Snippets Everest: Guía de Referencia

Rick

2025-11-29

Table of contents

1 Presentación

Este es un compendio de todo el codigo que necesitaré para buscar a Sujhey y perdirle una oportunidad. Todo el codigo debe ser referido a Api(Controller).

Part I

I. Git

2 Comandos básicos

2.1 Pasos

1. Ubicarse en la carpeta y crear reposiorio local
2. Agregar al pool local las nuevas modificaciones

2.2 Código

1. Ubicarse en la carpeta y crear reposiorio local

```
# Iniciar repositorio
git init

# Agregar los archivos al pool
git add .

# Verificar si fueron agregados
git status

# Hacer primer commit
git commit -m "Init"

# Verificar el commit
git log --oneline

# Crear repositorio en github, si se desea publicar debe ser publico.
git remote add repos https://github.com/aerostain/SnippetsBook_011225.git

# Push los archivos del pool local al repositorio
git push repos master

# Verificar
git log
```

2. Agregar al pool local las nuevas modificaciones

```
# Agregar nuevos cambios al pool
git add .

# Hacer commit
git commit -m "Add Git"

# Agregar a repositorio remoto
git push repos master

# Verificar
git log
```

Part II

II. Quarto

3 Book

3.1 Plan

1. Crear _quarto.yml.
2. Crear index.qmd -> Aquí ya funciona!.
3. Agregar estilos css con style.css.
4. Previsualizar / renderizar.

3.2 Código

1. Crear _quarto.yml

```
project:
  type: book
  output-dir: _book

book:
  title: "Snippets Everest: Guía de Referencia"
  author: "Rick"
  date: "2025-11-29"

  chapters:
    - index.qmd
    - part: "I. PostgreSQL"
      chapters:
        - PostgreSql/Conexion.qmd
        - PostgreSql/ConfigApi.qmd
        - PostgreSql/ComandosBasicos.qmd
        - PostgreSql/Utilidades.qmd
        - PostgreSql/Ejemplo1.qmd

  downloads: [pdf]

format:
```

```yaml
  html:
    theme:
      light: lux
      dark: slate

    css:
      - https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1,300..800&di
      - https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css
      - style.css

    # Opciones de libro
    toc: true # Muestra la Tabla de Contenidos en la navegación
    #toc-depth: 3 # Muestra encabezados hasta el nivel H3 en la tabla de contenidos
    code-fold: true # Permite plegar bloques de código para un aspecto más limpio

  pdf:
    documentclass: scrreprt
```

2. Crear index.qmd Siempre debe existir un archivo index.qmd

```yaml
# Introducción
Escribir un resumen.
```

3. Crear style.css

```css
/* Aplica la fuente "Open Sans" a todo el cuerpo del texto y encabezados */
body {
  //font-family: 'Calibri';
  font-size:.8rem;
}

/* Opcional: para asegurarte de que los encabezados también la usen */
h1, h2, h3, h4, h5{
  font-family: 'Calibri', sans-serif;
}

h1{
  font-size:1.3rem;
}

h2{
  font-size:1.2rem;
```

```
}

h3{
  font-size:1rem;
}
```

4. Visualizar Si se usa desde block de notas la vista solo se actualiza si se modifica index.qmd y _quarto.yml

```
quarto preview
```

Part III

III. PostgreSQL

4 Conexion

4.1 Powershell + PgAdmin + Aiven

Pasos:

1. Ubicar el psql.exe, al tener instalado PgAdmin: `Z:\Program Files\pgAdmin 4\runtime`.
2. En Powershell:

```
# Alias temporal
Set-Alias psql "Z:\Program Files\pgAdmin 4\runtime\psql.exe"

# Contraseña Aiven
$env:PGPASSWORD="AVNS_9oZSu3QxeQTqm33fAUN"

# Conexión (Al pareceer las comillas son opcionales)
psql -h postgreusa101-richims026-65e7.k.aivencloud.com -p 28194 -U avnadmin -d defaultdb
```

La mejor opción es usar una variable de entorno para la contraseña, como se hizo arriba, pero sino puedes usar:

```
# Ejecutalá y te pedirá ingresar la contraseña:
& "Z:\Program Files\pgAdmin 4\runtime\psql.exe" `
-h TU_HOST_AIVEN `
-p TU_PUERTO_AIVEN `
-U TU_USUARIO_AIVEN `
-d TU_NOMBRE_DB `
**-W**
```

5 Configuración Api

5.1 Referencias:

https://x.com/i/grok?conversation=1994596166160257348 https://gemini.google.com/app/a9e250c34ae5b

5.2 Recomendaciones:

- Nunca uses el usuario postgres (superusuario) para conectar tu API en producción. Lo correcto y más seguro es crear un rol específico para tu aplicación con el principio de mínimo privilegio.
- Primero crea la base de datos y el esquema con el superusuario (postgres), luego crea el rol de la aplicación con mínimos privilegios, y finalmente le das solo los permisos que necesita sobre lo que ya existe y lo que se creará en el futuro.

