**TECNOLOGICO NACIONAL DE MEXICO CAMPUS CULIACAN**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**



**MATERIA**

ADMINISTRACION DE BASE DE DATOS

**INTEGRANTES**

LOZANO CORVERA DANIEL ANTONIO

GARCIA AGUILAR JOSE ALFREDO

**MAESTRO**

DANIEL ESPARZA SOTO

**FECHA**

13-SEPTIEMBRE-2022

**PROYECTO 2 POSTGRESQL (POSTGRES)**

1. **HISTORIA**

PostgreSQL ha tenido una larga evolución, la cual se inicia en 1982 con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con el mismo, Michael decidió volver a la Universidad en 1985 para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la capacidad de definir tipos, pero también la capacidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En Postgres la base de datos comprendía las relaciones y podía obtener información de tablas relacionadas utilizando reglas. Postgres usó muchas ideas de Ingres pero no su código.

1. **VERSIONES**

Actualmente solo hay soporte para las versiones 10 a la 14

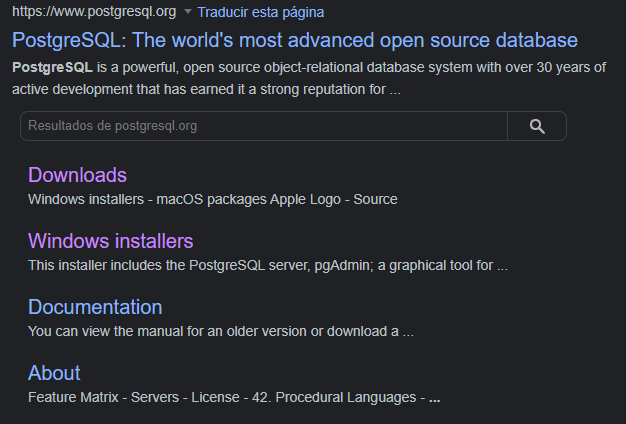
* PostgreSQL 14 (30 de septiembre de 2021)
* PostgreSQL 13 (24 de septiembre de 2020)
* PostgreSQL 12 (3 de octubre de 2019)
* PostgreSQL 11 (18 de octubre de 2018)
* PostgreSQL 10 (5 de octubre de 2017)
* PostgreSQL 9.6 (29 de septiembre de 2016)
* PostgreSQL 9.5 (7 de enero de 2016)
* PostgreSQL 9.4 (18 de diciembre de 2014)
* PostgreSQL 9.3 (9 de septiembre de 2013)
* PostgreSQL 9.2 (10 de septiembre de 2012)
* PostgreSQL 9.1 (12 de septiembre de 2011)
* PostgreSQL 9 (20 de septiembre de 2010)
* PostgreSQL 8.4 (1 de julio de 2009)
* PostgreSQL 8.3 (4 de febrero de 2008)
* PostgreSQL 8.2 (5 de diciembre de 2006)
* PostgreSQL 8.1 (8 de noviembre de 2005)
* PostgreSQL 8 (19 de enero de 2005)
* PostgreSQL 7.4 (17 de noviembre de 2003)
* PostgreSQL 7.3 (27 de noviembre de 2002)
* PostgreSQL 7.2 (4 de febrero de 2002)
* PostgreSQL 7.1 (13 de abril de 2001)
* PostgreSQL 7 (8 de mayo de 2000)
* PostgreSQL 6.5 (9 de junio de 1999)
* PostgreSQL 6.4 (30 de octubre de 1998)
* PostgreSQL 6.3 (1 de marzo de 1998)

1. **Características**

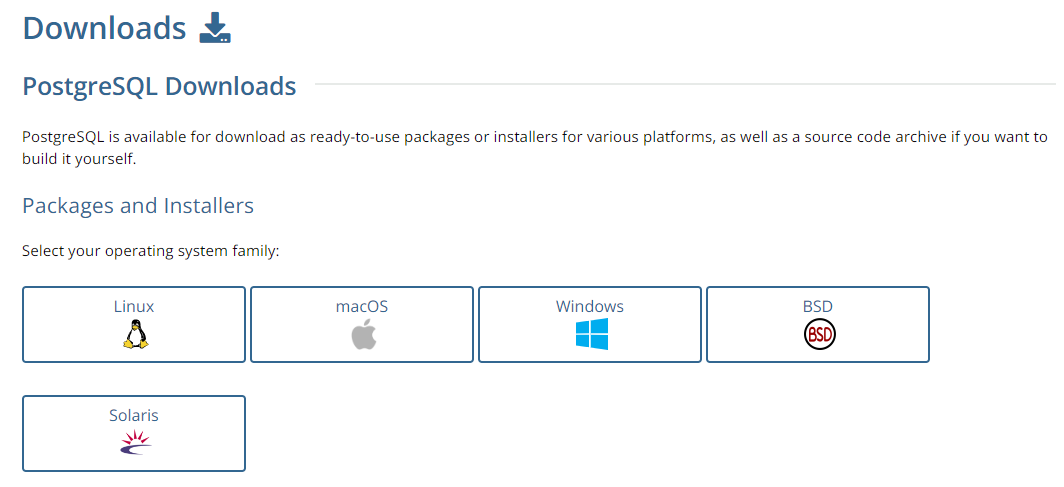
* Es el sistema gestor de bases de datos Open Source más avanzado
* Alta concurrencia. Es capaz de atender a muchos clientes al mismo tiempo y entregar la misma información de sus tablas, sin bloqueos.
* Soporte para múltiples tipos de datos de manera nativa. Ofrece los tipos de datos habituales en los sistemas gestores, pero además muchos otros que no están disponibles en otros competidores, como direcciones IP, direcciones MAC, Arrays, números decimales con precisión configurable, figuras geométricas, etc.
* Soporte a triggers. Permite definir eventos y generar acciones cuando estos se disparan.
* Trabajo con vistas. Esto quiere decir que pueden consultar los datos de manera diferente al modo en el que se almacenan.
* Objeto-relacional. Otra de sus principales características, que permite trabajar con sus datos como si fueran objetos y ofrece mecanismos de la orientación a objetos, como herencia de tablas.
* Soporte para bases de datos distribuidas. Donde el trabajo con transacciones asegura que estas tendrán éxito cuando han podido realizarse en todos los sistemas involucrados.
* Soporte para gran cantidad de lenguajes. PostgreSQL es capaz de trabajar con funciones internas, que se ejecutan en el servidor, escritas en diversos lenguajes como C, C++, Java, PHP, Python o Ruby. Además, ofrece interfaces para ODBC y JDBC, así como interfaces de programación para infinidad de lenguajes de programación.
* Todas las anteriores características y muchas otras convierten a PostgreSQL en una elección ideal para la mayoría de proyectos, en los que su funcionalidad, la seguridad o la integridad referencial nos resultan de gran importancia.

1. **MANUAL DEL PROCESO DE INSTALACIÓN Y CONEXIÓN AL SERVIDOR**

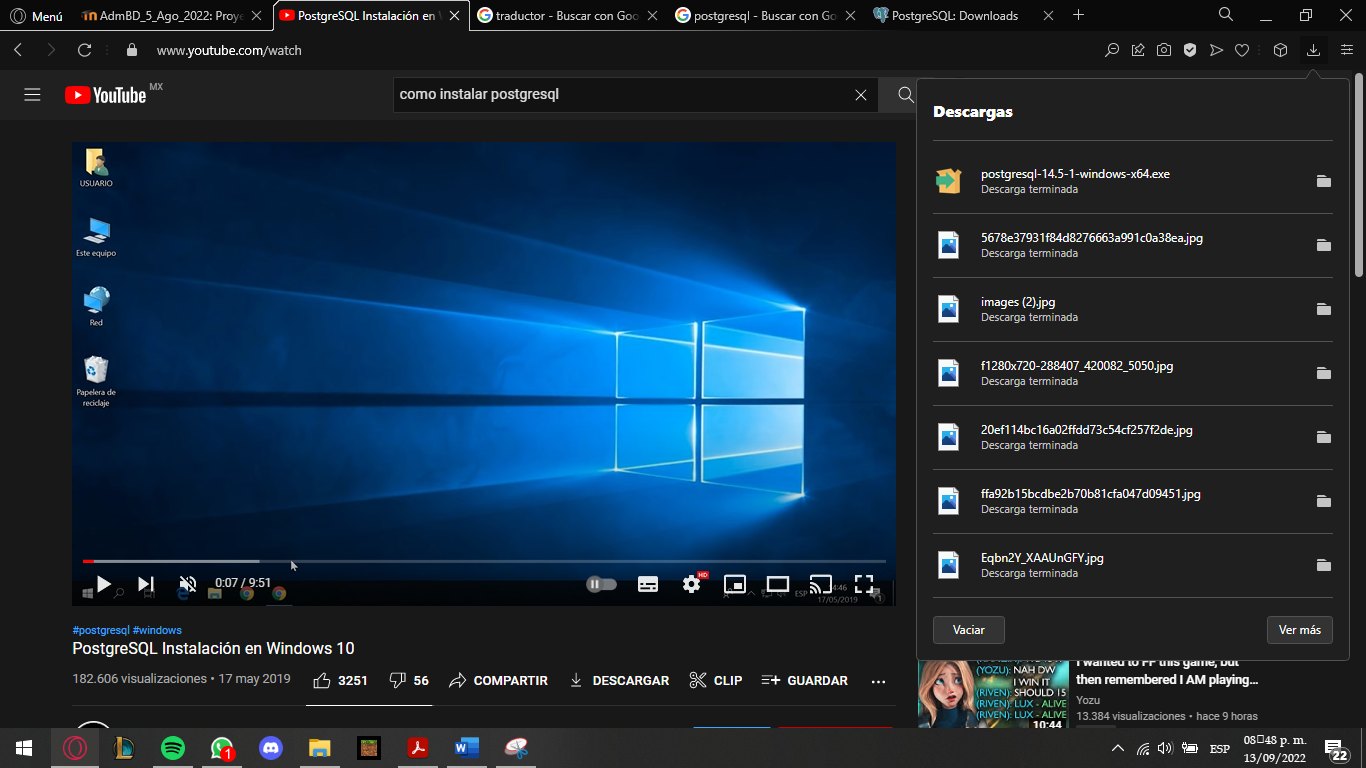
Vamos a la página oficial de descargas de PostgresSQL <https://www.postgresql.org/download/>



