk_mentoria_mentor	mentorias
16	fk_avance_fase	estadisticas_idea
17	fk_estado	ideas_negocio

Revisión de tipos y modularidad:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

	column_name name	data_type character varying
1	id	integer
2	usuario_id	integer
3	categoria_id	integer
4	titulo	character varying
5	descripcion	text
6	estado	integer

- Oficial de Seguridad:** creación de roles, privilegios, revisión de logs, pruebas de vulnerabilidad, gestión de datos sensibles.

Encriptación de datos sensibles:

```
34 -- Ejemplo de enmascarar correo
35 ALTER TABLE personas
36 ALTER COLUMN correo_old TYPE text;
37
38 UPDATE personas
39 SET correo_old = pgp_sym_encrypt(correo_old::text, 'nueva_clave');
40
41
```

La captura de pantalla muestra una interfaz de usuario para administrar bases de datos. En la parte superior, se visualiza el código SQL que encrypta el campo 'correo_old' de la tabla 'personas'. La ejecución del script se indica por los números de línea 34 a 41. Abajo de esto, se muestra la lista de resultados de la consulta, titulada 'Data Output'. Los resultados muestran seis filas de datos con columnas: 'id' (PK integer), 'nombres' (character varying (100)), 'apellidos' (character varying (100)) y 'correo_old' (text). Los valores de 'correo_old' son representaciones hexadecimales de cadenas de texto.

	id [PK] integer	nombres character varying (100)	apellidos character varying (100)	correo_old text
1	3	Eduardo Steven	Chacha Guzman	\xc30d040703023cc3bfd04e55f96e77d23d016791e89b6663908860e7bbe6688;
2	4	Rosa Maria	Ganchala Chulde	\xc30d040703021f137cec84fc20e64d23f0181e69a627fd026e4b227c829cd53;
3	5	Angelica Fernanda	Garcia Torres	\xc30d0407030239877bcc04f426dc6ad23f01da1059e72bd4b76beef06113ccb
4	6	Jose Manuel	Rojas Cruz	\xc30d0407030223bda783dd8231947ad23f015513a0ff344f10da6842bb2e390e
5	11	Pepe Manuel	Espin Vasquez	\xc30d04070302fc54c6d2d3d1914f73d23f012c6c03473482723943bdda969b28
6	14	Laura Sofia	Gomez Franco	\xc30d04070302c5ec7a9c768e4dd278d240010ba2cd70130a40f940c7c7c004f4

Desencriptamos:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
42 ✓ SELECT LEFT(pgp_sym_decrypt(correo_old::bytea, 'nueva_clave')::text, 255)
43 FROM personas;
44
```

Data Output Messages Notifications

	left text	lock
1	edu@mail.com	
2	rosa@gmail.com	
3	ange@gmail.co...	
4	jose@gmail.com	
5	pepe@gmail.co...	
6	laura@gmail.co...	
7	alejo@gmail.com	
8	emy@gmail.com	

Revisión de actividad:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
CREATE TABLE logs_sistema_2 (
    id SERIAL PRIMARY KEY,
    accion VARCHAR(50),
    usuario VARCHAR(50),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    datos JSONB
);

CREATE OR REPLACE FUNCTION log_acciones_personas()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO logs_sistema_2 (accion, usuario, datos)
        VALUES ('INSERT', current_user, row_to_json(NEW));
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO logs_sistema_2 (accion, usuario, datos)
        VALUES ('UPDATE', current_user, row_to_json(NEW));
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO logs_sistema_2 (accion, usuario, datos)
        VALUES ('DELETE', current_user, row_to_json(OLD));
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```
74 ✓ CREATE TRIGGER personas_audit
75 AFTER INSERT OR UPDATE OR DELETE ON personas
76 FOR EACH ROW EXECUTE FUNCTION log_acciones_personas();
77
78 ✓ insert into personas(nombres,apellidos,correo_old,telefono,direccion,correo) values
79 ('Samuel','Cahngo','sam@gmail.com','0996584716','Mena2','sam@gmail.com');
80
81 select * from logs_sistema_2;
```

Data Output Messages Notifications

	id [PK] integer	accion character varying (50)	usuario character varying (50)	fecha timestamp without time zone	datos jsonb
1	1	INSERT	postgres	2025-08-02 23:22:05.783556	{"id": 17, "correo": "sam@gmail.com", "nombres": "Sa"}

- **Desarrollador de Consultas:** construcción de vistas eficientes, reportes complejos, procedimientos y funciones reutilizables.

