

COMPUTACION EN LA NUBE CON HEROKU

(SERVICIOS WEB, JAVA, POSTGRESQL)

Autor: Msc. Juan Antonio Castro Silva

Versión: 1.4 01JUN2017-06:49

Requisitos

Para este tutorial se debe crear una cuenta en heroku, instalar el cliente de heroku, git, eclipse, jboss tools y postgresql.

Clonar la aplicación base de github

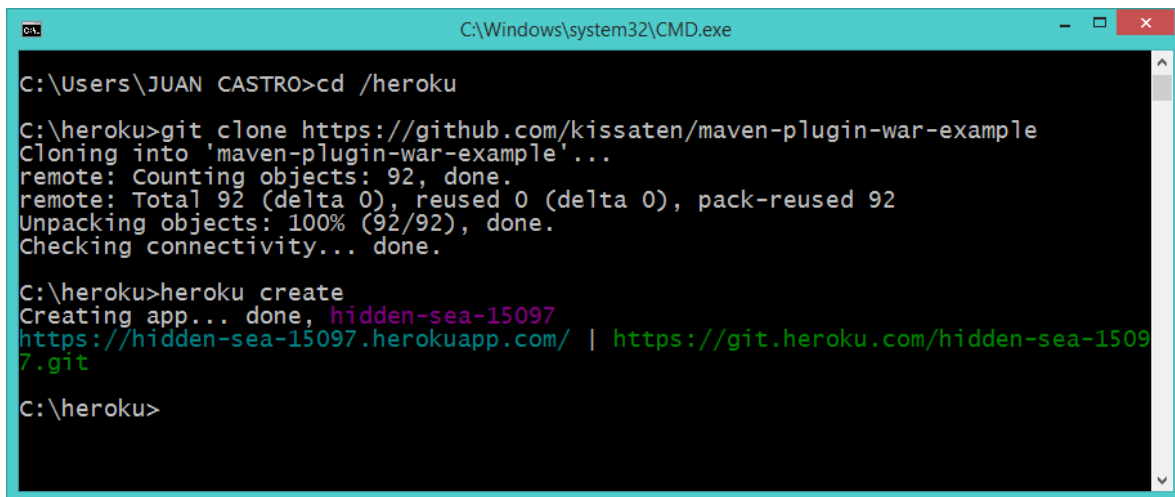
Se debe clonar la aplicación base de github.

```
git clone https://github.com/kissaten/maven-plugin-war-example
```

Crear la aplicación en Heroku

Para crear la aplicación se utilizará el cliente de heroku

```
heroku create
```



```
C:\Windows\system32\CMD.exe

C:\Users\JUAN CASTRO>cd /heroku

C:\heroku>git clone https://github.com/kissaten/maven-plugin-war-example
Cloning into 'maven-plugin-war-example'...
remote: Counting objects: 92, done.
remote: Total 92 (delta 0), reused 0 (delta 0), pack-reused 92
Unpacking objects: 100% (92/92), done.
Checking connectivity... done.

C:\heroku>heroku create
Creating app... done, hidden-sea-15097
https://hidden-sea-15097.herokuapp.com/ | https://git.heroku.com/hidden-sea-15097.git

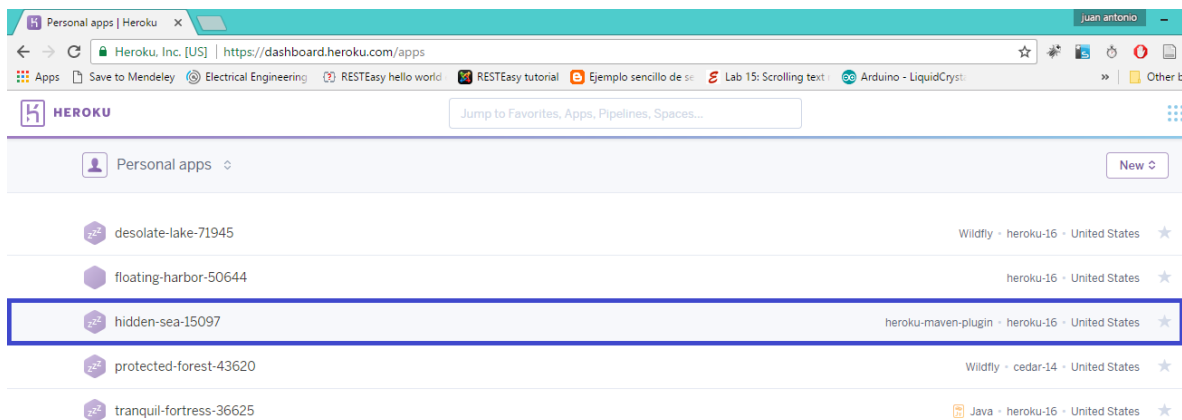
C:\heroku>
```

La consola retorna el nombre y la url de la aplicación.

```
hidden-sea-15097
https:// hidden-sea-15097.herokuapp.com
```

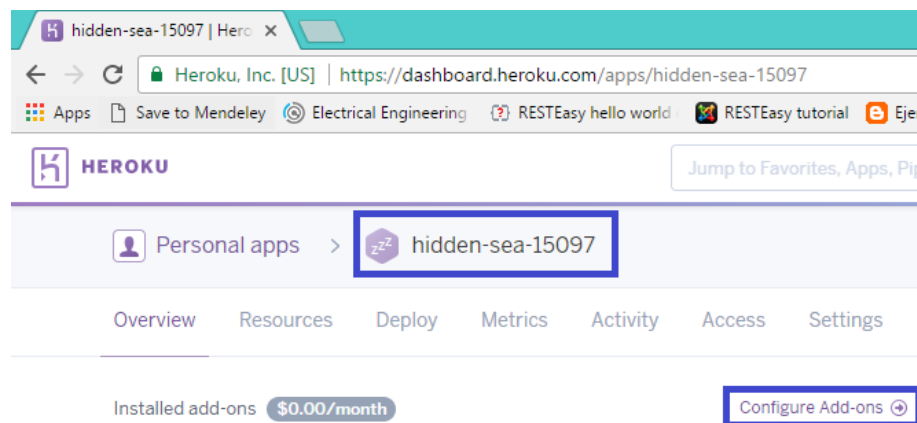
[Nota: El nombre de la aplicacion cambiará debido a que es aleatorio.]

Para verificar la creación de la aplicación abra la página de heroku, ingrese al sistema con sus credenciales (usuario, clave), en la sección de [Personal apps] puede observar la aplicación creada [hidden-sea-15097].

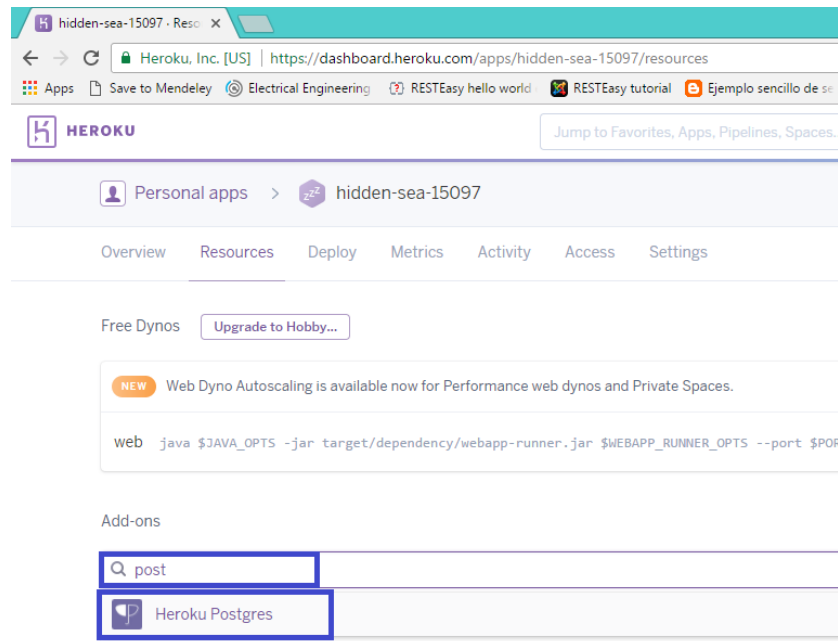


ADICIONAR LA BASE DE DATOS (postgresql)

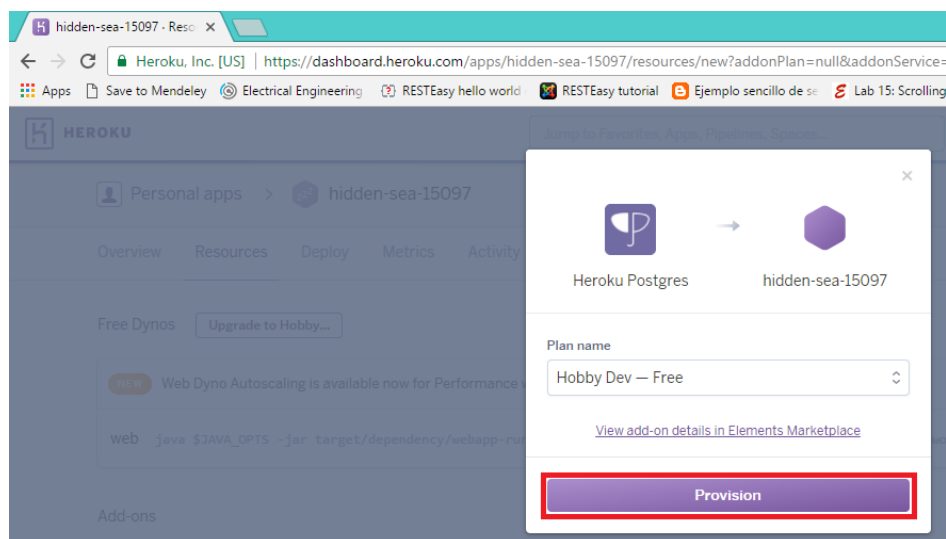
Para adicionar la base de datos postgresql, haga click en la aplicación y luego haga click en el boton [Configure Add-ons].