Descargamos la que ocupemos dependiendo de nuestro sistema operativo. En nuestro caso Windows



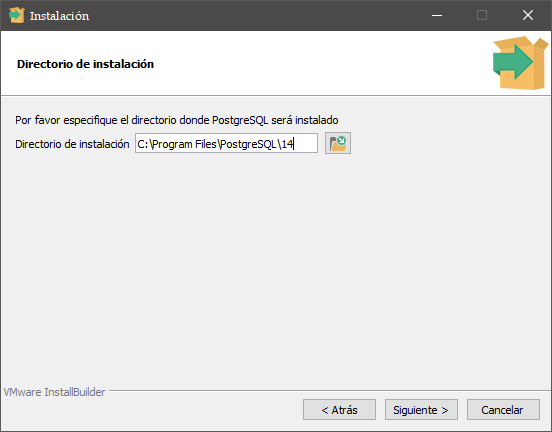
Una vez tengamos el archivo ejecutable, hacemos doble click en él



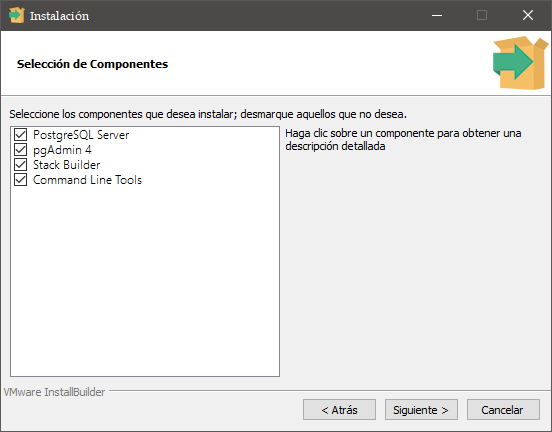
Se nos abre el instalador. Damos Next



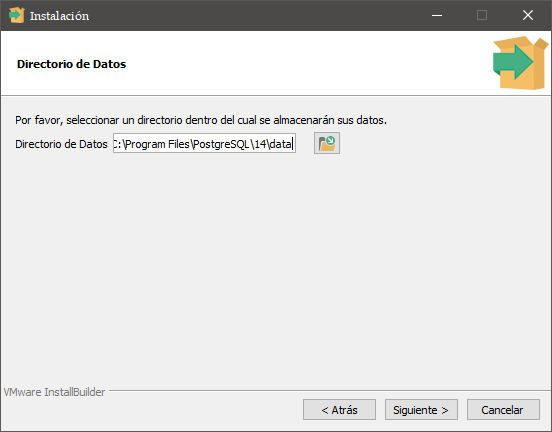
Nos muestra la ubicación de instalación. Damos Next



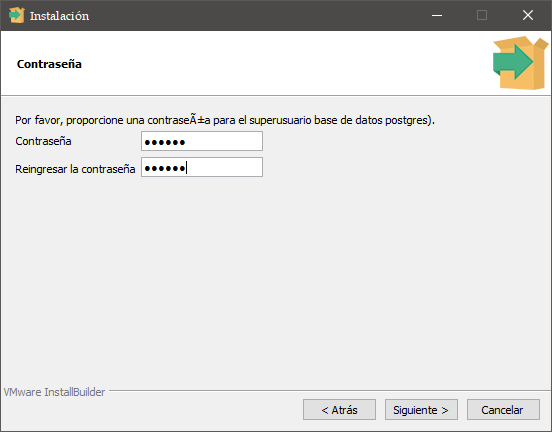
Dejamos marcados todos los componentes a instalar. Damos Next



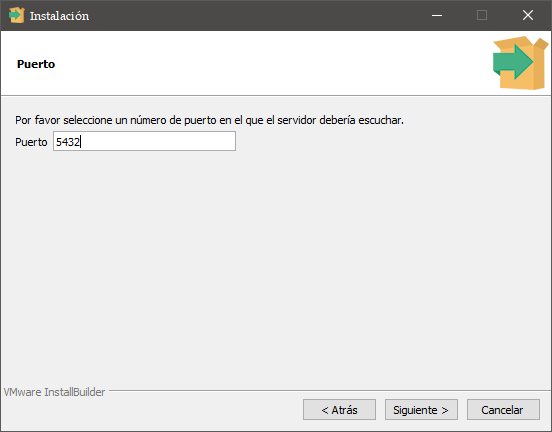
Dejamos el directorio de datos predeterminado. Next



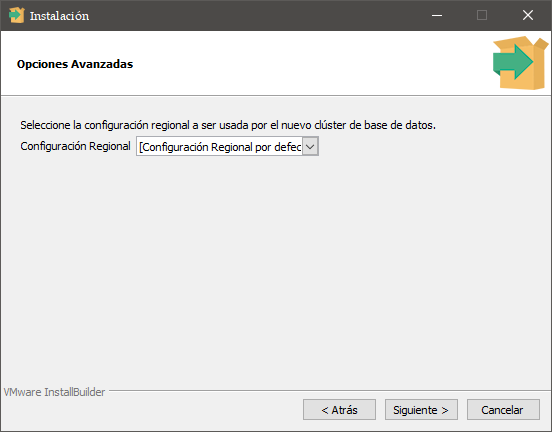
Le damos clave al superusuario, en nuestro caso 123456. Next



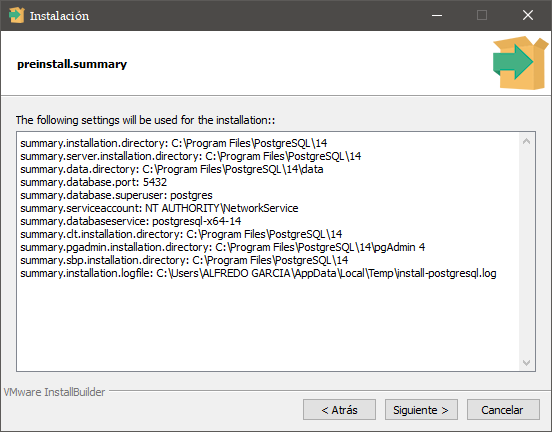
Dejamos el puerto predeterminado. Next



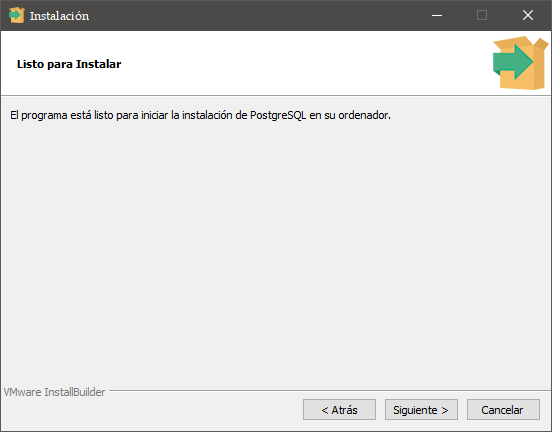
Configuración regional por defecto. Next



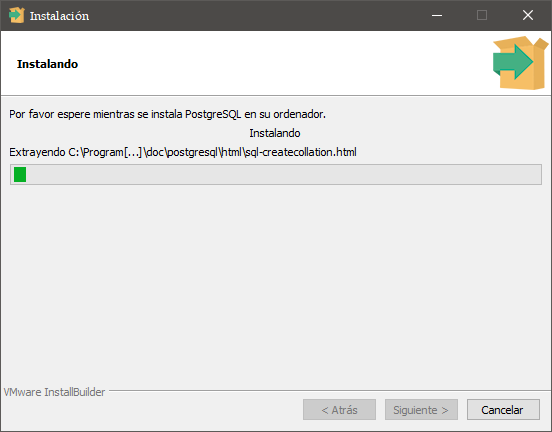
Configuraciones usadas para la instalación. Next



