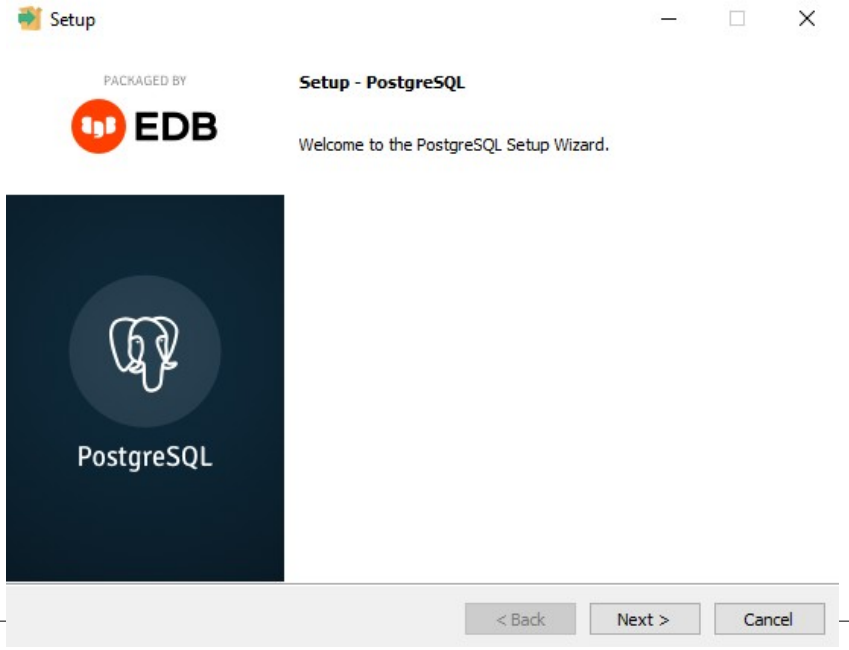
Usando o Banco PostGreSQL

Introdução

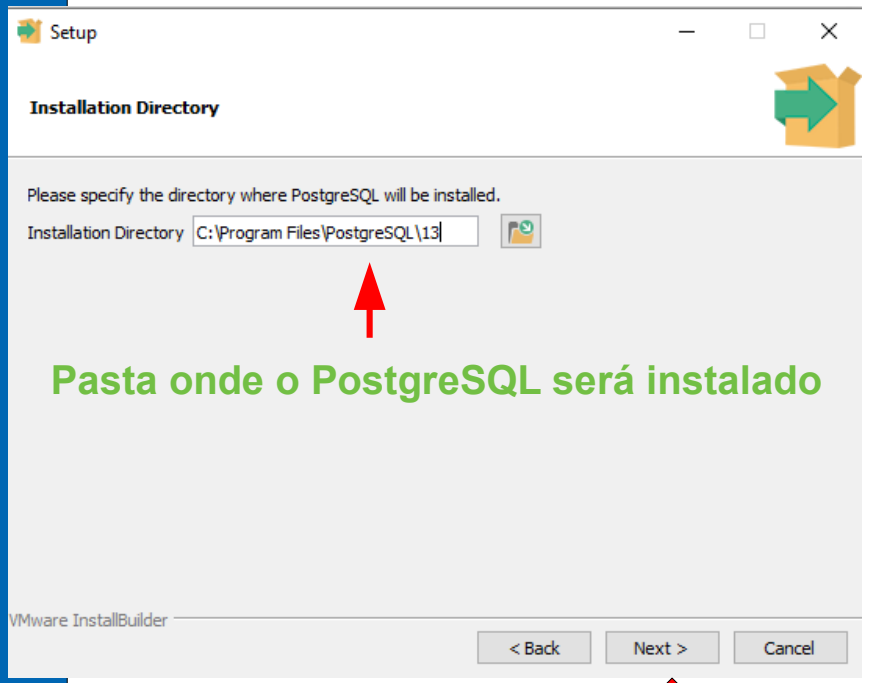
- Até o momento temos usado o banco H2, que trabalha com memória volátil (memória ram). Precisaremos a partir daqui de utilizar um banco de dados “tradicional”.
- Entre as diversas opções temos o PostGreSQL. O PostGreSQL é um sistema de gerenciamento de dados desenvolvido como projeto de código aberto. O PostGreSQL possui as funcionalidades avançadas para tratamento de dados (integridade transacional, triggers, stored procedures, etc.).
- O PostGreSQL tem adquido bastante prestígio na comunidade Linux tendo recebido diversos prêmios.
- O PostGreSQL é uma ótima opção para quem busca um banco de dados confiável, seguro e de baixo custo para utilização e manutenção.