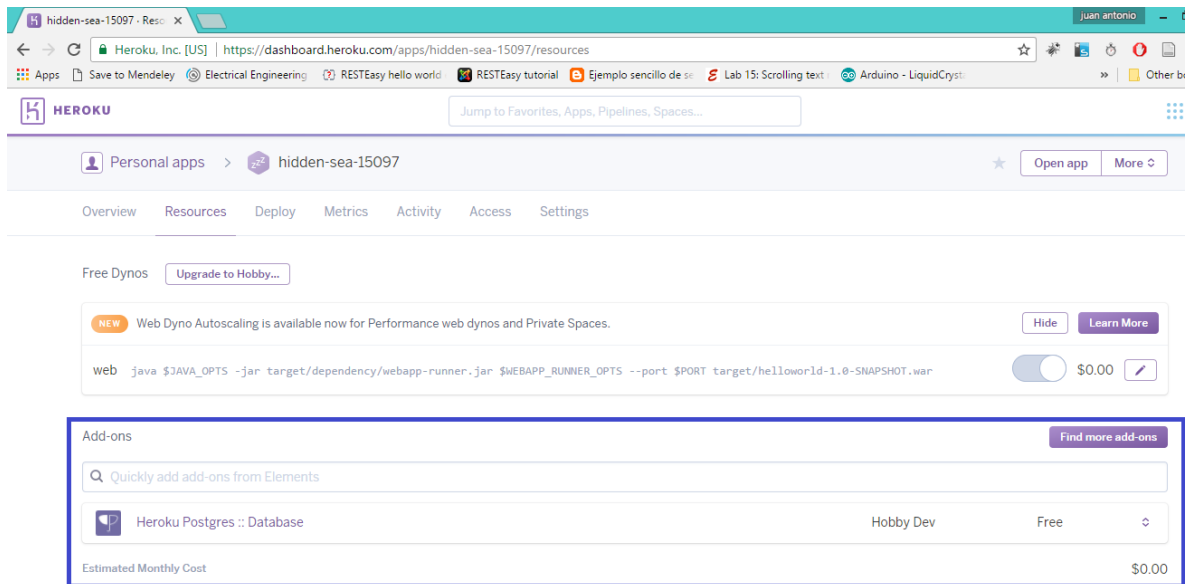
En la sección de [Add-ons] digite post y aparecerá la opción [Heroku Postgres], haga click en este Item.



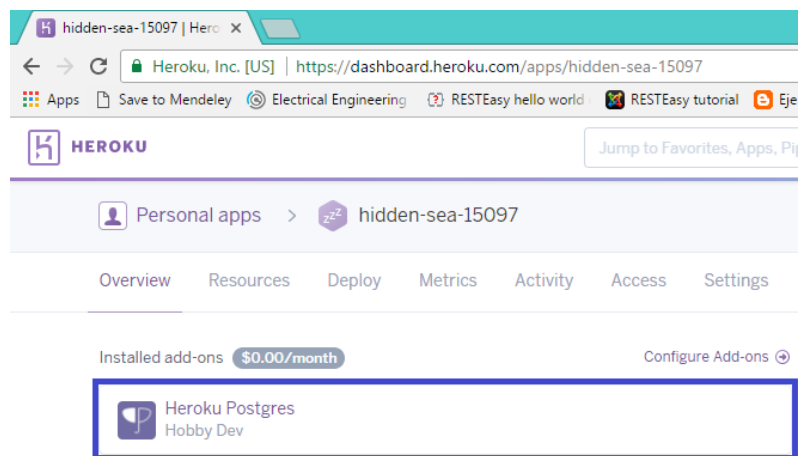
Seleccione el plan gratuito [Hobby Dev - Free] y haga click en el boton [Provision].



Ahora puede observar en la sección de [Add-ons] que está incluido [Heroku Postgres - Database].



Para obtener las credenciales de la base de datos haga click sobre ella [Heroku Postgres].



Haga click en el boton [View Credentials].

Heroku Data

Databases > postgresql-spherical-39931

SERVICE heroku-postgresql PLAN hobby-dev BILLING APP hidden-sea-15097

HEALTH

Available

PRIMARY Yes VERSION 9.6.1 CREATED 21 hours ago MAINTENANCE Unsupported ROLLBACK Unsupported

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

View Credentials...

Guarde estas credenciales.

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

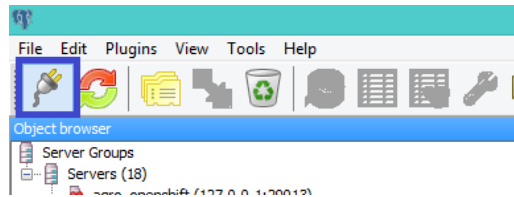
Please note that these credentials are not permanent.

Heroku rotates credentials periodically and updates applications where this database is attached.

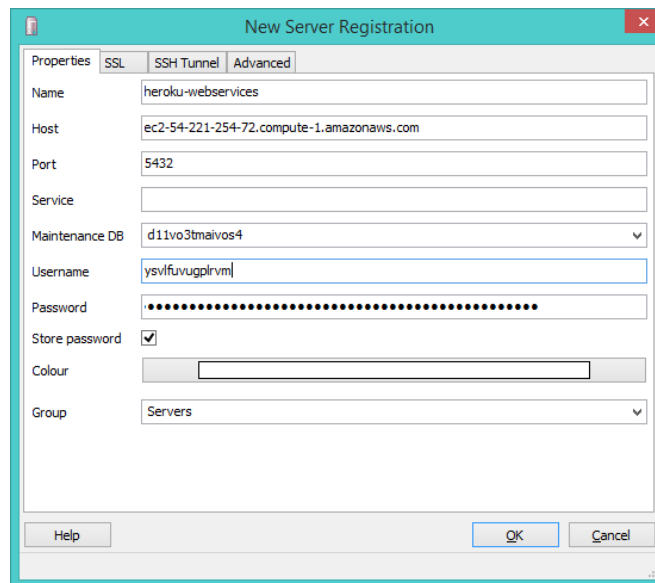
Host	ec2-54-221-254-72.compute-1.amazonaws.com
Database	d11vo3tmaivos4
User	ysvlfuvugplrv
Port	5432
Password	1717f32d692f8d8e827b0f942d156b2c1c70c857cfdb203ee01a9bfdbbaa500
URI	postgres://ysvlfuvugplrv:1717f32d692f8d8e827b0f942d156b2c1c70c857cfdb203ee01a9bfdbbaa500@ec2-54-221-254-72.compute-1.amazonaws.com:5432/d11vo3tmaivos4
Heroku CLI	heroku pg:psql postgresql-spherical-39931 --app hidden-sea-15097

Acceso a la base de datos desde pgAdmin.

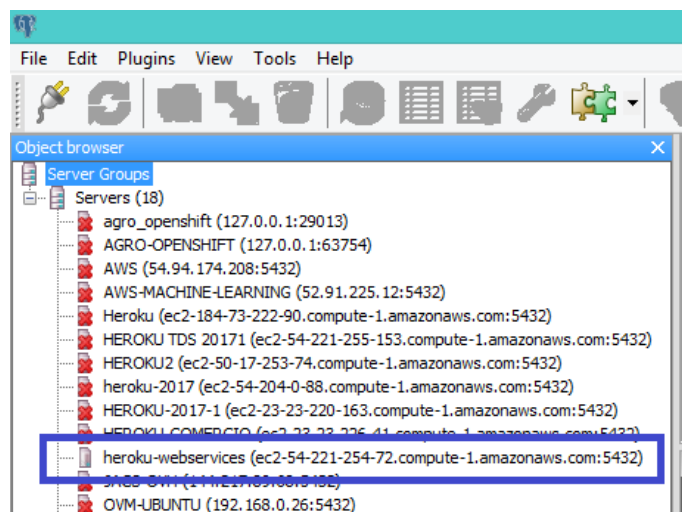
Haga click en el boton [Add a connection to a server].



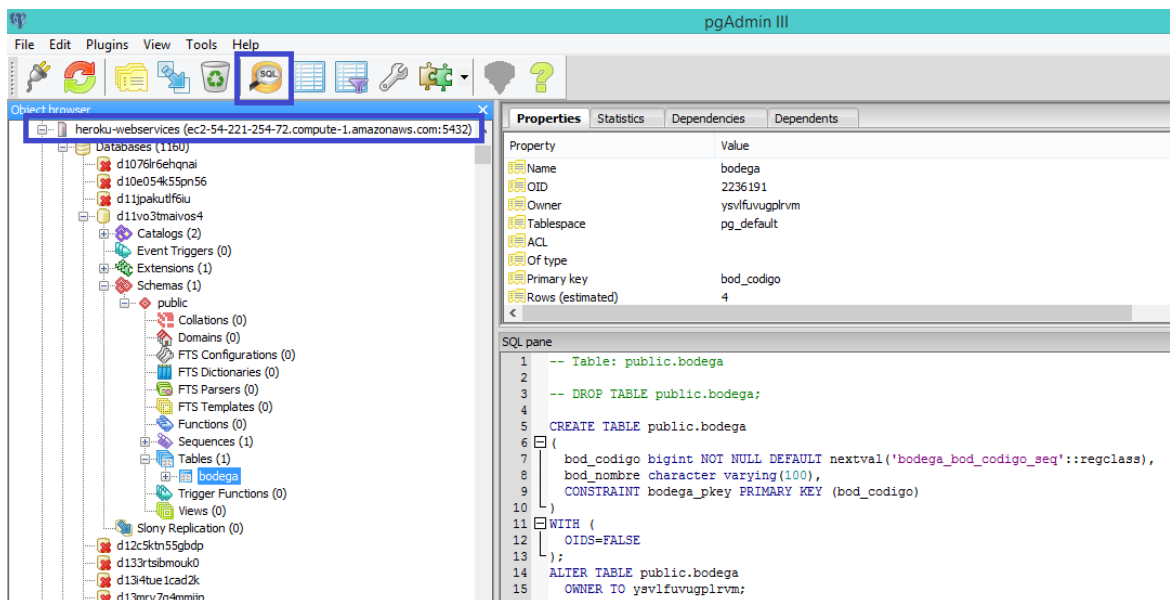
Digite los datos de las credenciales en la ventana [New Server Registration] y finalmente haga click en el boton [OK].



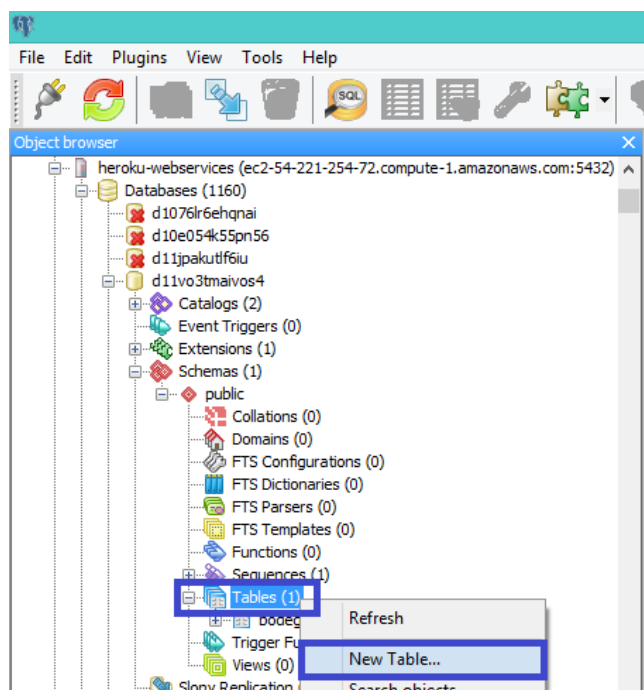
En la sección de servidores podrá ver el nuevo servidor.