Next



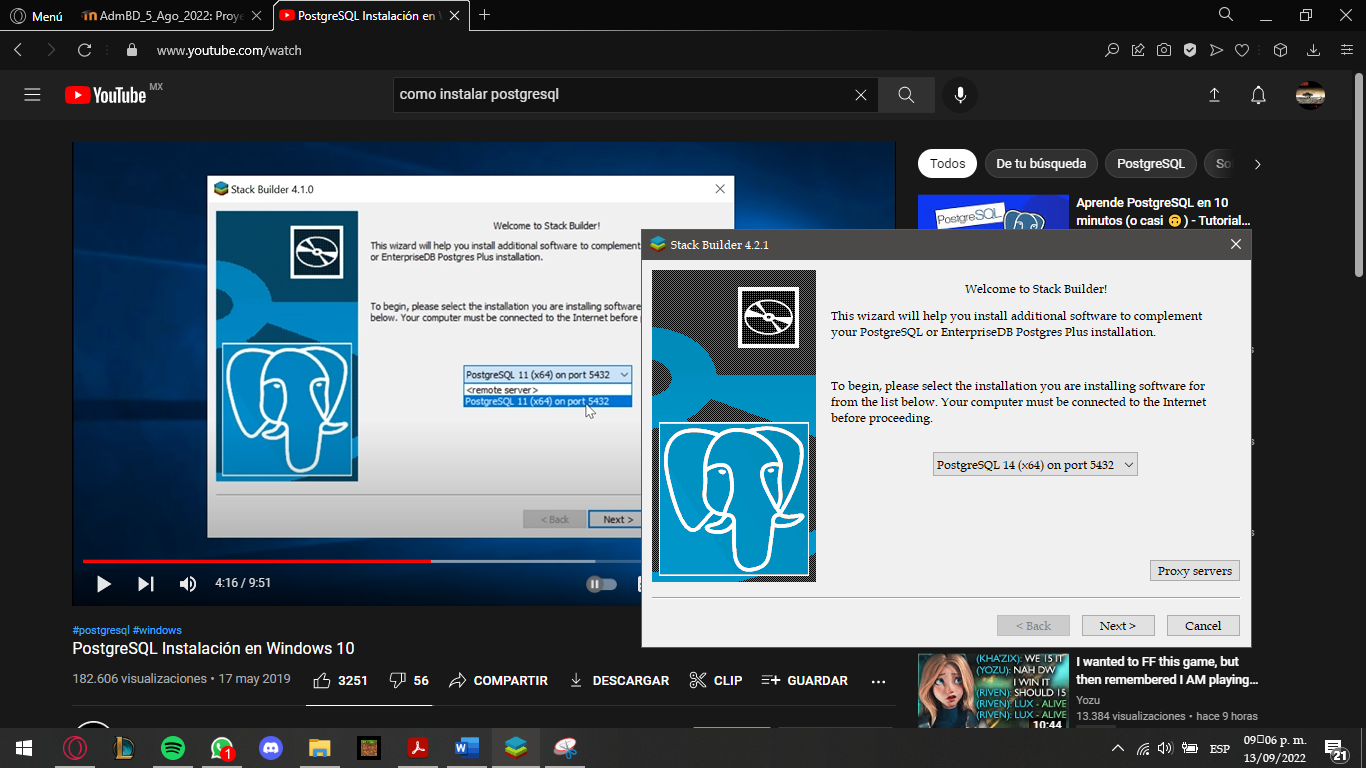
Esperamos a la instalación y damos Next



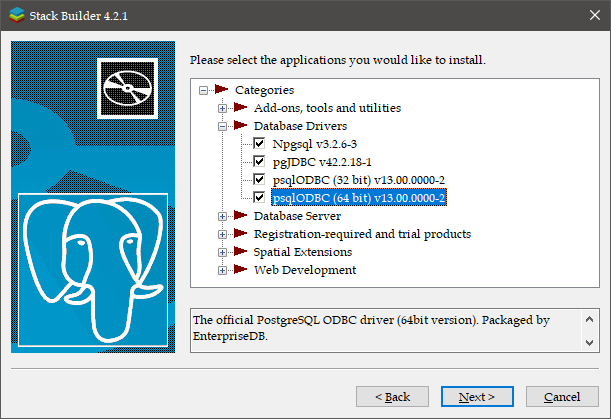
Next



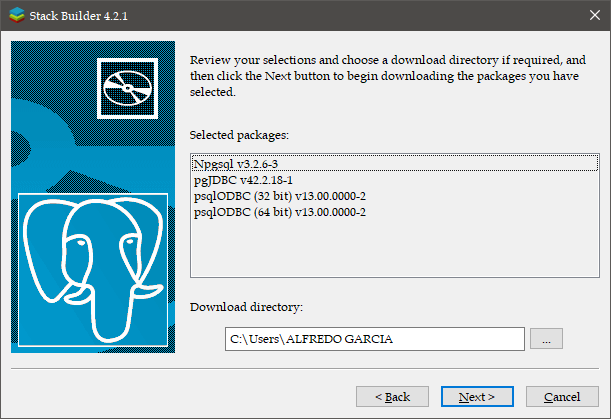
Le instalamos el complemento a la base de datos. Next

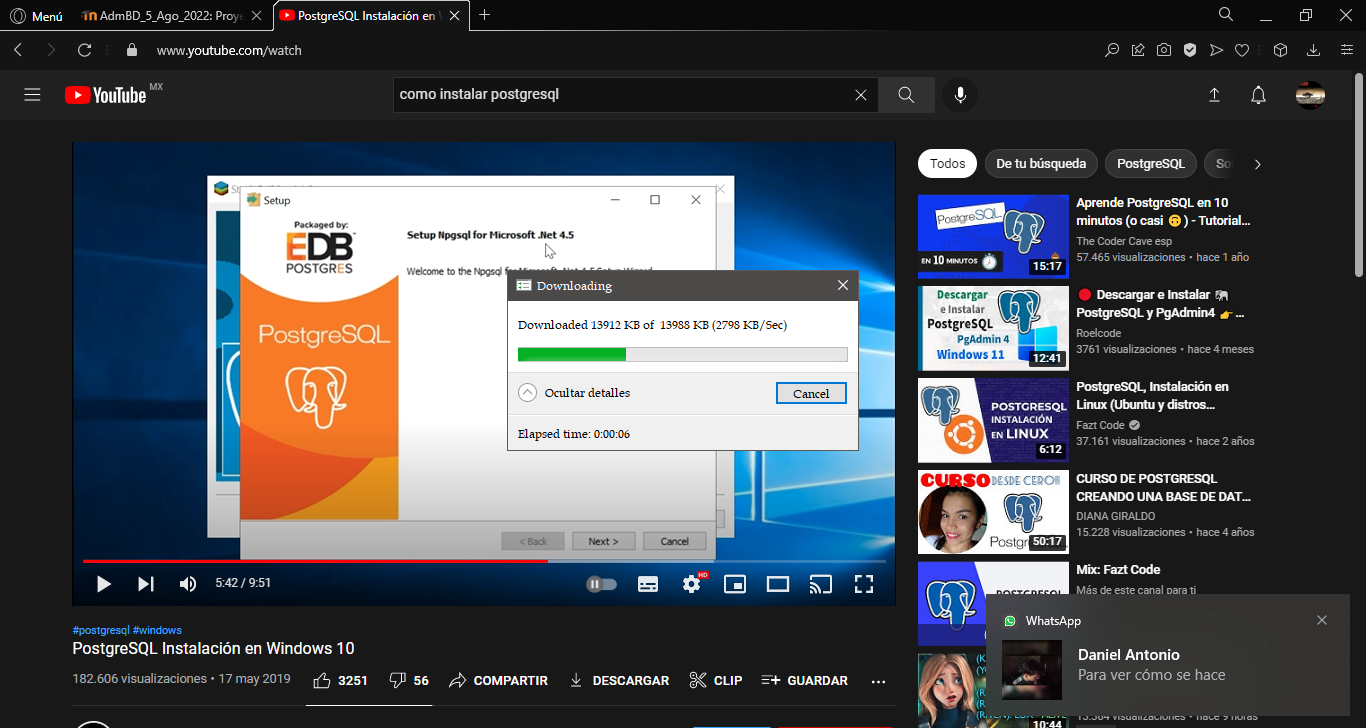


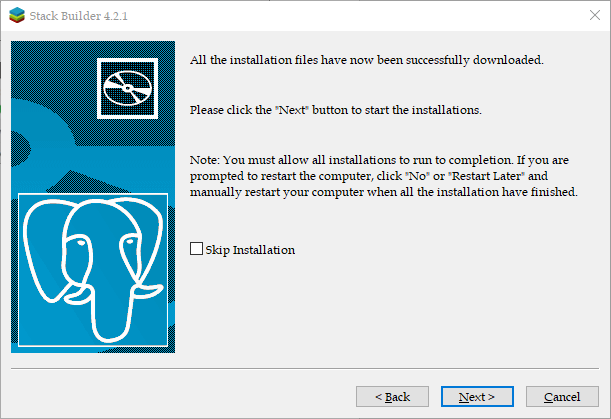
En las aplicaciones a instalar marcamos solo los drivers de la base de datos. Next