Vista ideas por estado:



BASES DE DATOS

(TDSD353)

```
85 ✓ CREATE OR REPLACE VIEW vista_ideas_estado AS
86   SELECT estado, COUNT(*) AS total
87   FROM ideas_negocio
88   GROUP BY estado;
89
90   select * from vista_ideas_estado;
```

Data Output Messages Notifications

	estado integer	total bigint
1	2	2

Función para obtener ideas por usuario:

```
93   -- función
94   drop function ideas_por_usuario;
95 ✓ CREATE OR REPLACE FUNCTION ideas_por_usuario(uid INT)
96   RETURNS TABLE(titulo_idea varchar(200), estado_idea int) AS $$
97   BEGIN
98       RETURN QUERY
99       SELECT titulo, estado FROM ideas_negocio WHERE usuario_id = uid;
100   END;
101   $$ LANGUAGE plpgsql;
102   select ideas_por_usuario(5);
103
104
```

Data Output Messages Notifications

	ideas_por_usuario record
1	AuntoBufanda,2

Procedimiento para aprobar idea:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TDSD353)

```
106 CREATE OR REPLACE PROCEDURE aprobar_idea(idea INT, mentor INT)
107 LANGUAGE plpgsql AS $$ 
108 BEGIN
109     UPDATE ideas_negocio SET estado = 2 WHERE id = idea;
110     INSERT INTO mentorias(idea_id, mentor_id, fecha, estado)
111         VALUES (idea, mentor, NOW(), 2);
112 END;
113 $$;
114
115 insert into ideas_negocio(usuario_id, categoria_id, titulo, descripcion, estado) values
116 (4,1,'SuperTablet','Tablet con ia',1);
117
118 call aprobar_idea(5,1);
119
```

Data Output Messages Notifications

	id [PK] integer	idea_id integer	mentor_id integer	fecha date	estado integer
1		2	3	1	2025-07-30
2		3	5	1	2025-08-02

- Analista de Datos:** generación de métricas, KPIs, tablas resumen, integración con Power BI o Excel (opcional).

Métricas: ideas por características.

```
122 SELECT c.nombre AS categoria, COUNT(*) AS total_ideas
123 FROM ideas_negocio i
124 JOIN categorias c ON c.id = i.categoria_id
125 GROUP BY c.nombre;
```

Data Output Messages Notifications

	categoria categoria_enum	total_ideas bigint
1	Tecnología	2
2	Moda	1

Importar como csv:



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
COPY (
  SELECT * FROM ideas_negocio
) TO 'C:\Users\Flia Guanoluisa\OneDrive\Documentos\Proyecto_Bases_de_datos\export_ideas_estado.csv' DELIMITER ',' CSV HEADER;
```

export_ideas_estado.csv

Origen de archivo		Delimitador		Detección del tipo de datos	
1252: Europeo occidental (Windows)		Coma		Basado en las primeras 200 filas	
	id	usuario_id	categoria_id	título	descripcion
	3	4	1	SuperTV	Televisión con IA
	4	5	5	AuntoBufanda	Bufanda que se cierra sola
	5	4	1	SuperTablet	Tablet con IA

- **Usuario Final:** interacción con vistas controladas, ejecución de consultas básicas.

Vista mis ideas de negocio (por usuario):