Instalação

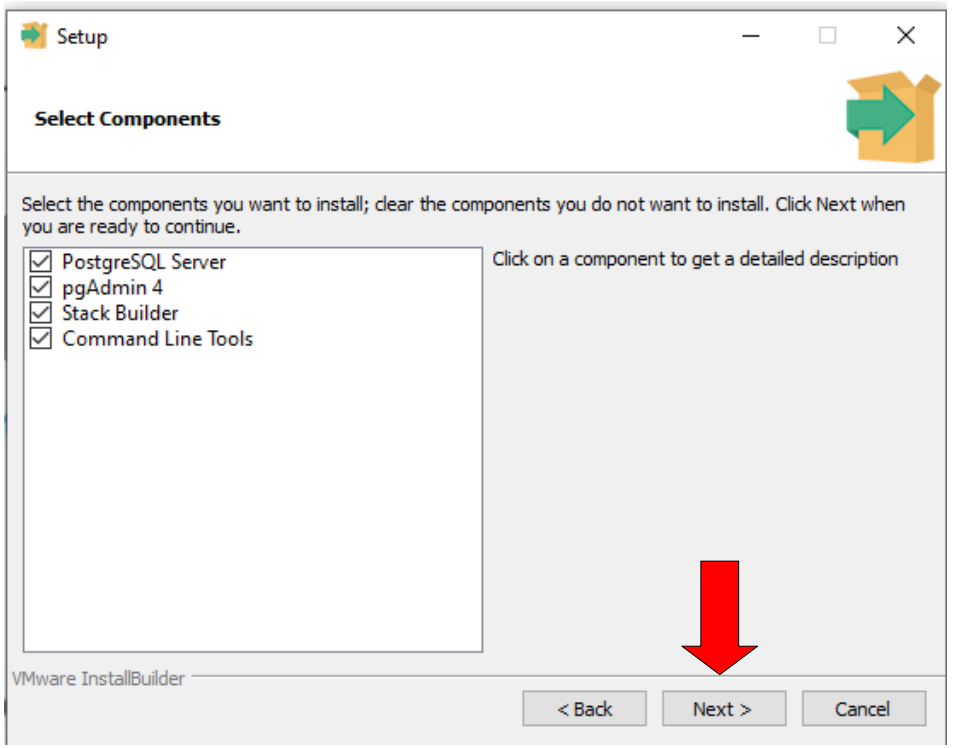
- O primeiro passo é baixar o instalador do PostGreSQL em: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- Sugiro baixar a última versão referente ao seu sistema operacional (as telas seguintes serão da instalação em Windows):