5.3 Pasos:

1. Usar el superusuario postgre para:
    - Crear base de datos.
    - Crear tablas y relaciones (ejecutar script).
    - Crear siempre un esquema.
    - Crea rol/usuario y darle permisos.
    - Conceder al rol/usuario permisos para modificaciones futuras.
2. Al usar PgAdmin
    - Seleccionar la base de datos y elegir ambos:
        - PSQL Tool
        - Query Tool

Comandos básicos para explorar:

```
# Para ver todos los roles/usuarios y permisos
\du+

# Eliminar roles
drop role mirol;

# Cambiar de rol
set role avnadmin;

# Saber mi rol actual
select current_user;
\conninfo

# Ver los permisos de un rol/usuario a una tabla especifica
\z postulantes."Test"
\dp postulantes."Test"

# Todas las bases de datos
\l

# Conectarse a una
\c

# Ver esquemas
\dn

# Ver tablas
\dt

# Estructura de la tabla
\d "Destinos"
```

5.4 Script completo de creación:

```
-- Conéctate como postgres o un superusuario, solo un superusuario puede crear bases de datos
CREATE DATABASE postulantes_db;

-- Conectarme a la base de datos
\c postulantes_db;
```

```sql
-- Creamos el esquema
CREATE SCHEMA IF NOT EXISTS postulantes AUTHORIZATION postgres;

-- Creamos el usuario que usará la API
CREATE ROLE miapi_postulantes WITH LOGIN PASSWORD 'Sup3rS3cr3tP4ssw0rd!2025!';

-- Todas las tablas, secuencias, funciones, etc. se crean con owner = postgres
-- Ejecutas todas las migraciones / scripts SQL (EF Core migrations, Flyway, Liquibase, etc.)

-- Le permitimos entrar a la base
GRANT CONNECT ON DATABASE postulantes_db TO miapi_postulantes;

-- Que pueda ver el esquema
GRANT USAGE ON SCHEMA public TO miapi_postulantes;

-- Permiso para crear objetos temporales (EF los usa)
GRANT TEMPORARY ON DATABASE postulantes_db TO miapi_postulantes;

-- Permisos sobre objetos existentes
-- Lectura/escritura en tablas existentes
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO miapi_postulantes;

-- Para que funcione el auto-increment
GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO miapi_postulantes;

-- Si tienes stored procedures o funciones
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO miapi_postulantes;

-- ¡¡IMPORTANTÍSIMO!! Permisos por defecto para objetos futuros (EF Core los crea en runtime)
-- Para tablas que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO miapi_postulantes;

-- Para secuencias que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  GRANT USAGE, SELECT, UPDATE ON SEQUENCES TO miapi_postulantes;

-- Para funciones que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  GRANT EXECUTE ON FUNCTIONS TO miapi_postulantes;

-- Quitamos permiso de crear objetos nuevos (solo el equipo de despliegue los crea)
```

```
REVOKE CREATE ON SCHEMA postulantes FROM miapi_postulantes;

-- El owner de la base sigue siendo un superusuario o un rol de despliegue, nunca el de la ap
ALTER DATABASE postulantes_db OWNER TO postgres;
```

5.5 Api

Configuración del appsettings.json para usar en la Api:

```
"ConnectionStrings": {
  "DefaultConnection": "Host=localhost;Database=postulantes_db;Username=miapi_postulantes;Pas
}
```

6 Comandos Básicos

6.1 Referencias

Procedimientos Almacenados y comandos básicos link

6.2 Exploración

```
select * from pg_database;
select * from pg_tables;
select * from pg_tables where schemaname like 'pg_catalog';
select * from pg_roles;
```