```
CREATE OR REPLACE VIEW vista_mis_ideas AS
SELECT
  i.id AS id,
  i.título,
  i.descripcion,
  i.estado,
  c.nombre AS categoria,
  u.id AS usuario_id,
  p.nombres || ' ' || p.apellidos AS emprendedor
FROM ideas_negocio i
JOIN categorias c ON c.id = i.categoría_id
JOIN usuarios u ON u.id = i.usuario_id
JOIN personas p ON p.id = u.persona_id;
select * from vista_mis_ideas;
```

	id integer	título character varying (200)	descripcion text	estado integer	categoria categoria_enum	usuario_id integer	emprendedor text
1	5	SuperTablet	Tablet con IA	2	Tecnología	4	Rosa María Ganchala Chulde
2	3	SuperTV	Televisión con IA	2	Tecnología	4	Rosa María Ganchala Chulde
3	4	AuntoBufanda	Bufanda que se cierra sola	2	Moda	5	Angelica Fernanda García Torres

Vista mis mentorías (para mentor):



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
152 ✓ CREATE OR REPLACE VIEW vista_mis_mentorias AS
153   SELECT
154     m.id,
155     m.fecha,
156     m.estado,
157     i.titulo AS idea,
158     p.nombres || ' ' || p.apellidos AS mentor
159   FROM mentorias m
160   JOIN mentores mt ON mt.id = m.mentor_id
161   JOIN personas p ON p.id = mt.persona_id
162   JOIN ideas_negocio i ON i.id = m.idea_id;
163   select * from vista_mis_mentorias;
164
165
```

Data Output Messages Notifications

	id integer	fecha date	estado integer	idea character varying (200)	mentor text
1	2	2025-07-30	2	SuperTV	Eduardo Steven Chacha Guzman
2	3	2025-08-02	2	SuperTablet	Eduardo Steven Chacha Guzman

Vista mis observaciones (por mentoría):

```
168 ✓ CREATE OR REPLACE VIEW vista_mis_observaciones AS
169   SELECT
170     o.id,
171     o.comentario,
172     o.mentoría_id,
173     m.fecha,
174     i.titulo AS idea
175   FROM observaciones o
176   JOIN mentorias m ON m.id = o.mentoría_id
177   JOIN ideas_negocio i ON i.id = m.idea_id;
178   select * from vista_mis_observaciones;
179
```

Data Output Messages Notifications

	id integer	comentario text	mentoría_id integer	fecha date	idea character varying (200)
1	1	Le falta documentacion	2	2025-07-30	SuperTV

Vista resultados por usuario:



BASES DE DATOS

(TDSD353)

```
190 ✓ CREATE OR REPLACE VIEW vista_mis_resultados AS
191   SELECT
192     r.id,
193     p.nombres || ' ' || p.apellidos AS emprendedor,
194     e.avance_fase_id AS fase_asignada,
195     i.titulo AS idea
196   FROM resultados r
197   JOIN usuarios u ON u.id = r.usuario_id
198   JOIN personas p ON p.id = u.persona_id
199   JOIN estadisticas_idea e ON e.id = r.estadisticas_id
200   JOIN mentorias m ON m.id = e.mentoría_id
201   JOIN ideas_negocio i ON i.id = m.idea_id;
202   select * from vista_mis_resultados;
```

Data Output Messages Notifications

≡+ ↻ 🔍 ↴ 🗑️ 🔍 ↵ SQL

	id integer	emprendedor text	fase_asignada integer	idea character varying (200)
1	2	Eduardo Steven Chacha Guzman	2	SuperTV

Vista: Ideas con estado y mentor asignado (para ver estado de aprobación):



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS

(TDSD353)

```
205 ✓ CREATE OR REPLACE VIEW vista_estado_ideas AS
206   SELECT
207     i.id,
208     i.titulo,
209     i.estado,
210     COALESCE(p.nombres || ' ' || p.apellidos, 'Sin mentor') AS mentor_asignado
211   FROM ideas_negocio i
212   LEFT JOIN mentorias m ON m.idea_id = i.id
213   LEFT JOIN mentores mt ON mt.persona_id = m.mentor_id
214   LEFT JOIN personas p ON p.id = mt.persona_id;
215   select * from vista_estado_ideas;
```

Data Output Messages Notifications

Showing rows: 1

	id integer	titulo character varying (200)	estado integer	mentor_asignado text
1	3	SuperTV	2	Sin mentor
2	5	SuperTablet	2	Sin mentor
3	4	AuntoBufanda	2	Sin mentor