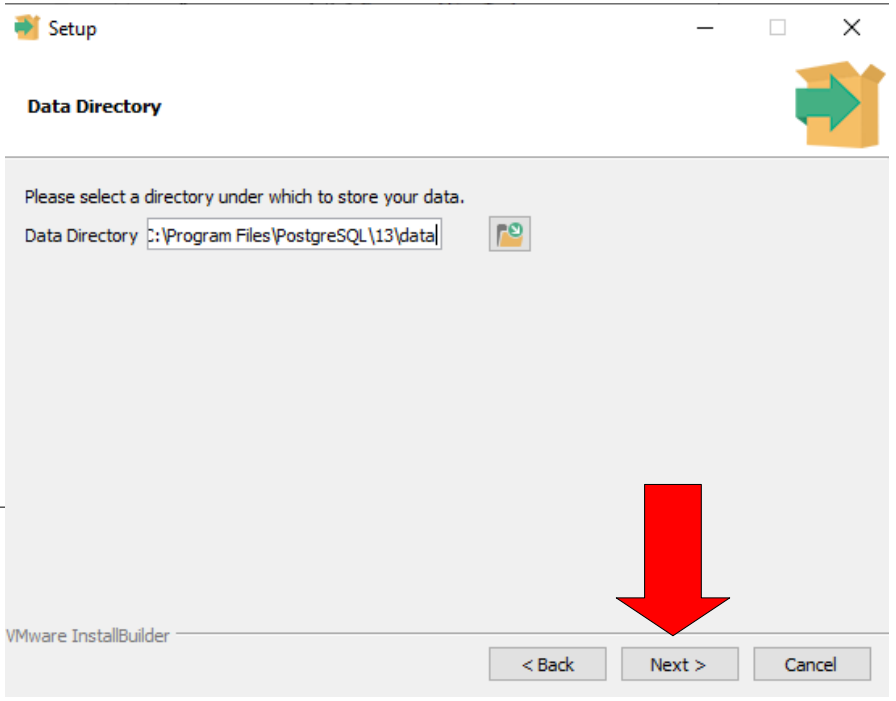


Instalação



Pasta onde o PostgreSQL será instalado

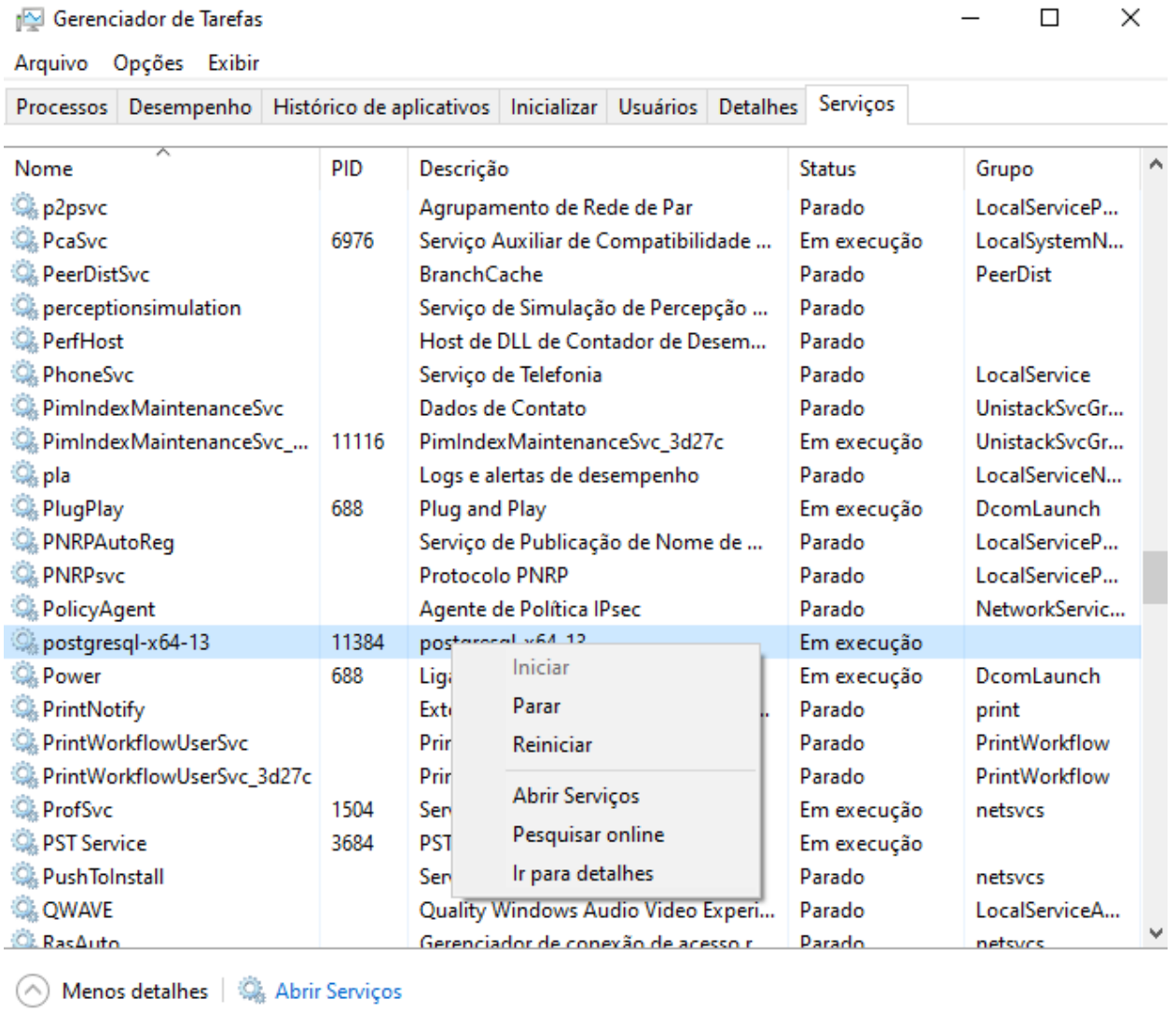




Instalação

- Na sequência defina a senha para o superusuário do PostgreSQL.
- Informe também a porta por onde o banco de dados será acessado (sugiro manter 5432).
- Nos próximos passos é só fazer Next, Next, Next.
- Na última tela (ao lado), desmarque a opção pois não a utilizaremos.

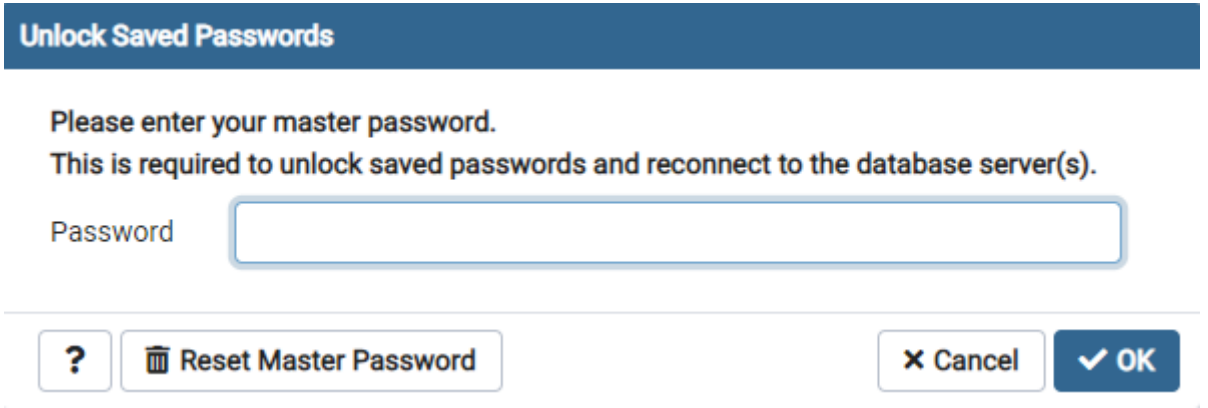
Iniciando ou parando o PostGreSQL



O PostGreSQL é um serviço rodando na máquina como qualquer outro, dessa forma, para que o banco funcione o serviço precisa estar em execução