Haga doble click en el nuevo servidor para su gestión (administración).



Para crear tablas lo puede hacer por este entorno gráfico, haga click con el boton derecho del mouse sobre la carpeta [Tables] y seleccione la opción de menú [New Table...].



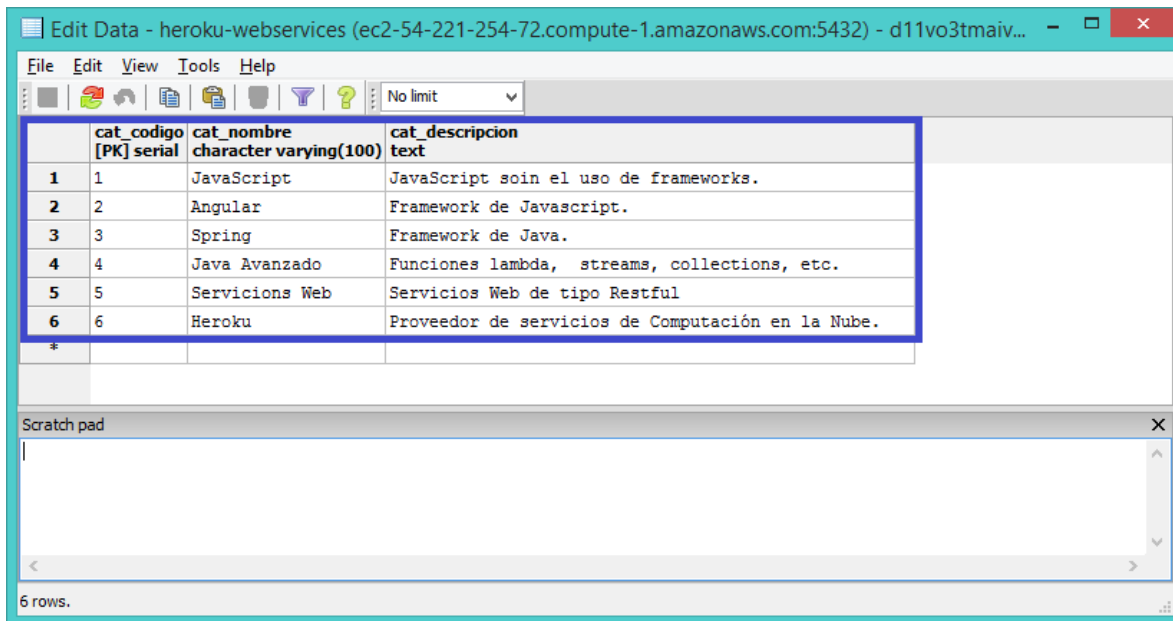
En la ventana [New Table], digite la información requerida (Name, Owner), adicione los campos [Columns] y cree las restricciones (llave primaria) [Constraints].

The 'New Table...' dialog box shows the 'Columns' tab. The 'Name' field is empty. The 'Owner' dropdown is set to 'public'. The 'Schema' dropdown is set to 'public'. The 'Comment' field is empty. The 'Use Slony' checkbox is unchecked. The 'OK' and 'Cancel' buttons are at the bottom right. A status bar at the bottom says 'Please specify name.'

También puede crear las tablas en la ventana de [Query], digite los comandos SQL y haga click arriba en el boton [Ejecutar].

The 'Query' window shows the 'SQL Editor' tab. The SQL code is:
1 CREATE TABLE categoria
2 (
3 cat_codigo serial,
4 cat_nombre character varying(100) NOT NULL,
5 cat_descripcion text,
6 CONSTRAINT categoria_pkey PRIMARY KEY (cat_codigo)
7)
8 WITH (
9 OIDS=FALSE
10);
11 ALTER TABLE categoria
12 OWNER TO ysvlfuvugplrv;
The 'Execute' button (a green play icon) is highlighted. The 'Output pane' at the bottom shows the message: 'Query returned successfully with no result in 106 msec.'

Puede adicionar los registros por la ventana de edición de datos [Edit Data].



Edit Data - heroku-webservices (ec2-54-221-254-72.compute-1.amazonaws.com:5432) - d11vo3tmaiv...

File Edit View Tools Help

No limit

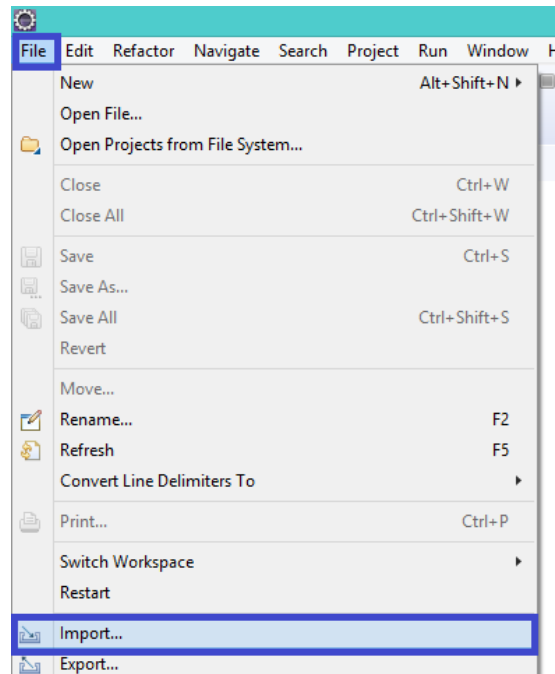
	cat_codigo [PK] serial	cat_nombre character varying(100)	cat_descripcion text
1	1	JavaScript	JavaScript soín el uso de frameworks.
2	2	Angular	Framework de Javascript.
3	3	Spring	Framework de Java.
4	4	Java Avanzado	Funciones lambda, streams, collections, etc.
5	5	Servicions Web	Servicios Web de tipo Restful
6	6	Heroku	Proveedor de servicios de Computación en la Nube.
*			

Scratch pad

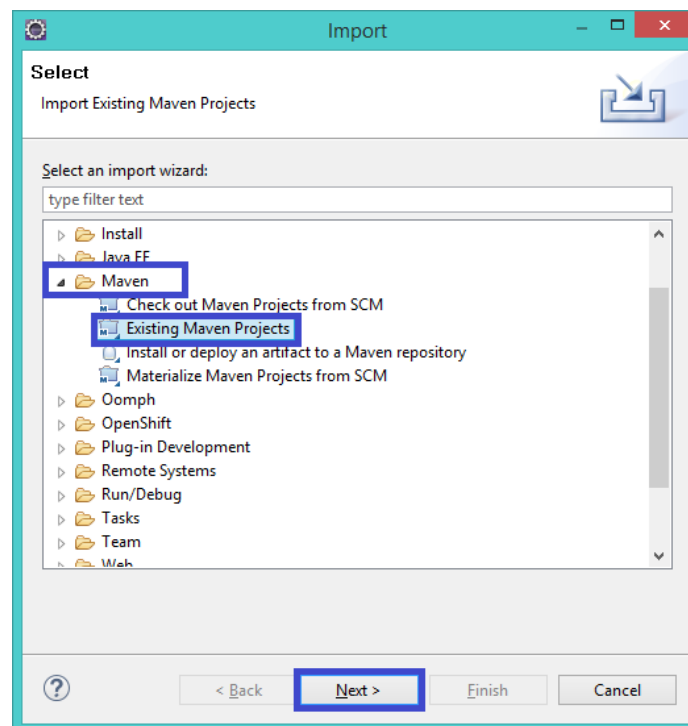
6 rows.

CREAR LA APLICACIÓN EN ECLIPSE

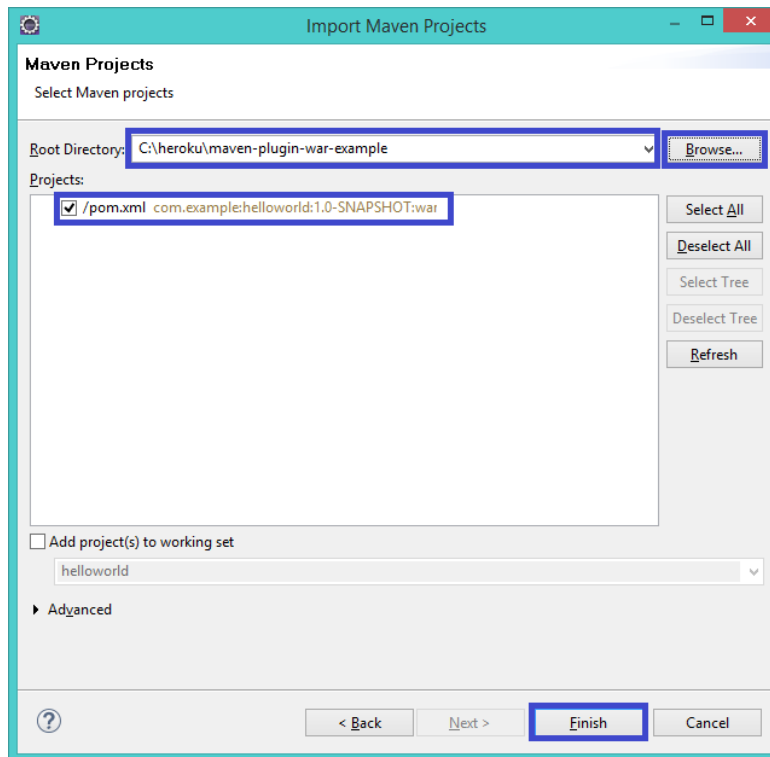
Importe la aplicación al IDE eclipse, haga click en el menú [File] y seleccione la opción [Import...].