6.2.1 Exploracion en consola

```
\l # Lista todas las bases de datos.
\c <nombre_base_de_datos> # Conecta a una base de datos diferente.

\dt # Lista las tablas de la base de datos actual, solo esquema public.
\dt *.* # Lista las tablas de todos los esquemas.
\dt+ postulantes.* # Lista las tablas de ese esquema.
\dt+ postulantes.mitabla # Lista una tabla especifica.

\dv # Lista todas las vistas.
\df # Lista todas las funciones.
\dn # Lista los esquemas.
\dn+ postulantes # Permisos sobre un esquema.

\d # Estructura de una Tabla. Describe una tabla específica (o todas si no se especifica nomb
\d postulantes.* # Estructura de las tablas de ese esquema.
\d postulantes.mitabla # Estructura de las tablas especifica.
```

```
# Lista los usuarios/roles y permisos.
\du
\du+

# Ver los permisos de un rol/usuario a una tabla especifica.
\dp postulantes.*
\dp postulantes."Test"
\z postulantes.*
\z postulantes."Test"

# Saber mi rol actual
select current_user;
\conninfo # Solo da información de como me conecte, no del usuario actual.

# Cambiar de rol
set role avnadmin;

# Eliminar roles
drop role mirol;

# Guarda la salida de la consulta en el archivo especificado.
# La salida por defecto esta en -> Z:\Program Files\pgAdmin 4\runtime
# Puedes escribir la ruta así:  \o z:/r1ck7/Documents/tmp13.txt
# Escribir \o al inicio y final para dejar de escribir en el archivo.
\o resultados.txt
SELECT id, nombre FROM clientes WHERE ciudad = 'Lima';
\o

# Ejecuta comandos desde un archivo.
\i /rutacompleta/mis_comandos.sql

# Muestra el tiempo de ejecución de cada sentencia sql no de los comandos.
# \timing para iniciar y otra vez para terminar.
\timing

\? # Muestra una lista de todos los comandos de meta.
\h # Muestra la ayuda para comandos SQL. Por ejemplo, \h SELECT mostrará la ayuda para el co

\s # Muestra el historial de comandos.
\q # Sale de la consola de psql.
```

6.3 Seguridad: Permisos y Privilegios

6.3.1 Base de datos

Al usar łobtenemos `avnadmin=CTc/avnadmin`. |Código|Permiso Completo|Significado| |—|—|—| |C|CREATE|Permite al rol crear nuevos esquemas dentro de esta base de datos.| |T|TEMPORARY|Permite al rol crear tablas temporales en esta base de datos.| |c|CONNECT|Permite al rol conectarse a esta base de datos.|

6.3.2 Roles

Al usar `\z postulantes.tabletest` obtenemos: `adminapp=arwd/avnadmin`, en el formato general es: rol=privilegios/otrorol (donde otrorol es el rol que concedió esos privilegios).

Carácter	Privilegio	Descripción
a	INSERT	Permite añadir nuevas filas.
r	SELECT	Permite leer datos de las filas.
w	UPDATE	Permite modificar datos existentes.
d	DELETE	Permite eliminar filas.
D	TRUNCATE	Permite vaciar la tabla rápidamente.
x	REFERENCES	Permite crear una clave foránea que apunte a esta tabla.
t	TRIGGER	Permite crear triggers sobre la tabla.

6.3.3 Schemas

Al usar `\dn+ postulantes` obtenemos: `avnadmin=UC/avnadmin+` con el formato: role_app=UC/owner.

- U: Privilegio USAGE (Permite acceder a los objetos dentro del schema).
- C: Privilegio CREATE (Permite crear nuevos objetos dentro del schema).

Carácter	Objeto	Privilegio	Descripción
U	Schema USAGE	Permite acceder a los objetos dentro del schema.	

Carácter	Objeto	Privilegio	Descripción
C	Schema CREATE	Permite crear nuevos objetos.	
r	Tabla/Vista SELECT	Lectura de datos.	
w	Tabla/Vista UPDATE	Modificación de datos.	
a	Tabla/Vista INSERT	Adición de datos.	
d	Tabla/Vista DELETE	Eliminación de datos.	