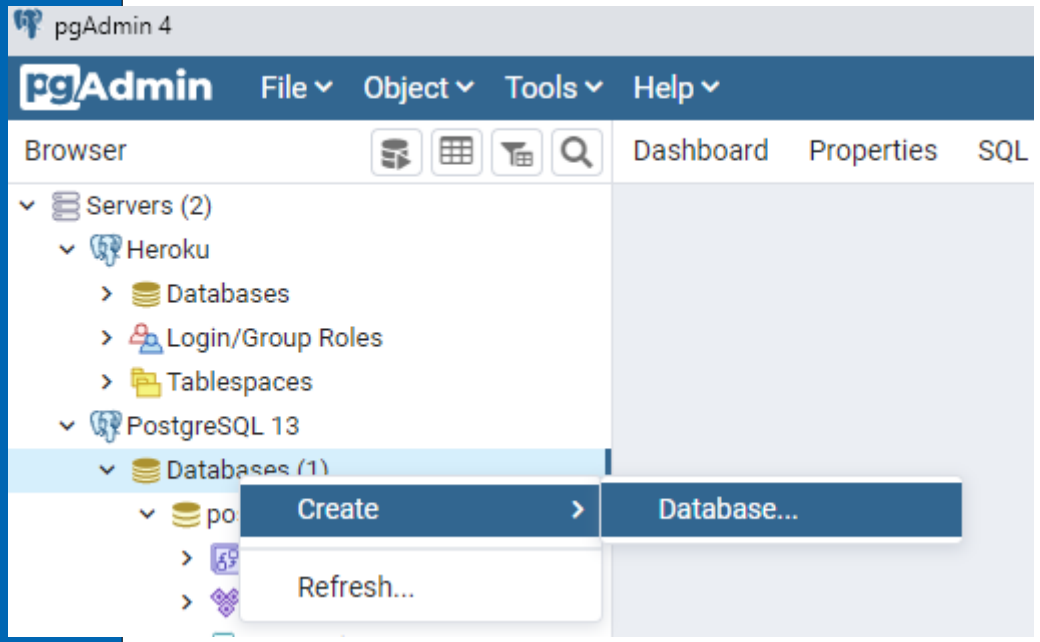
Usando o pgAdmin

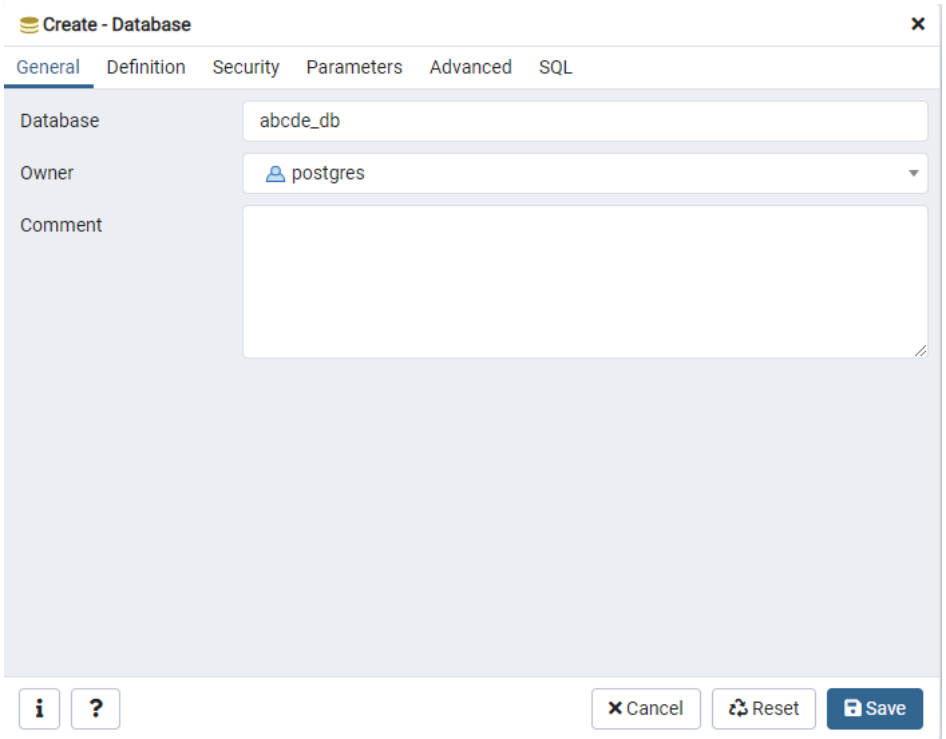
- O pgAdmin é o painel de administração de bancos de dados PostGreSQL.
- O primeiro passo para utilizá-lo é informar a senha do superusuário (definido durante a instalação do PostGreeSQL – slide 5):

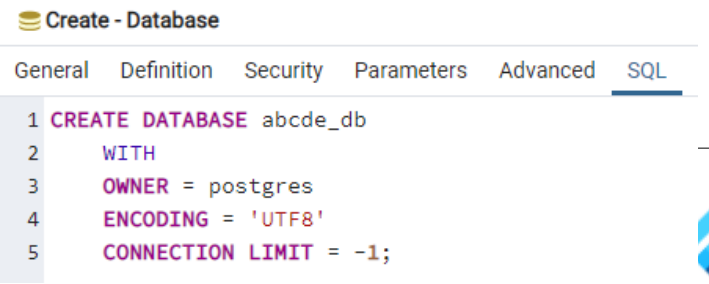


Usando o pgAdmin

- O passo seguinte é criar o nosso banco de dados. Vamos criar o banco abcde_db:

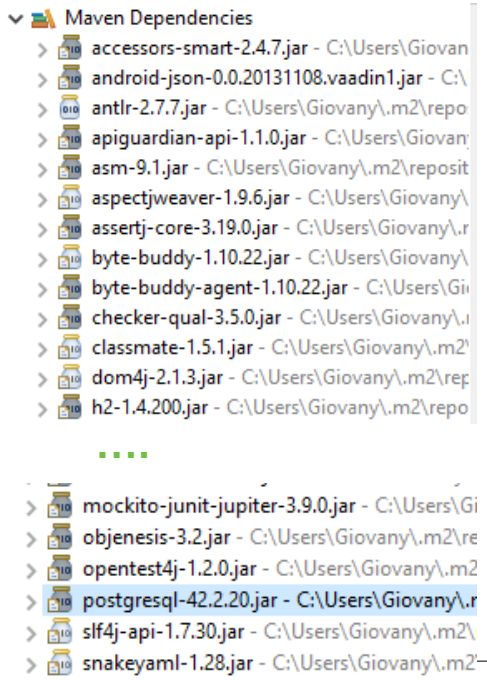






Acrescentando a dependência do PostGreSQL

- No arquivo **pom.xml** acrescentamos a dependência do PostGreSQL (quadro vermelho).
- Dessa forma o Maven buscará essa dependência:



abcde/pom.xml

```
18          <java.version>11</java.version>
19      </properties>
20      <dependencies>
21          <dependency>
22              <groupId>org.springframework.boot</groupId>
23              <artifactId>spring-boot-starter-web</artifactId>
24          </dependency>
25
26          <dependency>
27              <groupId>org.springframework.boot</groupId>
28              <artifactId>spring-boot-starter-test</artifactId>
29              <scope>test</scope>
30          </dependency>
31
32          <dependency>
33              <groupId>org.springframework.boot</groupId>
34              <artifactId>spring-boot-starter-data-jpa</artifactId>
35          </dependency>
36          <dependency>
37              <groupId>com.h2database</groupId>
38              <artifactId>h2</artifactId>
39              <scope>runtime</scope>
40          </dependency>
41
42          <dependency>
43              <groupId>org.postgresql</groupId>
44              <artifactId>postgresql</artifactId>
45              <scope>runtime</scope>
46          </dependency>
47      </dependencies>
48
```

application-dev.properties

- O próximo passo é criar o arquivo de configuração para usar o banco PostGreSQL.
- Vamos lembrar que no uso do banco H2 criamos o arquivo application-test.properties:

application-test.properties

```
1 spring.datasource.url=jdbc:h2:mem:abcdedb
2 spring.datasource.username=admin
3 spring.datasource.password=
4
5 spring.h2.console.enabled=true
6 spring.h2.console.path=/h2-console
7
8 spring.jpa.show-sql=true
9 spring.jpa.properties.hibernate.format_sql=true
10
```


application-dev.properties

- As primeiras 3 linhas definem a **url do banco, o usuário e a senha** (estamos utilizando o superusuário – cadastrado no momento da instalação do PostGreSQL).
- ***spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true*** banco criado de forma não contextual (evita problemas de versão do PostGreSQL).
- ***spring.jpa.hibernate.ddl-auto=update*** atualiza o banco quando necessário (se não houver necessidade não faz nada).
- ***spring.jpa.show-sql=true e spring.jpa.properties.hibernate.format_sql=true*** são para o log do SQL no console do SpringToolWorkspace.