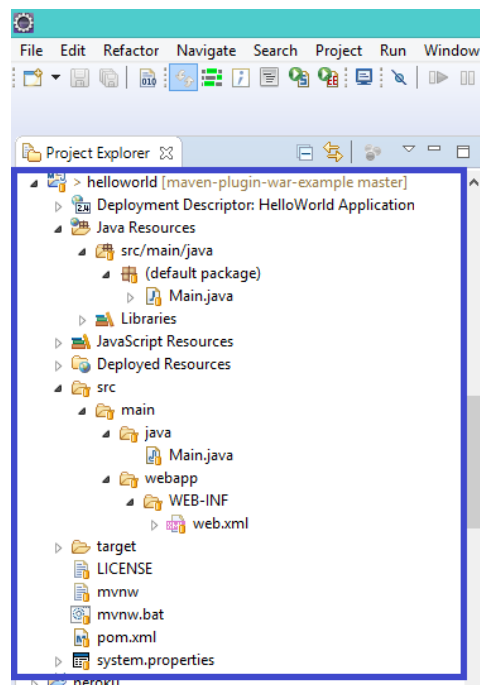
En la carpeta [Maven] seleccione la opción [Existing Maven Projects]



Haga click en el boton [Browse] y seleccione la carpeta donde está el clone del proyecto base de github y luego haga click en el boton [Finish].



En la sección [Project Explorer] podrá ver el proyecto importado.



Abra el archivo pom.xml (Project Object Model) y adicione el nombre de la aplicación [appName] al plugin [heroku-maven-plugin].

```
<appName>hidden-sea-15097</appName>
```

```
030     <build>
031       <plugins>
032         <plugin>
033           <groupId>com.heroku.sdk</groupId>
034           <artifactId>heroku-maven-plugin</artifactId>
035           <version>1.1.3</version>
036           <configuration>
037             <appName>hidden-sea-15097</appName>
038           </configuration>
039           <includes>
040             <include>target/dependency/newrelic-agent.jar</include>
041           </includes>
042         </plugin>
043       </plugins>
044     </build>
045   </project>
```

Busque las dependencias en la página del repositorio de Maven, por ejemplo el driver de postgresql.

The screenshot shows the Maven Repository search results for 'postgresql'. The search bar contains 'postgresql' and the 'Search' button is highlighted. The results show 'Found 276 results' and the first result is '1. PostgreSQL JDBC Driver JDBC 4.2' by 'org.postgresql'. The description is 'Java JDBC 4.2 (JRE 8+) driver for PostgreSQL database' and the last release is 'May 5, 2017'. There are 960 usages. The license is 'POSTGRESQL' and 'BSD'.

Seleccione el driver [PostgreSQL JDBC Driver JDBC 4.2].

The screenshot shows the Maven Repository artifact page for 'PostgreSQL JDBC Driver JDBC 4.2'. The breadcrumb is 'Home > org.postgresql > postgresql'. The artifact is 'PostgreSQL JDBC Driver JDBC 4.2' by 'org.postgresql'. The description is 'Java JDBC 4.2 (JRE 8+) driver for PostgreSQL database'. The license is 'BSD 2-clause' and 'POSTGRESQL'. The categories are 'PostgreSQL Drivers'. The tags are 'database', 'postgresql', and 'driver'. It is used by 960 artifacts. The version table shows the following data:

Version	Repository	Usages	Date
42.1.1	Central	39	(May, 2017)
42.1.1	PostgreSQL	0	(May, 2017)

Copie la dependencia y péguela en el archivo [pom.xml] en la sección <dependencies>.

The screenshot shows the Maven Repository website for the PostgreSQL JDBC Driver 42.1.1. The page includes a search bar at the top, a sidebar with 'Indexed Artifacts (6.48M)' and 'Popular Categories', and a main content area with a table of metadata and a code snippet for the dependency.

Indexed Artifacts (6.48M)

6470k
3235k
0
2004 2017

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection

PostgreSQL JDBC Driver JDBC 4.2 » 42.1.1
Java JDBC 4.2 (JRE 8+) driver for PostgreSQL database

License	BSD 2-clause POSTGRESQL
Categories	PostgreSQL Drivers
Organization	PostgreSQL Global Development Group
HomePage	https://github.com/pgjdbc/pgjdbc
Date	(May 05, 2017)
Files	Download (BUNDLE) (760 KB)
Repositories	Central Sonatype Releases
Used By	960 artifacts

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.1.1</version>
</dependency>
```

☒ Include comment with link to declaration

Copied to clipboard!

Esta dependencia se debe adicionar al nodo <dependencies>.

```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.1.1</version>
</dependency>
```

Archivo pom.xml final con todas las dependencias incluidas.

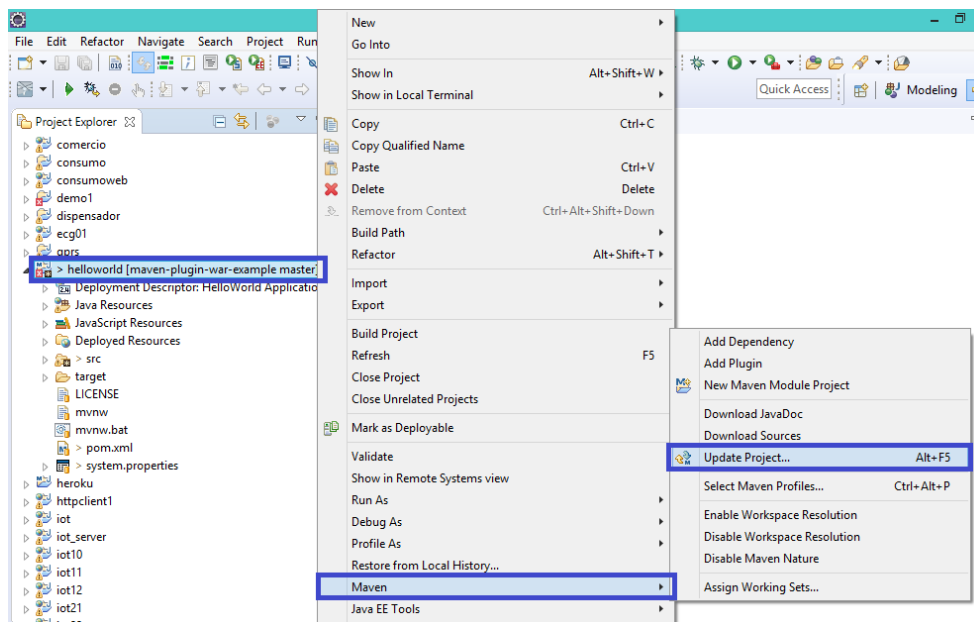
```
001 <?xml version="1.0" encoding="UTF-8"?>
002 <project xmlns="http://maven.apache.org/POM/4.0.0"
003 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
004 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
005 http://maven.apache.org/maven-v4_0_0.xsd">
006 <modelVersion>4.0.0</modelVersion>
007 <groupId>com.example</groupId>
008 <version>1.0-SNAPSHOT</version>
009 <artifactId>helloworld</artifactId>
010 <packaging>war</packaging>
011
012 <dependencies>
013 <dependency>
014 <groupId>org.eclipse.jetty</groupId>
015 <artifactId>jetty-servlet</artifactId>
016 <version>9.2.11.v20150529</version>
017 </dependency>
018 <dependency>
019 <groupId>javax.servlet</groupId>
020 <artifactId>javax.servlet-api</artifactId>
021 <version>3.1.0</version>
022 </dependency>
023 <dependency>
024 <groupId>com.newrelic.agent.java</groupId>
025 <artifactId>newrelic-agent</artifactId>
026 <version>3.18.0</version>
027 <scope>provided</scope>
028 </dependency>
029 <dependency>
030 <groupId>org.jboss.resteasy</groupId>
031 <artifactId>resteasy-jaxrs</artifactId>
032 <version>3.0.13.Final</version>
033 </dependency>
034 <dependency>
035 <groupId>org.jboss.resteasy</groupId>
036 <artifactId>resteasy-jaxb-provider</artifactId>
037 <version>3.0.13.Final</version>
038 </dependency>
039 <dependency>
040 <groupId>org.jboss.resteasy</groupId>
041 <artifactId>resteasy-jettison-provider</artifactId>
042 <version>3.0.13.Final</version>
043 </dependency>
044 <dependency>
045 <groupId>org.jboss.resteasy</groupId>
046 <artifactId>resteasy-jsapi</artifactId>
047 <version>3.0.13.Final</version>
048 </dependency>
049 <dependency>
050 <groupId>org.jboss.resteasy</groupId>
051 <artifactId>resteasy-jackson2-provider</artifactId>
052 <version>3.0.13.Final</version>
053 </dependency>
054 <dependency>
055 <groupId>org.jboss.resteasy</groupId>
056 <artifactId>resteasy-servlet-initializer</artifactId>
057 <version>3.0.13.Final</version>
058 </dependency>
059
```