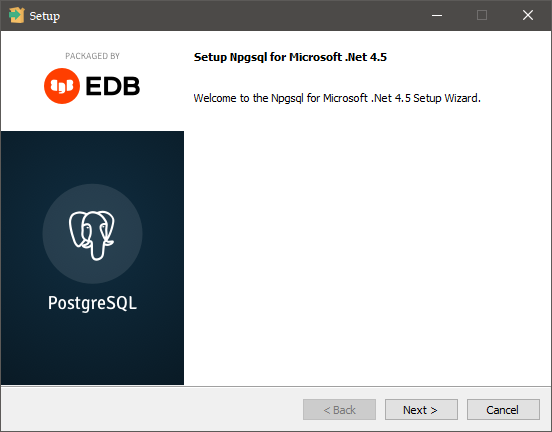
Más instaladores. Next



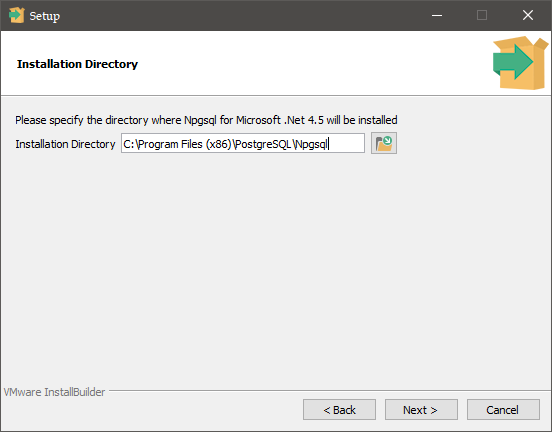
Esperamos

Next

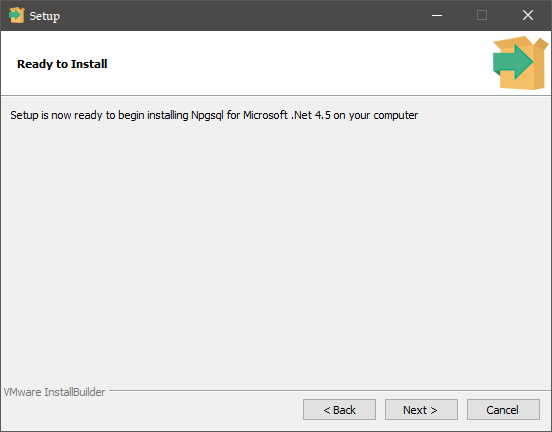
Next en el instalador



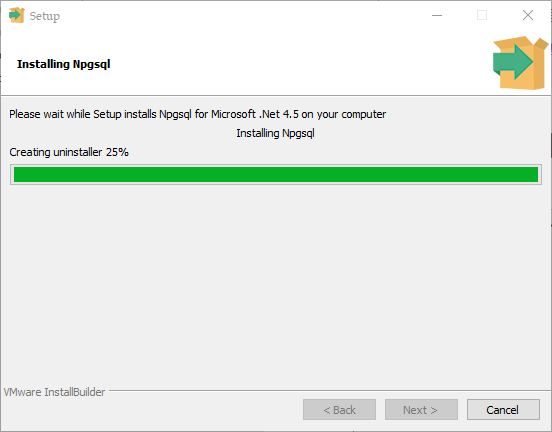
Next



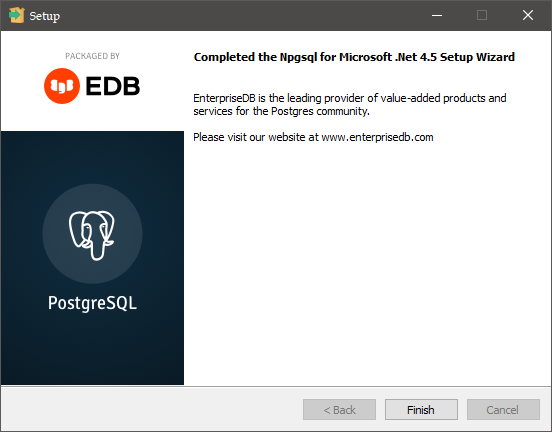