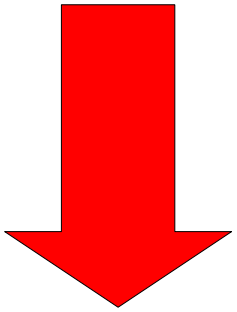
application-dev.properties

```
1  spring.datasource.url=jdbc:postgresql://localhost:5432/abcde_db
2  spring.datasource.username=postgres
3  spring.datasource.password=[redacted]
4
5  spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
6  spring.jpa.hibernate.ddl-auto=update
7  spring.jpa.show-sql=true
8  spring.jpa.properties.hibernate.format_sql=true
```


Perfil Ativo em application.properties

application.properties

```
1 spring.profiles.active=test
2 spring.jpa.open-in-view=true
```



application.properties

```
1 spring.profiles.active=dev
2 spring.jpa.open-in-view=true
```

Rodando o serviço

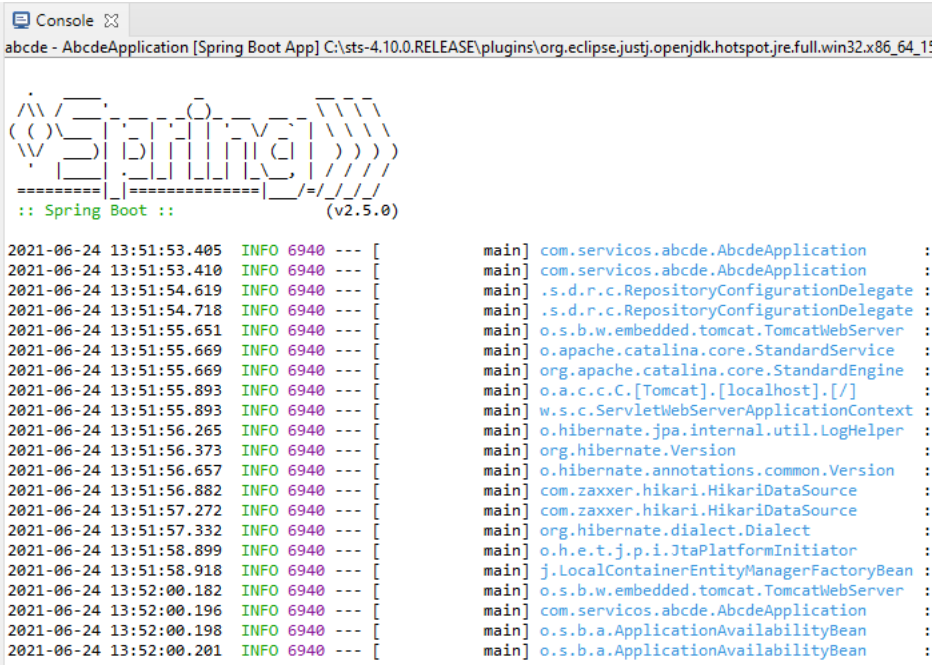
Console

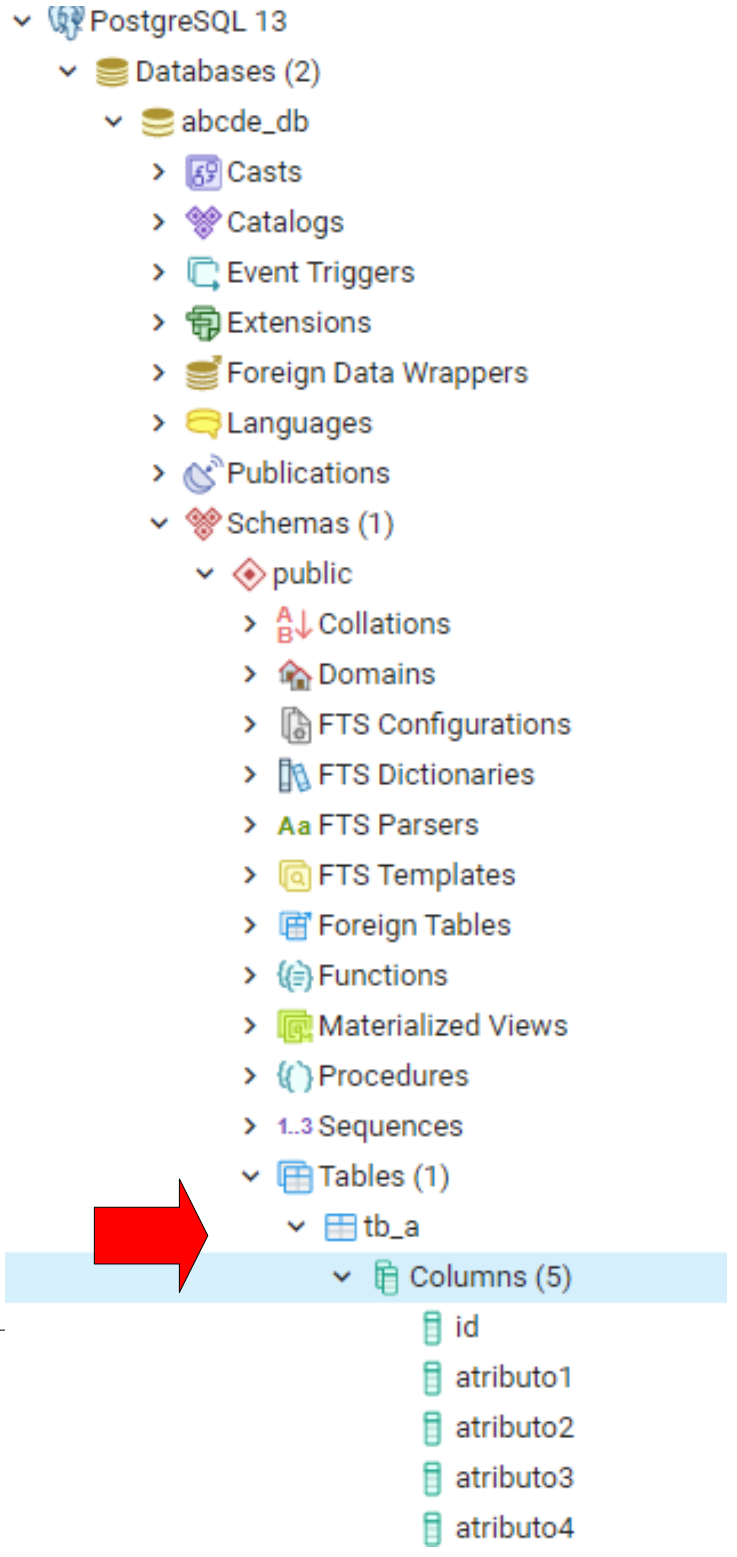
abcde - AbcdeApplication [Spring Boot App] C:\sts-4.10.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (24 de jun. de 2021 13:46:08)