```

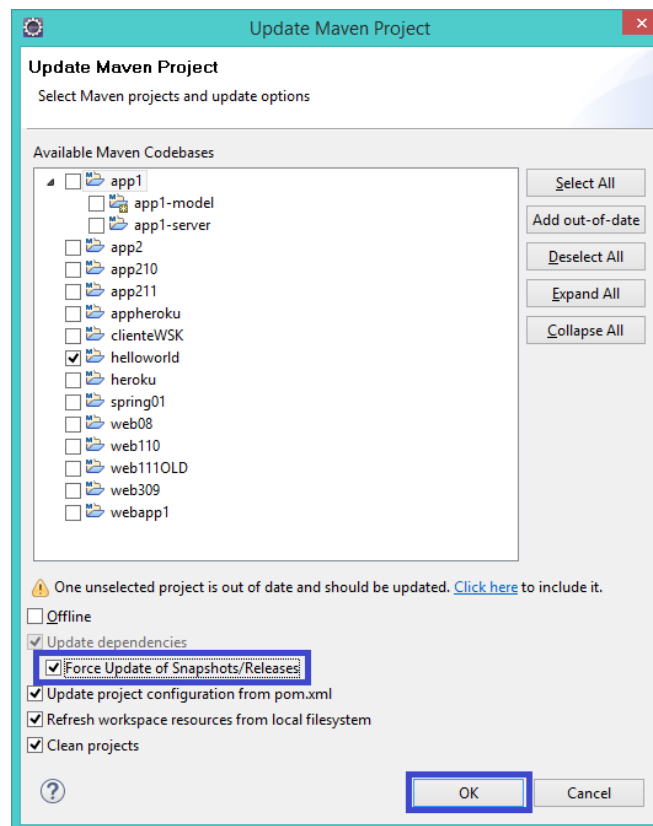
060 <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
061 <dependency>
062   <groupId>org.postgresql</groupId>
063   <artifactId>postgresql</artifactId>
064   <version>42.1.1</version>
065 </dependency>
066 </dependencies>
067 <build>
068   <plugins>
069     <plugin>
070       <groupId>com.heroku.sdk</groupId>
071       <artifactId>heroku-maven-plugin</artifactId>
072       <version>1.1.3</version>
073       <configuration>
074         <appName>hidden-sea-15097</appName>
075         <includes>
076           <include>target/dependency/newrelic-agent.jar</include>
077         </includes>
078       </configuration>
079     </plugin>
080     <plugin>
081       <groupId>org.apache.maven.plugins</groupId>
082       <artifactId>maven-dependency-plugin</artifactId>
083       <version>2.6</version>
084       <executions>
085         <execution>
086           <id>copy-new-relic</id>
087           <phase>package</phase>
088           <goals>
089             <goal>copy-dependencies</goal>
090           </goals>
091           <configuration>
092             <includeGroupIds>com.newrelic.agent.java</includeGroupIds>
093             <includeArtifactIds>newrelic-agent</includeArtifactIds>
094             <stripVersion>true</stripVersion>
095           </configuration>
096         </execution>
097       </executions>
098     </plugin>
099   </plugins>
100 </build>
101 </project>

```

Para actualizar las dependencias de Maven, haga click con el boton derecho del mouse sobre el Proyecto, seleccione la opción [Maven] y finalmente haga click en la opción [Update Project...].

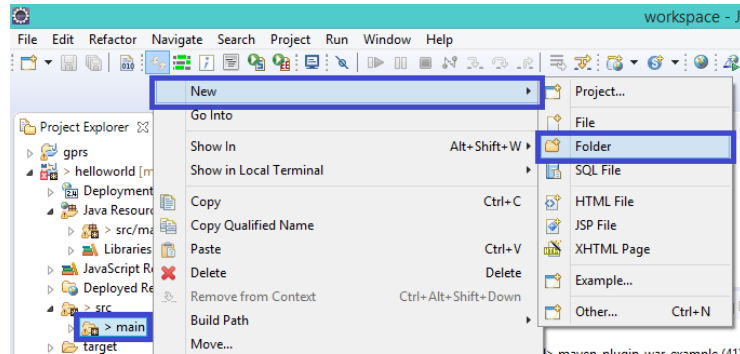


Seleccione la opción [Force Update of Snapshots Releases] y haga click en el boton [OK].

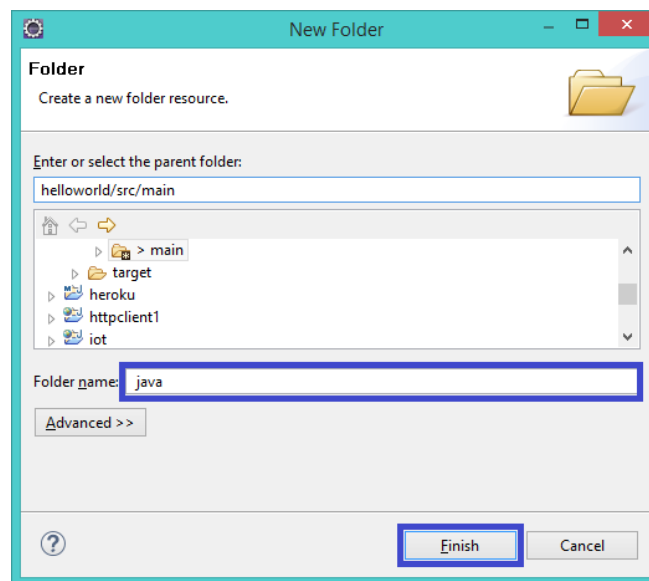


Creación de paquetes y clases.

Debe crear la carpeta [java] para almacenar los paquetes y clases, haga click con el boton derecho del mouse sobre el folder [src/main], seleccione [New] y finalmente haga click en [Folder].



En la ventana [New Folder] digite el nombre de la carpeta [java] y finalmente haga click en el boton [Finish].



Cree los siguientes paquetes:

- org.software.bodega
- org.software.servicio
- org.software.utilidad

Cree las siguientes clases dentro del paquete org.software.bodega:

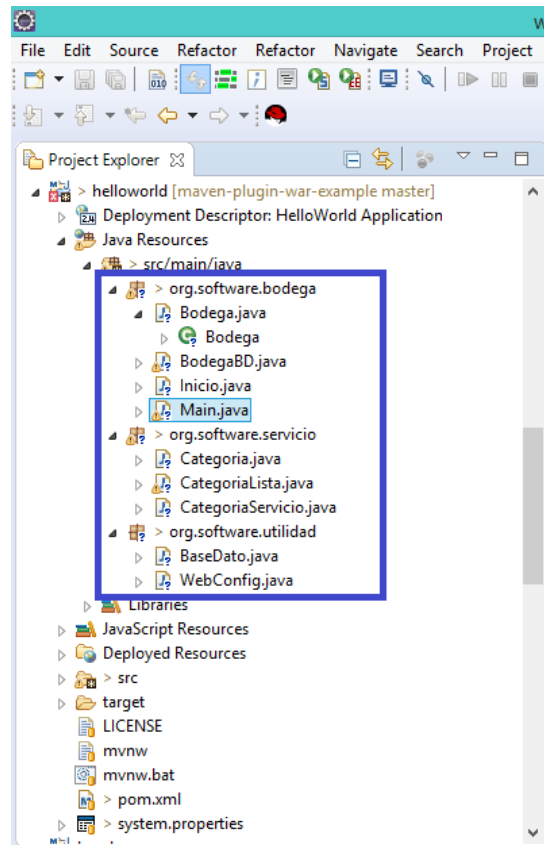
- Bodega.java
- BodegaBD.java
- Inicio.java
- Main.java

Cree las siguientes clases dentro del paquete org.software.servicio:

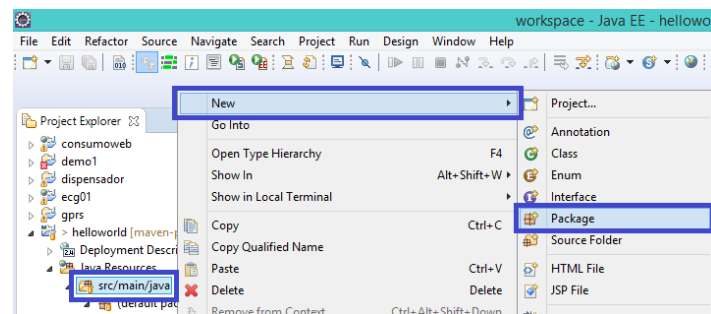
- Categoria.java
- CategoriaLista.java
- CategoriaServicio.java

Cree las siguientes clases dentro del paquete org.software.utilidad:

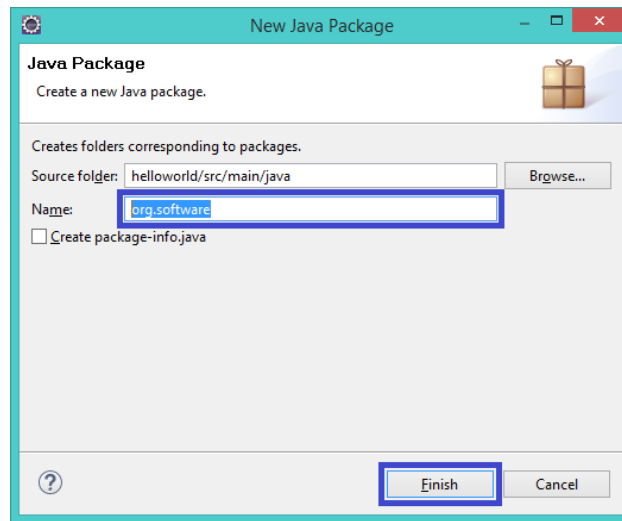
- BaseDato.java
- WebConfig.java



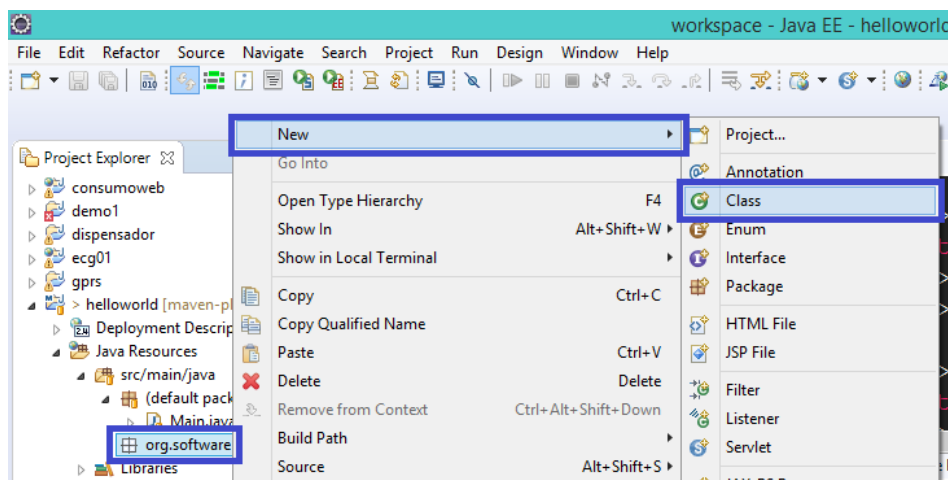
Para crear un paquete haga click con el boton derecho del mouse sobre la carpeta [src/main/java], seleccione la opción [New] y finalmente haga click en [Package].



En la ventana [New Java Package], digite el nombre del paquete y haga click en el boton [Finish].



Para crear una clase haga click sobre el paquete con el boton derecho del mouse, seleccione la opción [New] y haga click en [Class].



En la ventana [New Java Class], digite el nombre de la clase y haga click en el boton [Finish].

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Archivo [Bodega.java].

```
001 package org.software.bodega;
002
003 public class Bodega {
004
005     int codigo;
006     String nombre;
007
008     public Bodega() {
009         super();
010     }
011     public Bodega(int codigo, String nombre) {
012         super();
013         this.codigo = codigo;
014         this.nombre = nombre;
015     }
016
017     public int getCodigo() {
018         return codigo;
019     }
020     public void setCodigo(int codigo) {
021         this.codigo = codigo;
022     }
023     public String getNombre() {
024         return nombre;
025     }
026     public void setNombre(String nombre) {
027         this.nombre = nombre;
028     }
029
030     @Override
031     public String toString() {
032         return "Bodega [codigo=" + codigo + ", nombre=" + nombre + "];"
033     }
034
035 }
```

Archivo [BodegaBD.java].