Next



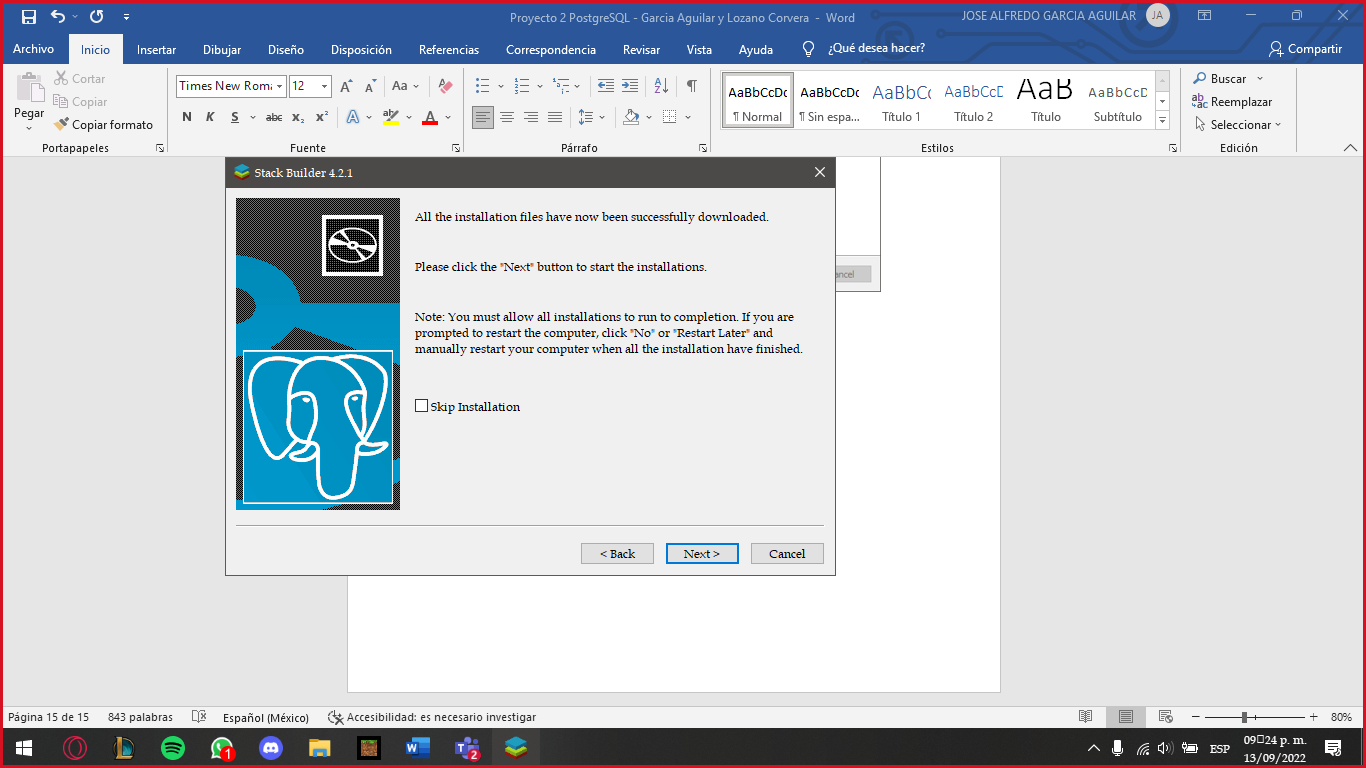
Esperamos



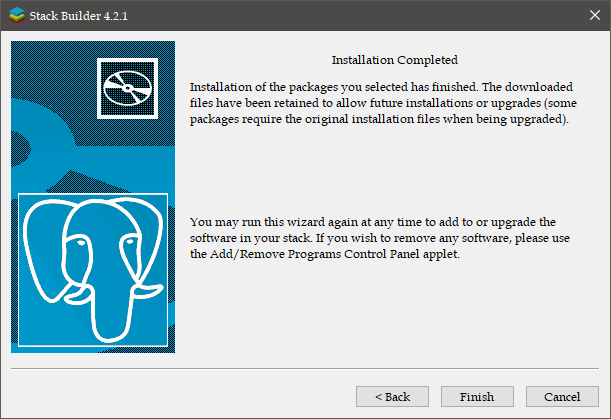
Finish



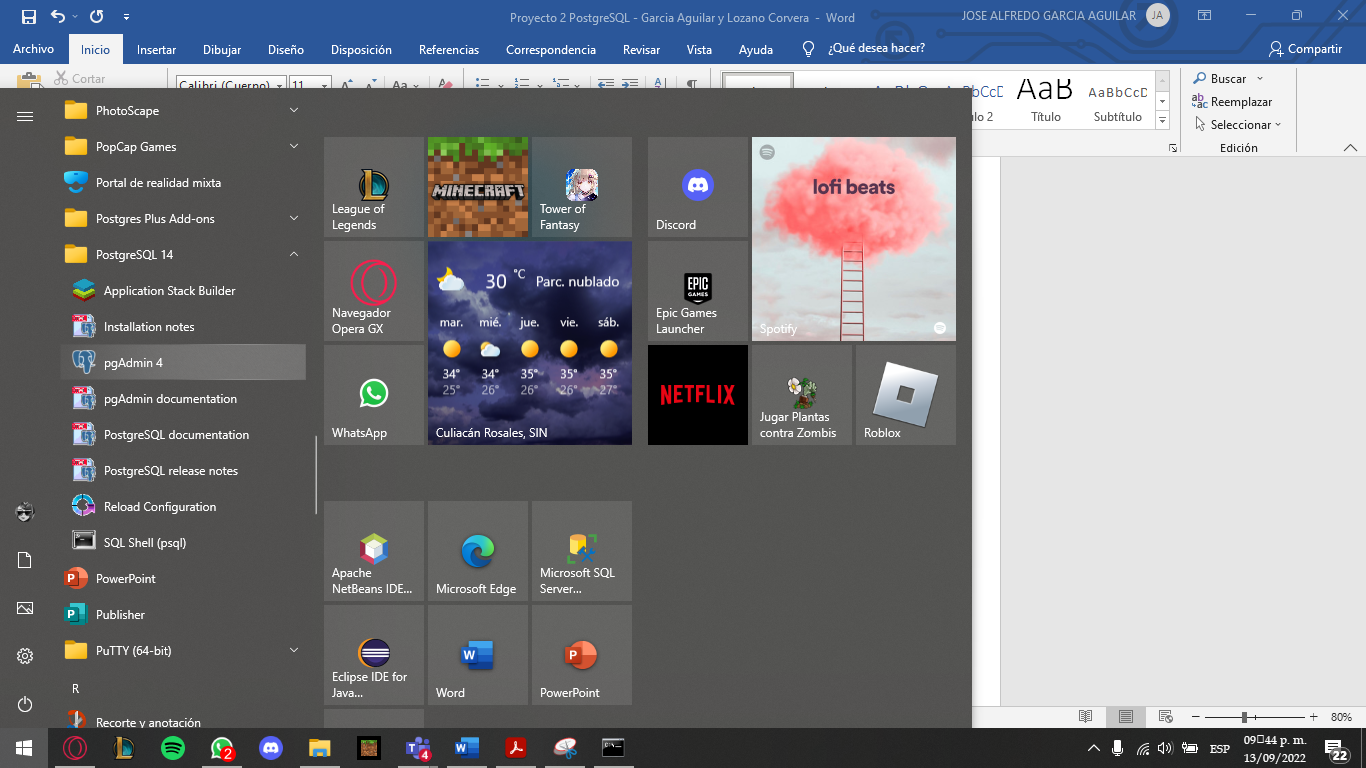
Next



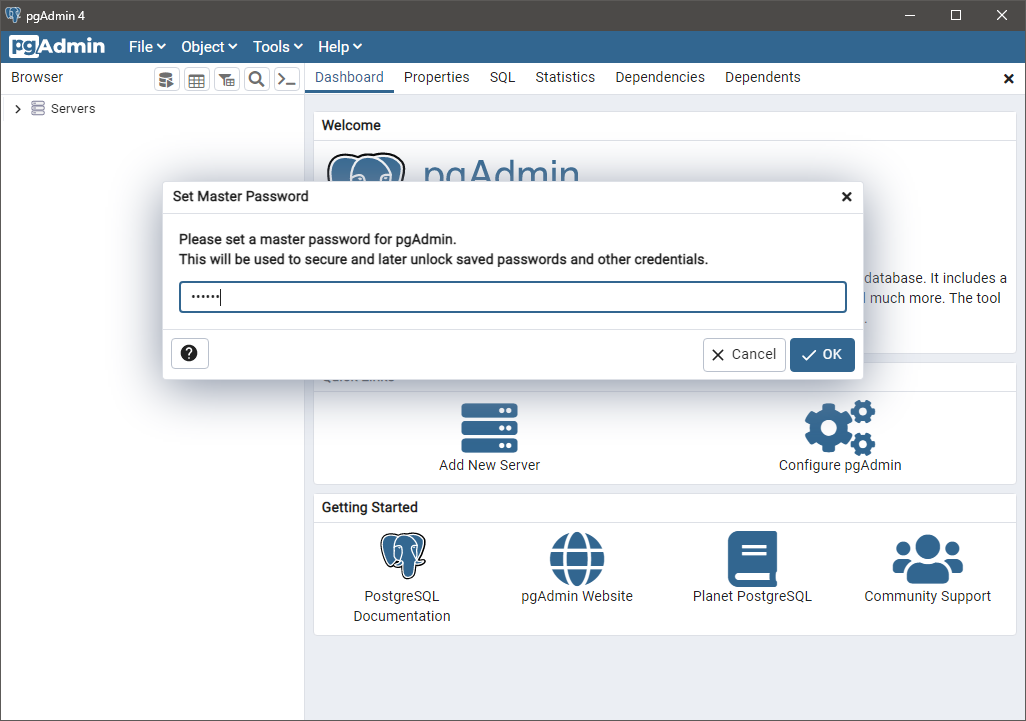
Hacemos ese paso las 4 veces que nos piden. Damos Finish