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.0)

2021-06-24 13:46:14.417  INFO 7256 --- [           main] com.servicos.abcde.AbcdeApplication      : Starting AbcdeApplication using Java 15.0.2 on DESKTOP-JKB61SJ with PID 7256 (C:\Users\Giovany\Documents\SpringToolWorkspa
2021-06-24 13:46:14.431  INFO 7256 --- [           main] com.servicos.abcde.AbcdeApplication      : The following profiles are active: dev
2021-06-24 13:46:16.096  INFO 7256 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-06-24 13:46:16.245  INFO 7256 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 126 ms. Found 1 JPA repository interfaces.
2021-06-24 13:46:17.471  INFO 7256 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-06-24 13:46:17.495  INFO 7256 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-06-24 13:46:17.495  INFO 7256 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.46]
2021-06-24 13:46:17.759  INFO 7256 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-06-24 13:46:17.759  INFO 7256 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3188 ms
2021-06-24 13:46:18.301  INFO 7256 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-06-24 13:46:18.510  INFO 7256 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.31.Final
2021-06-24 13:46:18.989  INFO 7256 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-06-24 13:46:19.340  INFO 7256 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-06-24 13:46:19.914  INFO 7256 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-06-24 13:46:19.999  INFO 7256 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQL10Dialect
Hibernate:

    create table tb_a (
       id int8 generated by default as identity,
        atributo1 varchar(255),
        atributo2 float8,
        atributo3 timestamp,
        atributo4 int4,
        primary key (id)
    )
2021-06-24 13:46:22.151  INFO 7256 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2021-06-24 13:46:22.176  INFO 7256 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-06-24 13:46:23.660  INFO 7256 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-06-24 13:46:23.676  INFO 7256 --- [           main] com.servicos.abcde.AbcdeApplication      : Started AbcdeApplication in 10.285 seconds (JVM running for 12.398)
2021-06-24 13:46:23.678  INFO 7256 --- [           main] o.s.b.a.ApplicationAvailabilityBean      : Application availability state LivenessState changed to CORRECT
2021-06-24 13:46:23.681  INFO 7256 --- [           main] o.s.b.a.ApplicationAvailabilityBean      : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
```

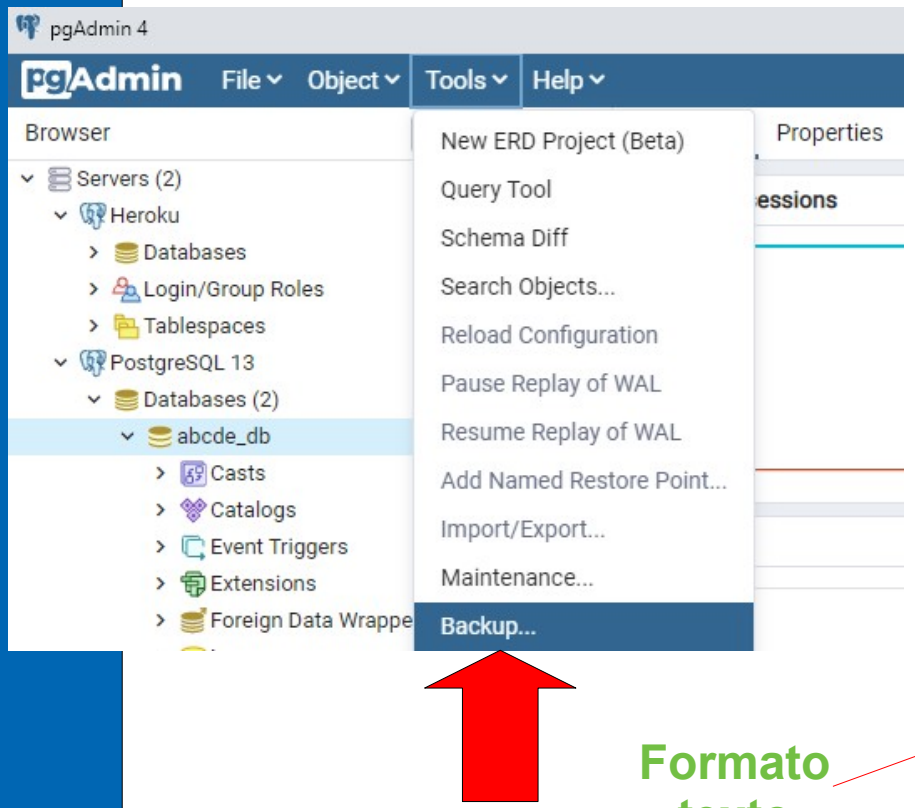
Verificando a criação da tabela

- Notar que se rodarmos o serviço novamente a tabela **tb_a** não será recriada. Isso porque utilizamos a opção: ***spring.jpa.hibernate.ddl-auto=update***

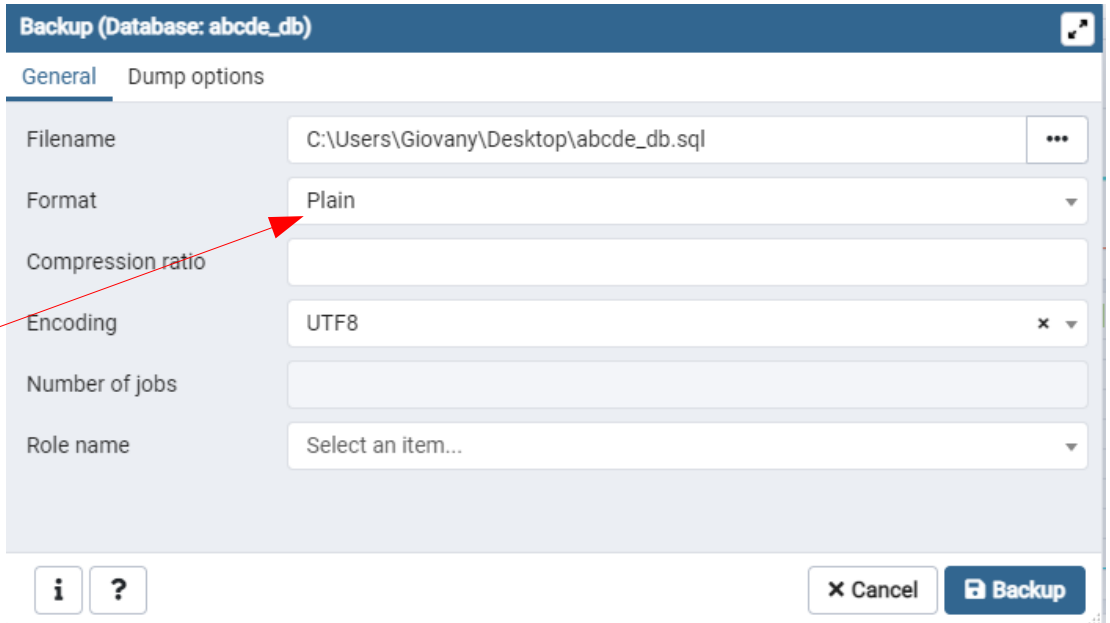