```
001 package org.software.bodega;
002
003 import java.io.IOException;
004 import java.io.PrintWriter;
005 import java.sql.Connection;
006 import java.sql.ResultSet;
006 import java.sql.Statement;
007 import java.util.ArrayList;
008
009 import org.software.utilidad.BaseDato;
010
011 public class BodegaBD {
012
013     public void crearTablas() {
014         BaseDato basedato = new BaseDato();
015
016         Connection conexion1 = null;
017         Statement sentencia1 = null;
018         ResultSet rs1 = null;
019         String sql = "";
020         try {
021             conexion1 = basedato.getConexion();
022             sentencia1 = conexion1.createStatement();
023
024             sql = "create table public.bodega ("
025                 + " bod_codigo bigserial,"
026                 + " bod_nombre character varying (100),"
027                 + " CONSTRAINT bodega_pkey PRIMARY KEY (bod_codigo))"
028                 + " WITH ( OIDS=FALSE );"
029                 + " ALTER TABLE public.bodega OWNER TO ysvlfuvugplrvm";
030
031             int creada = sentencia1.executeUpdate(sql);
032
033             sql = "insert into bodega (bod_nombre)"
034                 + " values ('Neiva - Central'),"
035                 + "('Pitalito'), ('Garzon'), ('La Plata')";
036             int insertados = sentencia1.executeUpdate(sql);
037
038         }
039         catch (Exception e) {
040             System.out.println("Error: " + e.toString());
041         }
042         finally{
043             basedato.cerrar(rs1);
044             basedato.cerrar(sentencia1);
045             basedato.cerrar(conexion1);
046         }
047     }
048
049     public ArrayList<Bodega> getBodegas() {
050         ArrayList <Bodega>bodegas = new ArrayList<Bodega>();
051
052         BaseDato basedato = new BaseDato();
053
054         Connection conexion1 = null;
055         Statement sentencia1 = null;
056         ResultSet rs1 = null;
057         String sql = "";
058     }
```

```

059         try {
060             conexion1 = basedato.getConexion();
061             sentencia1 = conexion1.createStatement();
062
063             sql = "select * from bodega";
064
065             rs1 = sentencia1.executeQuery(sql);
066             while(rs1.next()){
067                 int codigo = rs1.getInt("bod_codigo");
068                 String nombre = rs1.getString("bod_nombre");
069
070                 Bodega bodega = new Bodega();
071                 bodega.setCodigo(codigo);
072                 bodega.setNombre(nombre);
073                 bodegas.add(bodega);
074             }
075         }
076     }
077     catch (Exception e) {
078         System.out.println("Error: " + e.toString());
079     }
080     finally{
081         basedato.cerrar(rs1);
082         basedato.cerrar(sentencia1);
083         basedato.cerrar(conexion1);
084     }
085
086     return bodegas;
087 }
088 }

```

Archivo [Inicio.java] (Cree un Servlet).

```
001 package org.software.bodega;
002
003 import java.io.IOException;
004 import javax.servlet.ServletException;
005 import javax.servlet.http.HttpServlet;
006 import javax.servlet.http.HttpServletRequest;
007 import javax.servlet.http.HttpServletResponse;
008
009 /**
010  * Servlet implementation class Inicio
011  */
012 public class Inicio extends HttpServlet {
013     private static final long serialVersionUID = 1L;
014
015     /**
016      * @see HttpServlet#HttpServlet()
017      */
018     public Inicio() {
019         super();
020     }
021
022     /**
023      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
024      response)
025      */
026     protected void doGet(HttpServletRequest request, HttpServletResponse
027 response) throws ServletException, IOException {
028         BodegaBD bodegaBD = new BodegaBD();
029         bodegaBD.crearTablas();
030     }
031
032 }
```


Archivo [Main.java] (Cree un Servlet)

```
001 package org.software.bodega;
002 import java.io.IOException;
003 import java.io.PrintWriter;
004
005 import javax.servlet.ServletException;
006 import javax.servlet.http.*;
007
008 import java.util.ArrayList;
009
010 public class Main extends HttpServlet {
011     @Override
012     protected void doGet(HttpServletRequest request, HttpServletResponse
013 response)
014         throws ServletException, IOException {
015
016         PrintWriter out = response.getWriter();
017         try {
018
019             BodegaBD bodegaBD = new BodegaBD();
020
021             ArrayList<Bodega> bodegas = bodegaBD.getBodegas();
022
023             for (Bodega bodega : bodegas) {
024                 out.println("<p>" + bodega + "</p>");
025             }
026         }
027         catch (Exception e) {
028             out.println("Error: " + e.toString());
029         }
030     }
031 }
```

Archivo [BaseDato.java].

```
001 package org.software.utilidad;
002
003 import java.net.URI;
004 import java.net.URISyntaxException;
005 import java.sql.Connection;
006 import java.sql.DriverManager;
007 import java.sql.PreparedStatement;
008 import java.sql.ResultSet;
009 import java.sql.SQLException;
010 import java.sql.Statement;
011
012 public class BaseDato {
013
014     public Connection getConexion() throws URISyntaxException, SQLException {
015         URI dbUri = new URI(System.getenv("DATABASE_URL"));
016
017         String username = dbUri.getUserInfo().split(":")[0];
018         String password = dbUri.getUserInfo().split(":")[1];
019         String dbUrl = "jdbc:postgresql://" + dbUri.getHost() + ':' +
020 dbUri.getPort() + dbUri.getPath();
021
022         return DriverManager.getConnection(dbUrl, username, password);
023     }
024
025     public void cerrar(Object objeto) {
026         if(objeto instanceof ResultSet){
027             try {
028                 ((ResultSet) objeto).close();
029             } catch (SQLException e) {
030                 e.printStackTrace();
031             }
032         }
033
034         if(objeto instanceof Statement){
035             try {
036                 ((Statement) objeto).close();
037             } catch (SQLException e) {
038                 e.printStackTrace();
039             }
040         }
041
042         if(objeto instanceof PreparedStatement){
043             try {
044                 ((PreparedStatement) objeto).close();
045             } catch (SQLException e) {
046                 e.printStackTrace();
047             }
048         }
049
050         if(objeto instanceof Connection){
051             try {
052                 ((Connection) objeto).close();
053             } catch (SQLException e) {
054                 e.printStackTrace();
055             }
056         }
057     }
058
059 }
```

Archivo [WebConfig.java].

```
001 package org.software.utilidad;
002
003 import javax.ws.rs.ApplicationPath;
004 import javax.ws.rs.core.Application;
005
006 @ApplicationPath("/servicios")
007 public class WebConfig extends Application {
008
009     // No methods defined inside
010
011 }
```

Archivo [Categoria.java].

```
001 package org.software.servicio;
002
003 public class Categoria {
004     int codigo;
005     String nombre;
006     String descripcion;
007
008     public Categoria() {
009         super();
010         // TODO Auto-generated constructor stub
011     }
012     public Categoria(int codigo, String nombre, String descripcion) {
013         super();
014         this.codigo = codigo;
015         this.nombre = nombre;
016         this.descripcion = descripcion;
017     }
018
019     public int getCodigo() {
020         return codigo;
021     }
022     public void setCodigo(int codigo) {
023         this.codigo = codigo;
024     }
025     public String getNombre() {
026         return nombre;
027     }
028     public void setNombre(String nombre) {
029         this.nombre = nombre;
030     }
031     public String getDescripcion() {
032         return descripcion;
033     }
034     public void setDescripcion(String descripcion) {
035         this.descripcion = descripcion;
036     }
037
038     @Override
039     public String toString() {
040         return "Categoria [codigo=" + codigo + ", nombre=" + nombre + ",
041 descripcion=" + descripcion + "];"
042     }
043
044 }
```

Archivo [Categorialista.java].

```
001 package org.software.servicio;
002
003 import java.util.List;
004 import javax.xml.bind.annotation.XmlElement;
005 import javax.xml.bind.annotation.XmlRootElement;
006
007 @XmlRootElement(name = "listing")
008 public class Categorialista {
009     private List<Categoria> items;
010
011     public Categorialista() {
012     }
013
014     public Categorialista(List<Categoria> items) {
015         this.items = items;
016     }
017
018     @XmlElement(name = "data")
019     public List<Categoria> getItems() {
020         return items;
021     }
022 }
```

Archivo [CategoriaServicio.java].

```
001 package org.software.servicio;
002
003 import java.sql.Connection;
004 import java.sql.PreparedStatement;
005 import java.sql.ResultSet;
006 import java.sql.Statement;
007 import java.util.ArrayList;
008 import javax.ws.rs.Consumes;
009 import javax.ws.rs.GET;
010 import javax.ws.rs.POST;
011 import javax.ws.rs.PUT;
012 import javax.ws.rs.DELETE;
013 import javax.ws.rs.Path;
014 import javax.ws.rs.PathParam;
015 import javax.ws.rs.Produces;
016 import javax.ws.rs.core.Response;
017 import org.software.utilidad.BaseDato;
018
019 @Path("/categoria")
020 public class CategoriaServicio {
021
022     @POST
023     @Path("/")
024     @Consumes({ "application/json" })
025     @Produces("application/json")
026     public Response adicionar(Categoria categoria) {
027         BaseDato basedato = new BaseDato();
028         Connection conexion1 = null;
029         PreparedStatement sentenciaPreparada1 = null;
030         String sql = "";
031         String mensaje = "";
032         int insertados = 0;
033         try {
034             conexion1 = basedato.getConexion();
035             sql = "INSERT INTO categoria (cat_nombre, cat_descripcion)";
036             sql = sql + " VALUES (?,?)";
037             sentenciaPreparada1 = conexion1.prepareStatement(sql);
038             sentenciaPreparada1.setString(1, categoria.getNombre());
039             sentenciaPreparada1.setString(2, categoria.getDescripcion());
040             insertados = sentenciaPreparada1.executeUpdate();
041         }
042         catch (Exception e) {
043             System.out.println("Error: " + e.toString());
044         }
045         finally{
046             basedato.cerrar(sentenciaPreparada1);
047             basedato.cerrar(conexion1);
048         }
049
050         if (insertados > 0) {
051             mensaje = "{\"mensaje\":\"Adicionar OK\"}";
052             return Response.status(200).entity(mensaje).build();
053         }
054         else {
055             mensaje = "{\"mensaje\":\"Error al adicionar\"}";
056             return Response.status(400).entity(mensaje).build();
057         }
058     }
059 }
```