Lo abrimos

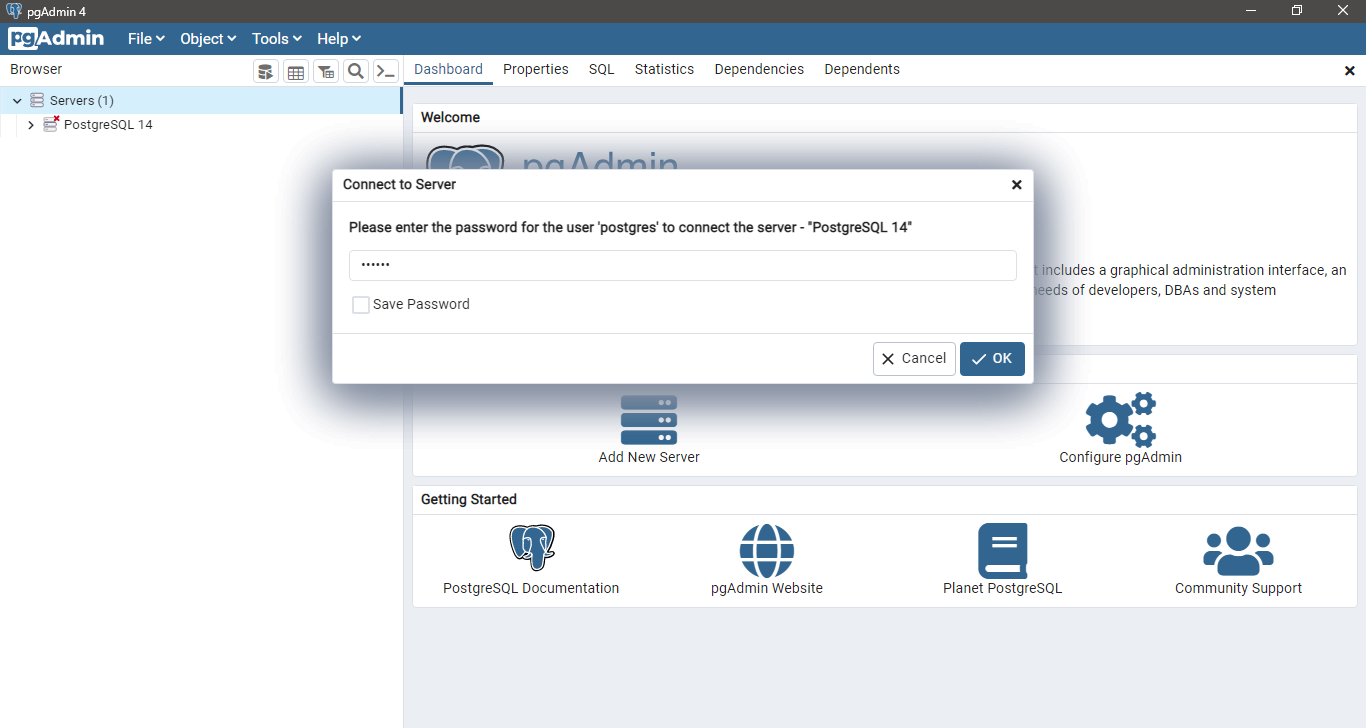


Ya estando dentro nos pide la clave que pusimos, 123456. Damos ok

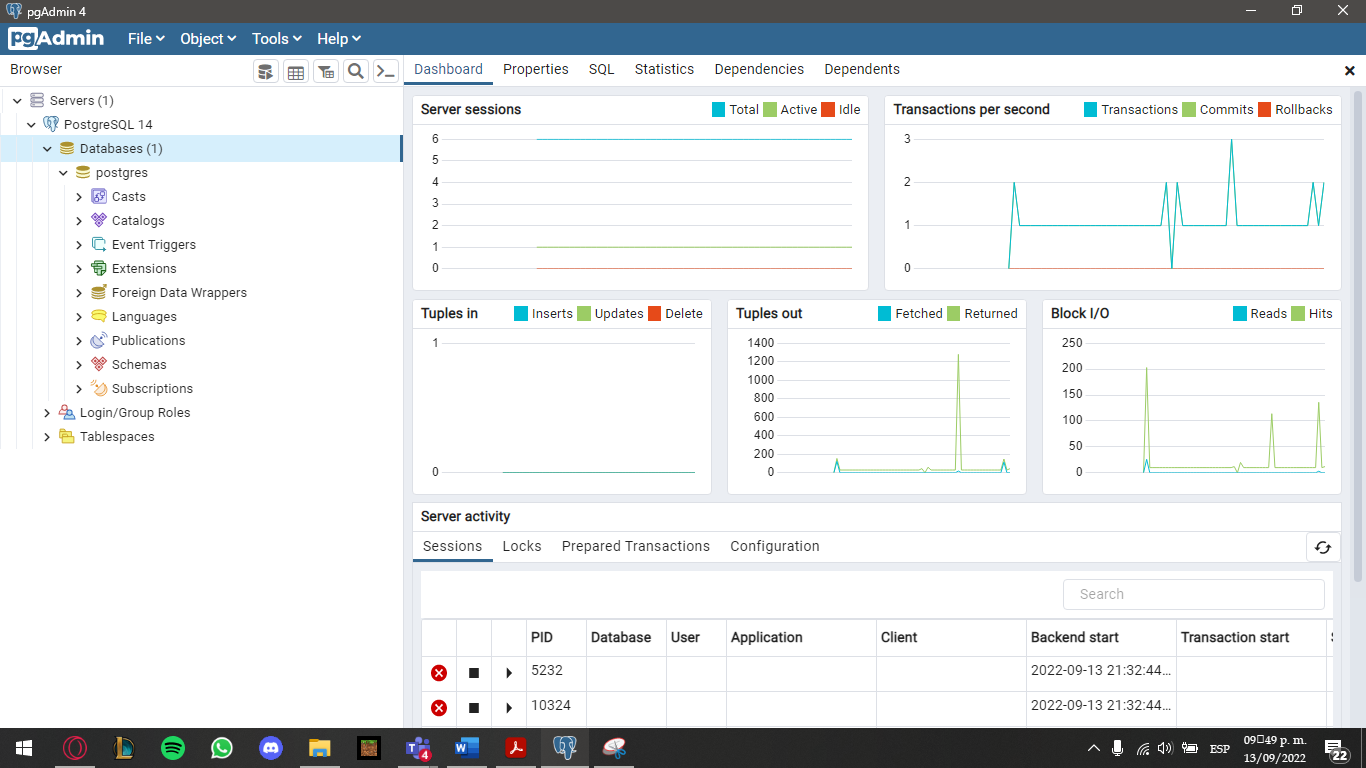


1. **MANUAL DE COMO REALIZAR UNA CONEXIÓN AL SERVIDOR Y COMO EJECUTAR INSTRUCCIONES SQL.**

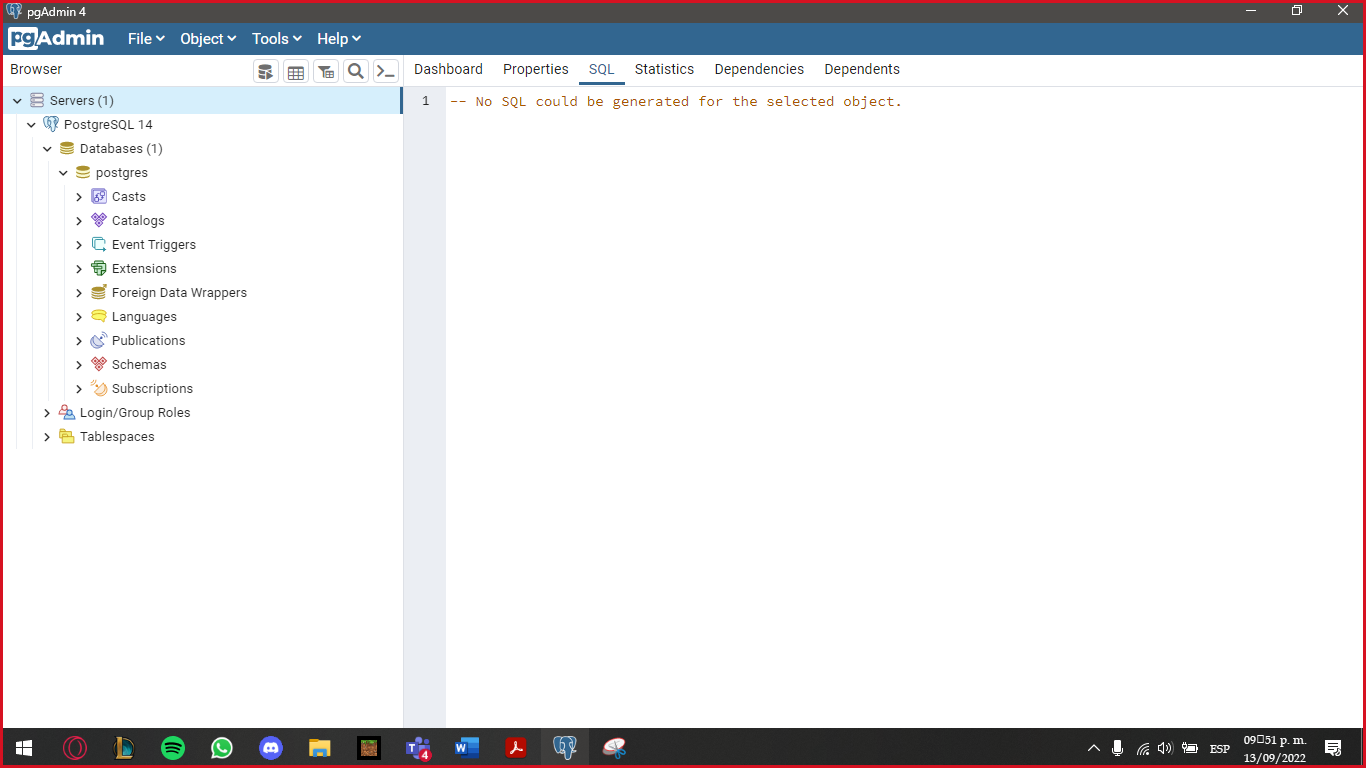
Hacemos doble click en Servers y ponemos de nuevo nuestra clave, 123456