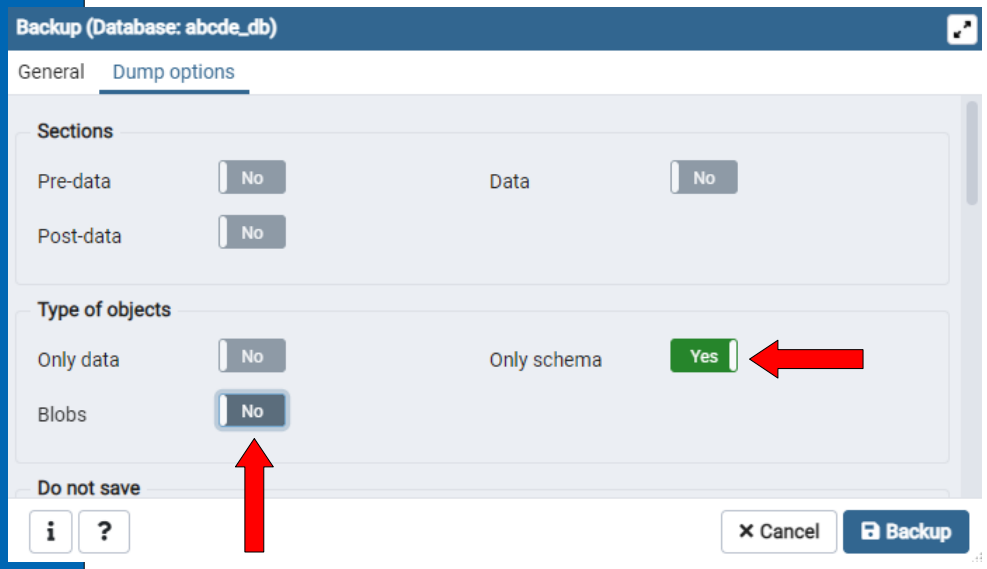
Exportar um banco PostGreSQL

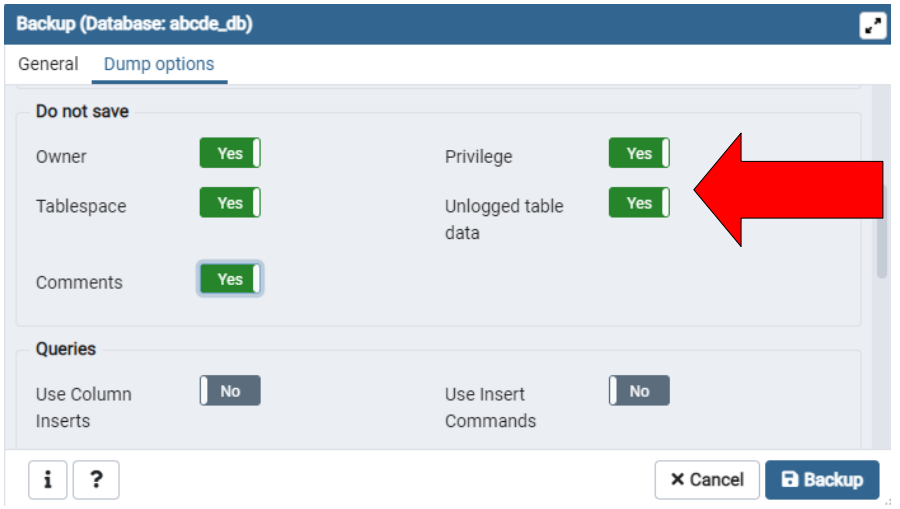


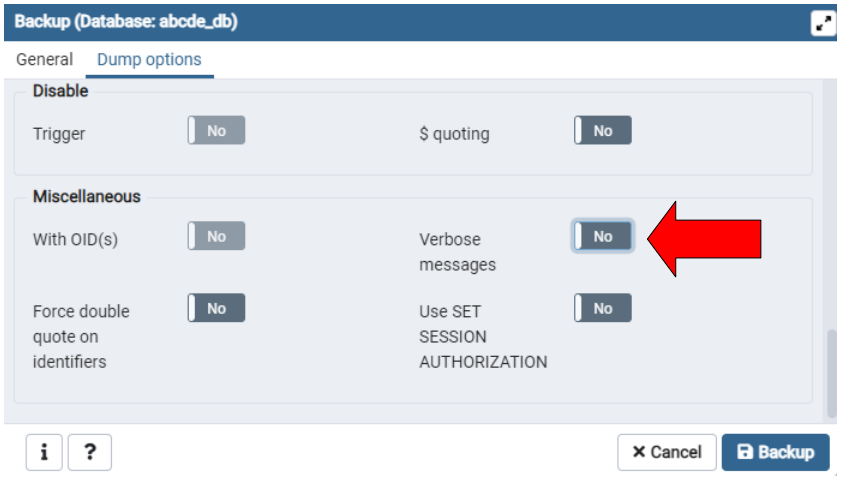
Formato texto



Exportar um banco PostGreSQL







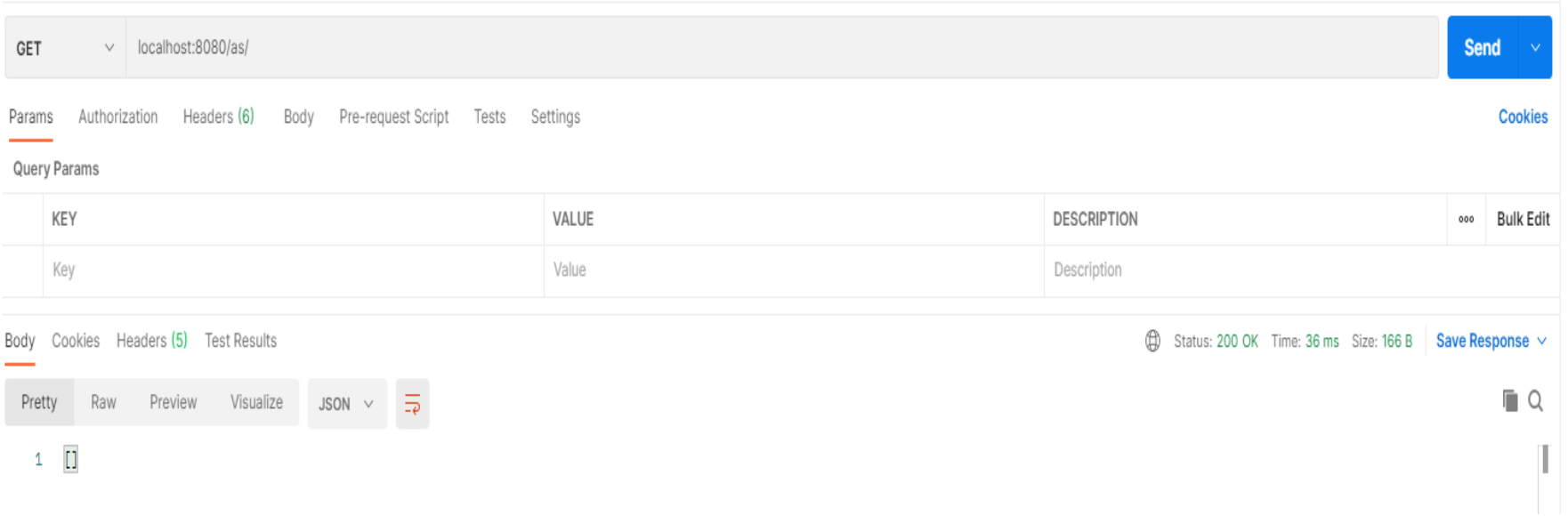
abcde_db.sql

- Podemos apagar as linhas que foram criadas antes do CREATE TABLE. Apagando também os comentários chegamos ao seguinte script:
- Com esse script criarmos o banco de dados em um serviço na web.

```
CREATE TABLE public.tb_a (
    id bigint NOT NULL,
    atributo1 character varying(255),
    atributo2 double precision,
    atributo3 timestamp without time zone,
    atributo4 integer
);

ALTER TABLE public.tb_a ALTER COLUMN id ADD GENERATED BY
DEFAULT AS IDENTITY (
    SEQUENCE NAME public.tb_a_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


ALTER TABLE ONLY public.tb_a
    ADD CONSTRAINT tb_a_pkey PRIMARY KEY (id);
```


Importante

- Em outras aulas criamos a classe **SemeadoraDadosTeste**. É importante notar que essa classe só será executada no perfil de teste (**@Profile("test")**).
- Dessa forma se solicitarmos a busca dos objetos do tipo “A” no banco PostGreSQL iremos obter como resultado uma consulta vazia.



Dúvidas?