```

060     @PUT
061     @Path("/{codigo}")
062     @Consumes({ "application/json" })
063     @Produces("application/json")
064     public Response modificar(Categoria categoria,
065         @PathParam(value = "codigo") int codigo) {
066         BaseDato basedato = new BaseDato();
067         Connection conexion1 = null;
068         PreparedStatement sentenciaPreparada1 = null;
069         String sql = "";
070         String mensaje = "";
071         int modificados = 0;
072
073         try {
074             conexion1 = basedato.getConexion();
075             sql = "UPDATE categoria set cat_nombre=?, cat_descripcion=?";
076             sql = sql + " WHERE cat_codigo=?";
077             sentenciaPreparada1 = conexion1.prepareStatement(sql);
078             sentenciaPreparada1.setString(1, categoria.getNombre());
079             sentenciaPreparada1.setString(2, categoria.getDescripcion());
080             sentenciaPreparada1.setInt(3, codigo);
081             modificados = sentenciaPreparada1.executeUpdate();
082         }
083         catch (Exception e) {
084             System.out.println("Error: " + e.toString());
085         }
086         finally{
087             basedato.cerrar(sentenciaPreparada1);
088             basedato.cerrar(conexion1);
089         }
090
091         if (modificados > 0) {
092             mensaje = "{\"mensaje\":\"Modificar OK\"}";
093             return Response.status(200).entity(mensaje).build();
094         }
095         else {
096             mensaje = "{\"mensaje\":\"Error al modificar\"}";
097             return Response.status(400).entity(mensaje).build();
098         }
099     }
100
101     @DELETE
102     @Path("/{codigo}")
103     @Consumes({ "application/json" })
104     @Produces("application/json")
105     public Response adicionar(@PathParam(value = "codigo") int codigo) {
106         BaseDato basedato = new BaseDato();
107         Connection conexion1 = null;
108         PreparedStatement sentenciaPreparada1 = null;
109         String sql = "";
110         String mensaje = "";
111         int eliminados = 0;
112
113         try {
114             conexion1 = basedato.getConexion();
115             sql = "DELETE FROM categoria WHERE cat_codigo=?";
116             sentenciaPreparada1 = conexion1.prepareStatement(sql);
117             sentenciaPreparada1.setInt(1, codigo);
118             eliminados = sentenciaPreparada1.executeUpdate();
119         }
120         catch (Exception e) {

```

```

121         System.out.println("Error: " + e.toString());
122     }
123     finally{
124         basedato.cerrar(sentenciaPreparada1);
125         basedato.cerrar(conexion1);
126     }
127
128     if (eliminados > 0) {
129         mensaje = "{\"mensaje\":\"Eliminar OK\"}";
130         return Response.status(200).entity(mensaje).build();
131     }
132     else {
133         mensaje = "{\"mensaje\":\"Error al eliminar\"}";
134         return Response.status(400).entity(mensaje).build();
135     }
136 }
137
138 @GET
139 @Path("/")
140 @Produces("application/json")
141 public CategoriaLista getCategorias() {
142     ArrayList<Categoria> lista = new ArrayList<Categoria>();
143     BaseDato basedato = new BaseDato();
144     Connection conexion1 = null;
145     Statement sentencia1 = null;
146     ResultSet rs1 = null;
147     String sql = "";
148     try {
149         conexion1 = basedato.getConexion();
150         sentencia1 = conexion1.createStatement();
151         sql = "select * from categoria";
152         rs1 = sentencia1.executeQuery(sql);
153         while (rs1.next()) {
154             int codigo = rs1.getInt("cat_codigo");
155             String nombre = rs1.getString("cat_nombre");
156             String descripcion = rs1.getString("cat_descripcion");
157             Categoria categoria = new Categoria();
158             categoria.setCodigo(codigo);
159             categoria.setNombre(nombre);
160             categoria.setDescripcion(descripcion);
161             lista.add(categoria);
162         }
163     }
164     catch (Exception e) {
165         System.out.println("Error: " + e.toString());
166     }
167     finally{
168         basedato.cerrar(rs1);
169         basedato.cerrar(sentencia1);
170         basedato.cerrar(conexion1);
171     }
172
173     return new CategoriaLista(lista);
174 }
175 }

```

CREAR LA INTERFACE GRAFICA DE USUARIO (GUI)

En los proyectos de tipo Maven el equivalente a la carpeta [WebContent] en un proyecto Web Dinámico es la carpeta [src/main/webapp].

Cree dentro de esa carpeta los siguientes archivos.

Archivo [css/estilo.css].

```
001  label, input {
002      display: block;
003  }
004  input.text {
005      margin-bottom: 12px;
006      width: 95%;
007      padding: .4em;
008  }
009  fieldset {
010      padding: 0;
011      border: 0;
012      margin-top: 25px;
013  }
014  h1 {
015      font-size: 2.2em;
016      margin: .6em 0;
017  }
```

Archivo [js/categoria.js].

```
001  $(function() {
002      var dialog, form;
003      var table = $('#example').DataTable({
004          // "ajax" : "servicios/categoria/",
005          "ajax" : {
006              "url": "servicios/categoria/",
007              "dataSrc": "listing.data"
008          },
009          "columns" : [
010              { "data" : "codigo" },
011              { "data" : "nombre" },
012              { "data" : "descripcion" }
013          ] });
014
015      $('#example tbody').on('click', 'tr', function() {
016
017          if ($(this).hasClass('selected')) {
018              $(this).removeClass('selected');
019          }
020          else {
021              table.$('tr.selected').removeClass('selected');
022              $(this).addClass('selected');
023          }
024      });
025
026
027
028
```



```

029     function editar(accion) {
030         var codigo = "0"; var nombre = "";
031         var descripcion = "";
032         if (accion != "Adicionar") {
033             codigo = table.row('.selected').data().codigo;
034             nombre = table.row('.selected').data().nombre;
035             descripcion = table.row('.selected').data().descripcion;
036         }
037         document.getElementById("accion").value = accion;
038         document.getElementById("codigo").value = codigo;
039         document.getElementById("nombre").value = nombre;
040         document.getElementById("descripcion").value = descripcion;
041         dialog.dialog("open");
042         $("#boton").html('<span class="ui-button-text">' + accion + '</span>');
043     }
044
045     function ejecutar() {
046         var accion = document.getElementById("accion").value;
047
048         if (accion == "Adicionar") {
049             adicionarCategoria();
050         }
051         if (accion == "Modificar") {
052             modificarCategoria();
053         }
054         if (accion == "Eliminar") {
055             eliminarCategoria();
056         }
057     }
058
059     function adicionarCategoria() {
060         // Collect input from html page
061         var nombre = document.getElementById("nombre").value;
062         var descripcion = document.getElementById("descripcion").value;
063
064         var r = new REST.Request();
065         r.setURI("https://hidden-sea-15097.herokuapp.com/servicios/categoria/");
066         r.setMethod("POST"); r.setContentType("application/json");
067         r.setEntity({ nombre : nombre, descripcion : descripcion });
068         r.execute(function(status, request, entity) {
069             mostrarRespuesta(entity);
070         });
071     }
072
073     function modificarCategoria() {
074         // Collect input from html page
075         var codigo = document.getElementById("codigo").value;
076         var nombre = document.getElementById("nombre").value;
077         var descripcion = document.getElementById("descripcion").value;
078
079         var r = new REST.Request();
080         r.setURI("https://hidden-sea-15097.herokuapp.com/servicios/categoria/" +
081         codigo);
082         r.setMethod("PUT"); r.setContentType("application/json");
083         r.setEntity({ nombre : nombre, descripcion : descripcion });
084         r.execute(function(status, request, entity) {
085             mostrarRespuesta(entity);
086         });
087     }
088
089

```

```

090     function eliminarCategoria() {
091         var codigo = document.getElementById("codigo").value;
092         var r = new REST.Request();
093         r.setURI("https://hidden-sea-15097.herokuapp.com/servicios/categoria/" +
094     codigo);
095         r.setMethod("DELETE");
096         r.execute(function(status, request, entity) {
097             mostrarRespuesta(entity);
098         });
099     }
100
101     function mostrarRespuesta(entity){
102         table.ajax.reload();
103         dialog.dialog("close");
104         document.getElementById("dialogo-mensaje").innerHTML = "<p>" +
105     entity.mensaje + "</p>";
106         $("#dialogo-mensaje").dialog({
107             modal : true,
108             buttons : {
109                 Ok : function() {
110                     $(this).dialog("close");
111                 }
112             }
113         });
114     }
115
116     dialog = $("#dialog-form").dialog({
117         autoOpen : false,
118         height : 360,
119         width : 640,
120         modal : true,
121         buttons : {
122             "Ejecutar": {
123                 id: 'boton',
124                 text: 'Ejecutar',
125                 click: function () {
126                     ejecutar();
127                 }
128             },
129             Cancel : function() {
130                 dialog.dialog("close");
131             }
132         },
133         close : function() {
134             form[0].reset();
135         }
136     });
137     form = dialog.find("form").on("submit", function(event) {
138         event.preventDefault();
139         ejecutar();
140     });
141     $("#adicionar").button().on("click", function() {
142         editar('Adicionar');
143     });
144     $("#modificar").button().on("click", function() {
145         editar('Modificar');
146     });
147     $("#eliminar").button().on("click", function() {
148         editar('Eliminar');
149     });
150 });

```

```

001 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
002 "http://www.w3.org/TR/html4/loose.dtd">
003 <html>
004 <head>
005 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
006 <title>Insert title here</title>
007 <!-- This will include the whole javascript file including ajax handling -->
008 <script lang="javascript" src="rest-api.js"></script>
009 <link rel="stylesheet"
010 href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
011 <script type="text/javascript"
012 src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
013 <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
014 <script type="text/javascript"
015 src="https://cdn.datatables.net/1.10.11/js/jquery.dataTables.min.js"></scr
016 ipt>
017 <link rel="stylesheet"
018 href="https://cdn.datatables.net/1.10.11/css/jquery.dataTables.min.css">
019 <link rel="stylesheet" href="css/estilo.css">
020 <script type="text/javascript" src="js/categoria.js"></script>
021 </head>
022 <body>
023
024 <h1>GESTION DE CATEGORIAS</h1>
025 <div style="float: right;">
026 <input type="button" value="Adicionar" id="adicionar" />
027 <input type="button" value="Modificar" id="modificar" />
028 <input type="button" value="Eliminar" id="eliminar" />
029 </div>
030 <table id="example" class="display" cellpadding="0" width="100%">
031 <thead>
032 <tr>
033 <th>Codigo</th>
034 <th>Nombre</th>
035 <th>Descripcion</th>
036 </tr>
037 </thead>
038 <tfoot>
039 <tr>
040 <th>Codigo</th>
041 <th>Nombre</th>
042 <th>Descripcion</th>
043 </tr>
044 </tfoot>
045 </table>
046 <div id="dialogo-mensaje" title="Gestión de Categorías"></div>
047 <div id="dialog-form" title="Forma Categoría">
048 <p class="validateTips">Todos los campos son requeridos.</p>
049 <form>
050 <fieldset>
051 <input type="hidden" name="accion" id="accion" value="" /> <input
052 type="hidden" name="codigo" id="codigo" value="" />
053 <label for="nombre">Nombre</label>
054 <input type="text" name="nombre" id="nombre" value=""
055 class="text ui-widget-content ui-corner-all" />
056 <label for="descripcion">Descripcion</label>
057
058 <input type="text" name="descripcion" id="descripcion" value=""
059 class="text ui-widget-content ui-corner-all" />

```