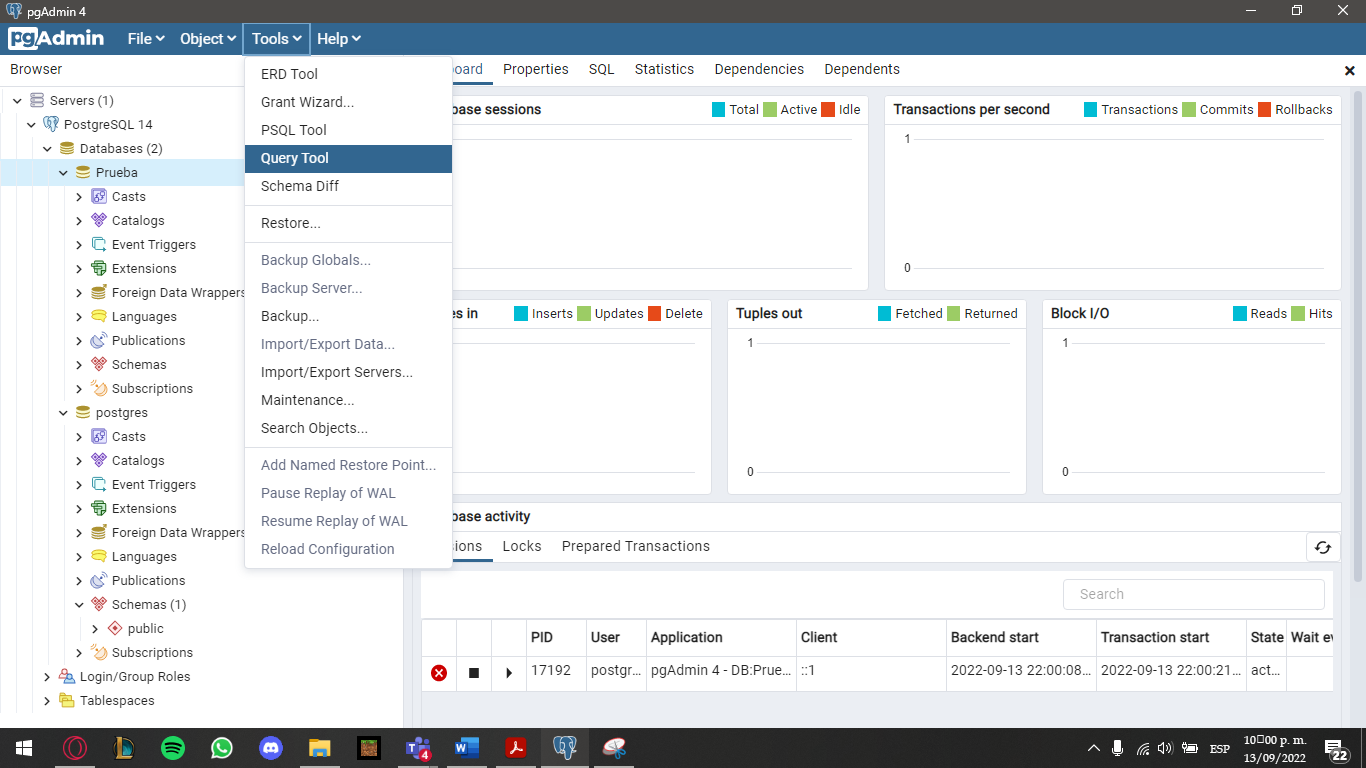
Nos aparecerá asi:



Hacemos click arriba en SQL

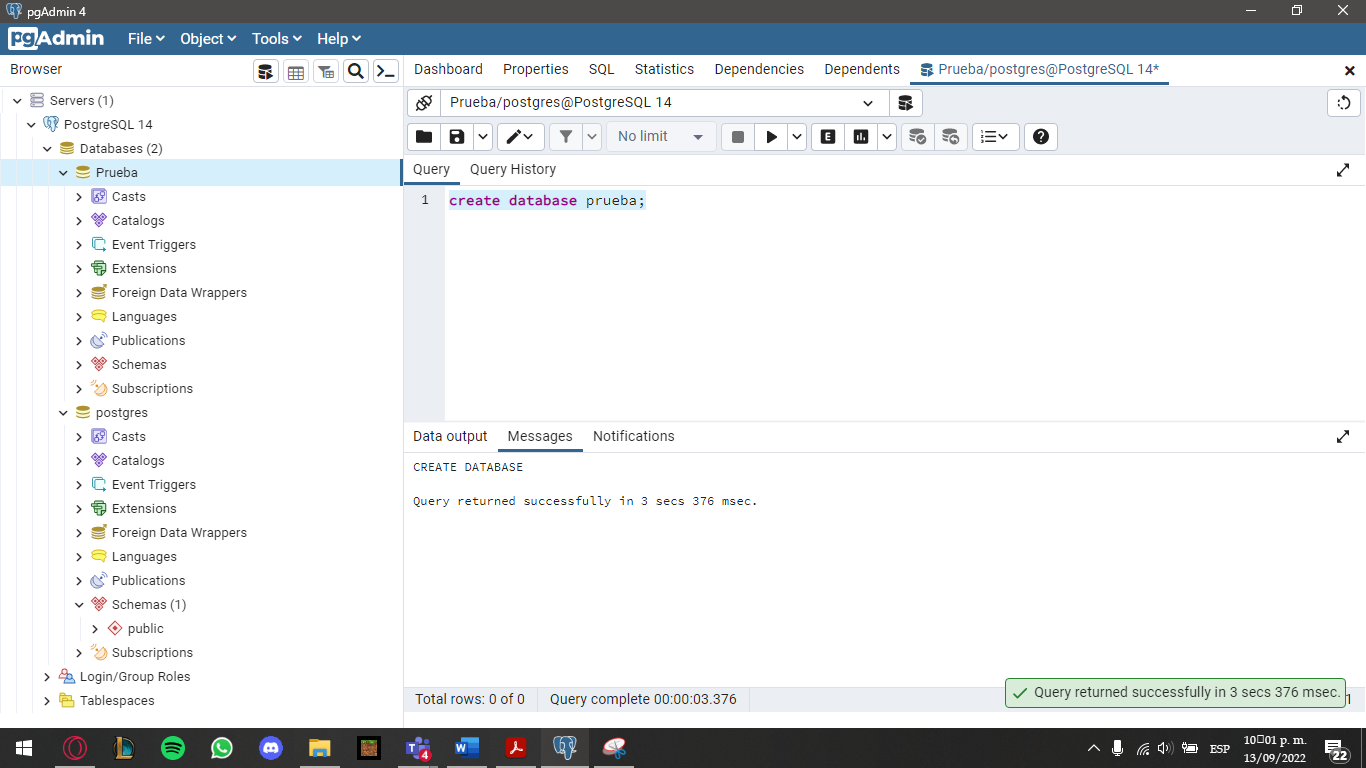


Para poder crear y editar un query creamos una base de datos, vamos arriba seleccionándola y damos en query tools



Cómo ejecutar instrucciones SQL:

Al igual que en SQL Server y MySQL, la marcamos con el mouse y corremos con la flechita



1. **CREACIÓN DE USUARIOS Y OTORGARLE PERMISO DE EJECUTAR SOLO SELECT/INSERT.**

Creamos 2 usuarios para probar, le dimos permisos de seleccionar e insertar manualmente a todas las tablas de la Northwind

Script usuario1:

--Crear usuario

create user usuario1 with password '123456';

--Dar permisos de seleccionar e insertar a todas las tablas

grant select on categories to usuario1;

grant insert on categories to usuario1;

grant select on suppliers to usuario1;

grant insert on suppliers to usuario1;

grant select on products to usuario1;

grant insert on products to usuario1;

grant select on shippers to usuario1;

grant insert on shippers to usuario1;

grant select on customers to usuario1;

grant insert on customers to usuario1;

grant select on employees to usuario1;

grant insert on employees to usuario1;

grant select on region to usuario1;

grant insert on region to usuario1;

grant select on territories to usuario1;

grant insert on territories to usuario1;

grant select on employeeterritories to usuario1;

grant insert on employeeterritories to usuario1;

grant select on orderdetails to usuario1;

grant insert on orderdetails to usuario1;

grant select on orders to usuario1;

grant insert on orders to usuario1;

grant select on customercustomerdemo to usuario1;

grant insert on customercustomerdemo to usuario1;

grant select on customerdemographics to usuario1;

grant insert on customerdemographics to usuario1;

Script usuario2:

--Crear usuario

create user usuario2 with password '123456';

--Dar permisos de seleccionar e insertar a todas las tablas

grant select on categories to usuario2;

grant insert on categories to usuario2;

grant select on suppliers to usuario2;

grant insert on suppliers to usuario2;

grant select on products to usuario2;

grant insert on products to usuario2;

grant select on shippers to usuario2;

grant insert on shippers to usuario2;

grant select on customers to usuario2;

grant insert on customers to usuario2;

grant select on employees to usuario2;

grant insert on employees to usuario2;

grant select on region to usuario2;

grant insert on region to usuario2;

grant select on territories to usuario2;

grant insert on territories to usuario2;

grant select on employeeterritories to usuario2;

grant insert on employeeterritories to usuario2;

grant select on orderdetails to usuario2;

grant insert on orderdetails to usuario2;

grant select on orders to usuario2;

grant insert on orders to usuario2;

grant select on customercustomerdemo to usuario2;

grant insert on customercustomerdemo to usuario2;

grant select on customerdemographics to usuario2;

grant insert on customerdemographics to usuario2;

1. **SCRIPT PARA CARGAR LA BD NORTHWIND (RESUMEN CON LOS TIPOS DE DATOS QUE CAMBIARON)**

Cambios que hicimos:

* Quitar las comillas de los nombres de las tablas
* Cambiar los nombres de algunos tipos de datos, por ejemplo en la clave primaria pasamos de identity a serial, de datetime cambiamos a timestamp, entre otros pocos
* Se quitan los Clustered o Not Clustered
* Cambiar la palabra go por ; (punto y coma)
* Los constraint que teníamos se cambian a solamente Default y con el valor que tomará ese campo

Script de la Northwind en Postgres:

create table categories (

categoryid serial not null ,

categoryname varchar(15) not null ,

description text null ,

constraint pk\_categories primary key

(

categoryid

)

)