Estos son comandos genericos. Para trabajar con Api revisar “Configuración Inicial”:

```
-- Crear rol/usuario
CREATE ROLE nombre_rol WITH LOGIN PASSWORD 'contraseña';

-- Conceder Permisos (Privilegios)
GRANT SELECT, INSERT ON nombre_tabla TO nombre_rol;
GRANT ALL PRIVILEGES ON DATABASE nombre_db TO nombre_rol;

-- Revocar Permisos: Elimina derechos otorgados.
REVOKE DELETE ON nombre_tabla FROM nombre_rol;

-- Convertir en Superusuario (con cuidado):
ALTER ROLE nombre_rol SUPERUSER;
```

6.4 Creación Base de Datos

```
-- Crear
CREATE DATABASE nombre_db;

-- Cambiar Nombre de la Base de Datos
ALTER DATABASE nombre_db RENAME TO nuevo_nombre;

-- Eliminar
DROP DATABASE nombre_db;
```

6.5 Creación Tablas

```sql
-- Crear Tabla
CREATE TABLE productos (id SERIAL PRIMARY KEY, nombre VARCHAR(100), precio NUMERIC(10, 2));

-- Actualizar Tabla
ALTER TABLE productos ADD COLUMN stock INTEGER DEFAULT 0;

-- Eliminar Tabla
DROP TABLE productos;
```

6.6 Creación Registros

```sql
-- Insertar
INSERT INTO tabla (col1, col2) VALUES ('valor1', 'valor2');

-- Leer
SELECT * FROM tabla WHERE condicion;

-- Actualizar
UPDATE tabla SET columna = 'nuevo_valor' WHERE id = 1;

-- Borrar
DELETE FROM tabla WHERE id = 5;
```

6.7 Referencias

Cualquier cambio en el padre desencadena una acción predefinida en el hijo (ON DELETE/ON UPDATE con CASCADE, SET NULL, etc.).

```sql
CREATE TABLE pedidos (
    pedido_id INT PRIMARY KEY,
    cliente_id INT,
    -- Define la clave foránea
    FOREIGN KEY (cliente_id) REFERENCES clientes (id) on delete cascade on update cascade
```

6.8 Filtro y Agrupación

```sql
-- WHERE
SELECT * FROM empleados WHERE salario > 50000;

-- GROUP BY
SELECT departamento, COUNT(*) FROM empleados GROUP BY departamento;

-- HAVING
SELECT departamento, AVG(salario) FROM empleados GROUP BY departamento HAVING AVG(salario) <

-- ORDER BY
SELECT nombre FROM empleados ORDER BY nombre DESC;

-- LIMIT
SELECT * FROM productos LIMIT 10;
```

6.9 Join

```sql
-- INNER JOIN
SELECT * FROM A INNER JOIN B ON A.fk = B.pk;

-- LEFT JOIN
SELECT * FROM clientes LEFT JOIN pedidos ON clientes.id = pedidos.cliente_id;
```

6.10 Restricciones y Check

```sql
CREATE TABLE notas (
    calificacion NUMERIC CHECK (calificacion >= 0 AND calificacion <= 10)
);
```

6.11 Procedimientos Almacenados/Funciones

Mientras que otros SGBD pueden usar los términos “procedimiento” y “función” indistintamente, PostgreSQL hace una distinción muy importante:

Característica	Procedimiento (CREATE PROCEDURE)	Función (CREATE FUNCTION)
Control Transaccional	Puede usar COMMIT y ROLLBACK.	No puede usar COMMIT o ROLLBACK.
Devuelve Valor	No devuelve un valor o conjunto de valores directamente (usa parámetros OUT).	Siempre devuelve un valor o un conjunto de valores (tabla).
Llamada	Se llama usando la sentencia CALL.	Se llama como una expresión en una consulta (SELECT).

Si necesita una unidad de trabajo que gestione su propia transacción (es decir, que decida cuándo grabar o deshacer los cambios), use un Procedimiento Almacenado. Si necesita una subrutina que calcule y devuelva un resultado, use una Función.

6.11.1 Procedimientos Almacenados

Cuando se requiere controlar transacciones. Si todo se cumple COMMIT. De lo contrario ROLLBACK.

```
-- Ver
\df

-- Crear
CREATE PROCEDURE insertarusuario_proc (_name varchar,_ischeck bool)
LANGUAGE plpgsql
AS $$
BEGIN
    insert into postulantes.tabletest values (default,_name,_ischeck,default);
    commit;
END;
$$;

-- Uso
call insertarusuario_proc('Raquel',true);

-- Borrar
drop procedure insertarusuario_proc;
```

6.11.2 Funciones

```
-- Ver
\df

-- Crear
CREATE FUNCTION calcular_iva (_precio NUMERIC, _tasa NUMERIC DEFAULT 0.19)
RETURNS NUMERIC -- Siempre tiene retorno.
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN _precio * _tasa;
END;
$$;

-- Uso
select name,calcular_iva(id) from postulantes.tabletest;

-- Borrar
drop function calcular_iva;
```

6.12 Triggers

7 Shortcuts

Codigo rápido.