;

create table suppliers (

supplierid serial not null ,

companyname varchar(40) not null ,

contactname varchar(30) null ,

contacttitle varchar(30) null ,

address varchar(60) null ,

city varchar(15) null ,

region varchar(15) null ,

postalcode varchar(10) null ,

country varchar(15) null ,

phone varchar(24) null ,

fax varchar(24) null ,

homepage text null ,

constraint pk\_suppliers primary key

(

supplierid

)

)

;

create table products (

productid serial not null ,

productname varchar(40) not null ,

supplierid int null ,

categoryid int null ,

quantityperunit varchar(20) null ,

unitprice money null constraint df\_products\_unitprice default (0),

unitsinstock smallint null constraint df\_products\_unitsinstock default (0),

unitsonorder smallint null constraint df\_products\_unitsonorder default (0),

reorderlevel smallint null constraint df\_products\_reorderlevel default (0),

discontinued boolean not null constraint df\_products\_discontinued default (false),

constraint pk\_products primary key

(

productid

),

constraint fk\_products\_categories foreign key

(

categoryid

) references categories (

categoryid

),

constraint fk\_products\_suppliers foreign key

(

supplierid

) references suppliers (

supplierid

),

constraint ck\_products\_unitprice check (unitprice >= '0'),

constraint ck\_reorderlevel check (reorderlevel >= 0),

constraint ck\_unitsinstock check (unitsinstock >= 0),

constraint ck\_unitsonorder check (unitsonorder >= 0)

)

;

create table employees (

employeeid serial not null ,

lastname varchar(20) not null ,

firstname varchar(10) not null ,

title varchar(30) null ,

titleofcourtesy varchar(25) null ,

birthdate timestamp null ,

hiredate timestamp null ,

address varchar(60) null ,

city varchar(15) null ,

region varchar(15) null ,

postalcode varchar(10) null ,

country varchar(15) null ,

homephone varchar(24) null ,

extension varchar(4) null ,

notes text null ,

reportsto int null ,

photopath varchar(255) null ,

constraint pk\_employees primary key

(

employeeid

),

constraint fk\_employees\_employees foreign key

(

reportsto

) references employees (

employeeid

),

constraint ck\_birthdate check ( birthdate < now() )

)

;

create table customers (

customerid nchar (5) not null ,

companyname varchar(40) not null ,

contactname varchar(30) null ,

contacttitle varchar(30) null ,

address varchar(60) null ,

city varchar(15) null ,

region varchar(15) null ,

postalcode varchar(10) null ,

country varchar(15) null ,

phone varchar(24) null ,

fax varchar(24) null ,

constraint pk\_customers primary key

(

customerid

)

)

;

create table shippers (

shipperid serial not null ,

companyname varchar(40) not null ,

phone varchar(24) null ,

constraint pk\_shippers primary key

(

shipperid

)

)

;

create table orders (

orderid serial not null ,

customerid nchar (5) null ,

employeeid int null ,

orderdate timestamp null ,

requireddate timestamp null ,

shippeddate timestamp null ,

shipvia int null ,

freight money null constraint df\_orders\_freight default (0),

shipname varchar(40) null ,

shipaddress varchar(60) null ,

shipcity varchar(15) null ,

shipregion varchar(15) null ,

shippostalcode varchar(10) null ,

shipcountry varchar(15) null ,

constraint pk\_orders primary key

(

orderid

),

constraint fk\_orders\_customers foreign key

(

customerid

) references customers (

customerid

),

constraint fk\_orders\_employees foreign key

(

employeeid

) references employees (

employeeid

),

constraint fk\_orders\_shippers foreign key

(

shipvia

) references shippers (

shipperid

)

)

;

create table orderdetails(

orderid int not null ,

productid int not null ,

unitprice money not null constraint df\_order\_details\_unitprice default (0),

quantity smallint not null constraint df\_order\_details\_quantity default (1),

discount real not null constraint df\_order\_details\_discount default (0),

constraint pk\_order\_details primary key

(

orderid,

productid

),

constraint fk\_order\_details\_orders foreign key

(

orderid

) references orders (

orderid

),

constraint fk\_order\_details\_products foreign key

(

productid

) references products (

productid

),

constraint ck\_discount check (discount >= 0 and (discount <= 1)),

constraint ck\_quantity check (quantity > 0),

constraint ck\_unitprice check (unitprice >= '0')

)

;

/\* the following adds stored procedures \*/

create table region

( regionid int not null ,

regiondescription nchar (50) not null

)

;

create table territories

(territoryid varchar (20) not null ,

territorydescription nchar (50) not null ,

regionid int not null

)

;

create table employeeterritories

(employeeid int not null,

territoryid varchar (20) not null

)

;

alter table region

add constraint pk\_region primary key

(

regionid

)

;

alter table territories

add constraint pk\_territories primary key

(

territoryid

)

;

alter table territories

add constraint fk\_territories\_region foreign key

(

regionid

) references region (

regionid

)

;

alter table employeeterritories

add constraint pk\_employeeterritories primary key

(

employeeid,

territoryid

)

;

alter table employeeterritories

add constraint fk\_employeeterritories\_employees foreign key

(

employeeid

) references employees (

employeeid

)

;

alter table employeeterritories

add constraint fk\_employeeterritories\_territories foreign key

(

territoryid

) references territories (

territoryid

)

;

create table customercustomerdemo

(customerid nchar (5) not null,

customertypeid nchar (10) not null

)

;

create table customerdemographics

(customertypeid nchar (10) not null ,

customerdesc text null

)

;

alter table customercustomerdemo

add constraint pk\_customercustomerdemo primary key

(

customerid,

customertypeid

)

;

alter table customerdemographics

add constraint pk\_customerdemographics primary key

(

customertypeid

)

;

alter table customercustomerdemo

add constraint fk\_customercustomerdemo foreign key

(

customertypeid

) references customerdemographics (

customertypeid

)

;

alter table customercustomerdemo

add constraint fk\_customercustomerdemo\_customers foreign key

(

customerid

) references customers (

customerid

)

;

1. **SCRIPT PARA CREAR LA FAMILIA DE VISTAS DE LA BD NORTHWIND**

create view vw\_products as

select p.productid, p.productname, p.quantityperunit, p.unitprice as produnitprice,

p.unitsinstock, p.unitsonorder, p.reorderlevel, p.discontinued,

s.supplierid, s.companyname, s.contactname, s.contacttitle, s.address, s.city, s.region,

s.postalcode, s.country, s.phone, s.fax, s.homepage, c.categoryid, c.categoryname, c.description

from products p

inner join suppliers s on p.supplierid = s.supplierid

inner join categories c on p.categoryid = c.categoryid

;

create view vw\_orders as

select o.orderid, o.orderdate, o.requireddate, o.shippeddate, o.freight, o.shipname, o.shipaddress, o.shipcity,

o.shipregion, s.shipperid, s.companyname as nomcomenvio, s.phone as envphone, c.customerid, c.companyname as nomcliente,

c.contactname as ctecontactname, c.contacttitle as ctecontacttitle,

c.address as cteaddress, c.city as ctecity, c.region as cteregion, c.postalcode as ctepostalcode,

c.country as ctecountry, c.phone as ctephone, c.fax as ctefax, e.employeeid, e.lastname, e.firstname,

e.title, e.titleofcourtesy, e.birthdate, e.hiredate,

e.address as empaddress, e.city as empcity, e.region as empregion, e.postalcode as emppostalcode,

e.country as empcountry, e.homephone, e.extension, e.reportsto

from orders o

inner join shippers s on o.shipvia = s.shipperid

inner join customers c on o.customerid = c.customerid

inner join employees e on o.employeeid = e.employeeid

;

create view vw\_order\_details as

select d.quantity, d.unitprice, d.discount, p.\*, o.\*

from orderdetails d

inner join vw\_products p on d.productid = p.productid

inner join vw\_orders o on d.orderid = o.orderid

;

1. **SCRIPT CON EL PROCEDIMIENTO ALMACENADO QUE INSERTE Y MODIFIQUE LA TABLA TERRITORIES**

create procedure sp\_mttoterritories(inout territoryid\_v varchar(20), in territorydescription\_v nchar(50), in territoryregion\_v int)

as

$$

begin

if exists(select \* from territories where territoryid = territoryid\_v)

then

update territories set territoryid = territoryid\_v, territorydescription = territorydescription\_v, regionid = territoryregion\_v

where territoryid = territoryid\_v;

else

insert into territories(territoryid, territorydescription, regionid) values (territoryid\_v, territorydescription\_v, territoryregion\_v);

end if;

end

$$

language 'plpgsql'; --PL/pgSQL (Procedural Language/PostgreSQL)

1. **SCRIPT CON UN TRIGGER QUE NO PERMITA ELIMINAR TERRITORIES**

create function fn\_eliminar\_territories() returns trigger

as

$$

begin

raise exception 'Por el momento no se puede eliminar registros';

end

$$

language plpgsql;

create trigger tr\_eliminar\_territories

before delete on territories

for each row

execute procedure fn\_eliminar\_territories();