- Para comillas en tablas y consultas:

```
-- Para nombres de Tablas usar comillas dobles:
select * from postulantes."Test";

-- Para valores de columna/celda usar comillas simples:
select * from pg_tables where schemaname like 'postulantes';

-- Ejemplo:
insert into postulantes."Test" values (default,'Luciana',true,default);
```

- Para crear la columna Fcreacion:

```
Fcreacion timestamptz default now()
```

- Roles/Usuarios

```
select current_user;
set role adminapp;
```

8 Ejemplo 1

Relación 1 a 1: Registro de Postulantes.

8.1 Pasos

1. Usando postgres/avnadmin crear la base de datos, el esquema y las tablas.
2. Crear un role/usuario y asignarle los permisos:
    - Acceso a la base de datos.
    - Usar el esquema, pero no agregar tablas al esquema.
    - Operaciones CRUD en las tablas (objetos existentes).
    - Permitir auto incremento.
    - Permitir objetos temporales.
    - Permitir stored procedures y/o funciones.
    - Permisos para EF Core (tablas futuras).
    - Permisos para auto incremento (tablas futuras).
    - Permisos para stored procedures y/o funciones (futuras).
    - Quitar el permiso de crear tablas en el esquema (por si acaso).
    - Asegurarse que el propietario siempre debe ser postgres/avnadmin (por si acaso).
3. Acceder con el usuario nuevo.

8.2 Código

Desde consola `psql.exe`.

1. Usando postgres/avnadmin crear la base de datos, el esquema y las tablas.

```
-- Inspeccionar y establecer usuario administrador
\l
\du+
select current_user;
set role avnadmin;
```

```
-- Crear Base de Datos y conectarse
CREATE DATABASE postulantes_db;
\c postulantes_db;

-- Crear esquema
CREATE SCHEMA IF NOT EXISTS postulantes AUTHORIZATION avnadmin;

-- Crear Tablas
create table postulantes."Postulante"(
id serial primary key,
nombre varchar(100),
adjuntos bool default true,
fcreacion timestamptz default now()
);

create table postulantes."Adjuntos"(
id serial primary key,
idpostulante int,
nombre_original varchar(250),
fcreacion timestamptz default now(),
foreign key (idpostulante) references postulantes."Postulante"(id) on delete cascade
);

-- Verificar
\dt+ postulantes.*
```

2. Crear un role/usuario y asignarle los permisos desde usuario avnadmin/postgres:

```
-- Creamos el usuario que usará la API
CREATE ROLE adminapp WITH LOGIN PASSWORD 'adminapp';

-- Verificar
\du+

-- Permiso para acceder a Base de datos
GRANT CONNECT ON DATABASE postulantes_db TO adminapp;

-- Verificar debe decir adminapp=Tc/avnadmin
\l

-- Permiso para acceder al esquema
GRANT USAGE ON SCHEMA postulantes TO adminapp;
```

```
-- Verificar debe decir adminapp=U/avnadmin
\dn+

-- Permiso para crear objetos temporales (EF los usa)
GRANT TEMPORARY ON DATABASE postulantes_db TO adminapp;

-- Permisos lectura/escritura en tablas existentes
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA postulantes TO adminapp;

-- Permiso para que funcione el auto-increment
GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA postulantes TO adminapp;

-- Permiso para stored procedures o funciones
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA postulantes TO adminapp;

-- ¡¡IMPORTANTÍSIMO!! Permisos por defecto para objetos futuros (EF Core los crea en runtime)

-- Permisos para tablas que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA postulantes
  GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO adminapp;

-- Permisos para secuencias que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA postulantes
  GRANT USAGE, SELECT, UPDATE ON SEQUENCES TO adminapp;

-- Permisos para funciones que se creen de ahora en adelante
ALTER DEFAULT PRIVILEGES IN SCHEMA postulantes
  GRANT EXECUTE ON FUNCTIONS TO adminapp;

-- Quitamos permiso de crear objetos nuevos (solo el equipo de despliegue los crea)
REVOKE CREATE ON SCHEMA postulantes FROM adminapp;

-- El owner de la base sigue siendo un superusuario o un rol de despliegue, nunca el de la ap
ALTER DATABASE postulantes_db OWNER TO avnadmin;
```

3. Acceder con el usuario nuevo.

```
-- Al ejecutar solicitará password
psql -h postgreusa101-richims026-65e7.k.aivencloud.com -p 28194 -U adminapp -d postulantes_db
```