```
060      <!a Allow form submission with keyboard without duplicating The
061      dialog button -->
062      <input type="submit" tabindex="-1" style="position: absolute;
063      top: -1000px">
064      </fieldset>
065      </form>
066      </div>
067 </body>
068 </html>
```

ARCHIVOS DE CONFIGURACION

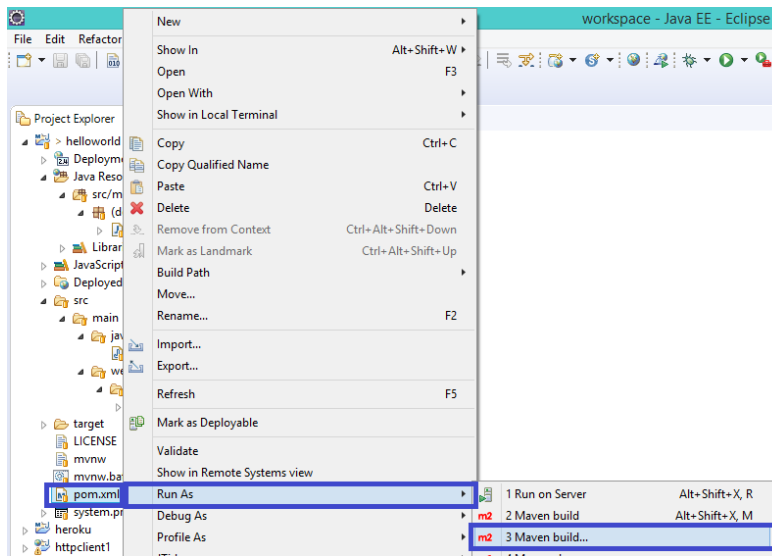
Dentro de la carpeta WEB-INF cree o modifique el archivo web.xml.

```
001 <?xml version="1.0" encoding="ISO-8859-1" ?>
002
003 <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
004 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
005 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
006 http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
007 version="2.4">
008
009 <display-name>HelloWorld Application</display-name>
010
011 <servlet>
012 <servlet-name>RESEasy-JSAPI</servlet-name>
013 <servlet-class>org.jboss.resteasy.jsapi.JSAPIServlet</servlet-class>
014 </servlet>
015 <servlet-mapping>
016 <servlet-name>RESEasy-JSAPI</servlet-name>
017 <url-pattern>/rest-api.js</url-pattern>
018 </servlet-mapping>
019
020 <servlet>
021 <servlet-name>MainServlet</servlet-name>
022 <servlet-class>org.software.bodega.Main</servlet-class>
023 </servlet>
024 <Servlet>
025 <servlet-name>Inicio</servlet-name>
026 <servlet-class>org.software.bodega.Inicio</servlet-class>
027 </servlet>
028
029 <servlet-mapping>
030 <servlet-name>MainServlet</servlet-name>
031 <url-pattern>/Main</url-pattern>
032 </servlet-mapping>
033 <servlet-mapping>
034 <servlet-name>Inicio</servlet-name>
035 <url-pattern>/Inicio</url-pattern>
036 </servlet-mapping>
037
038 </web-app>
```

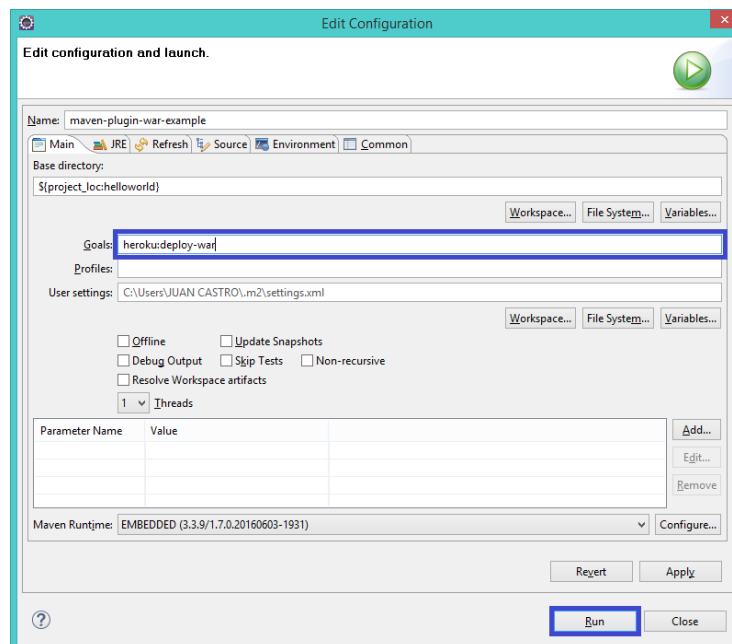
Archivo [system.properties].

```
001 java.runtime.version=1.8
```

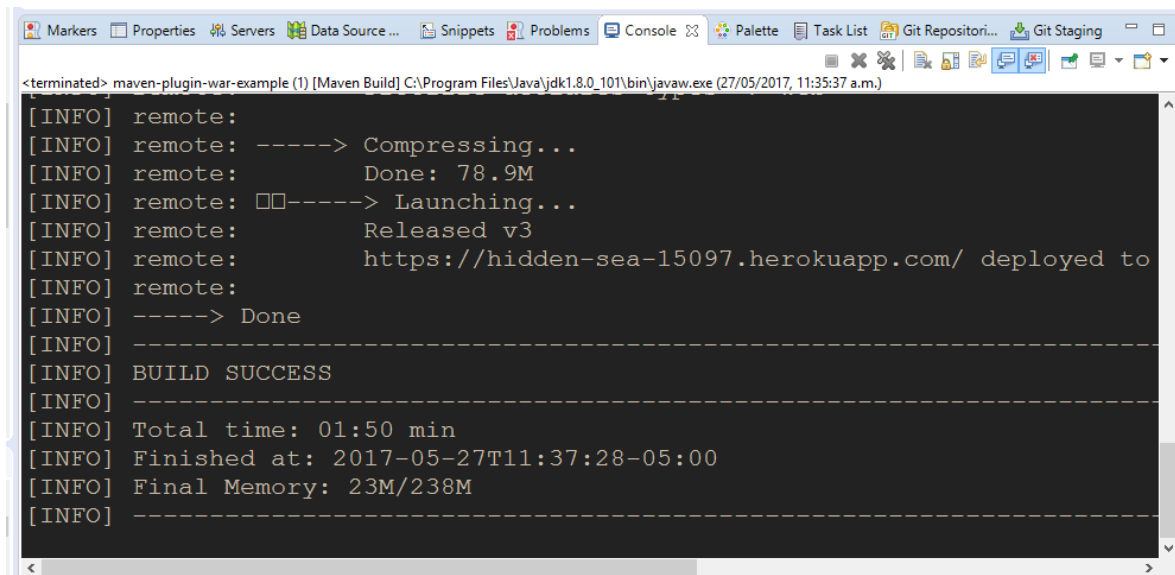
EJECUTAR LA APLICACIÓN



Digite heroku:deploy-war en el campo [Goals] y haga click en el boton [Run].



La consola muestra los resultados de la compilación, carga [upload] y publicación [deploy] de la aplicación en el servidor.



```
<terminated> maven-plugin-war-example (1) [Maven Build] C:\Program Files\Java\jdk1.8.0_101\bin\javaw.exe (27/05/2017, 11:35:37 a.m.)  
[INFO] remote:  
[INFO] remote: -----> Compressing...  
[INFO] remote: Done: 78.9M  
[INFO] remote: □□-----> Launching...  
[INFO] remote: Released v3  
[INFO] remote: https://hidden-sea-15097.herokuapp.com/ deployed to  
[INFO] remote:  
[INFO] -----> Done  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 01:50 min  
[INFO] Finished at: 2017-05-27T11:37:28-05:00  
[INFO] Final Memory: 23M/238M  
[INFO] -----
```

VER EL ARCHIVO DE LOGS DE LA APLICACIÓN

Para ver los archivos de log de la aplicación, digite en una consola:

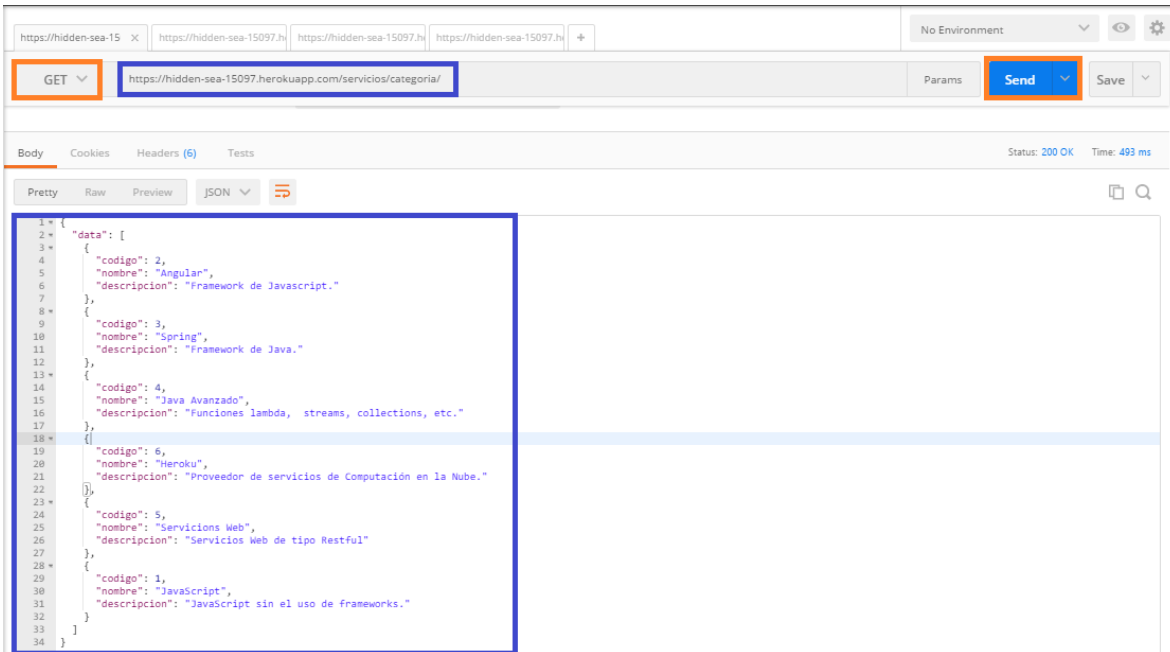
```
heroku logs -t
```

PRUEBA DE LOS SERVICIOS WEB

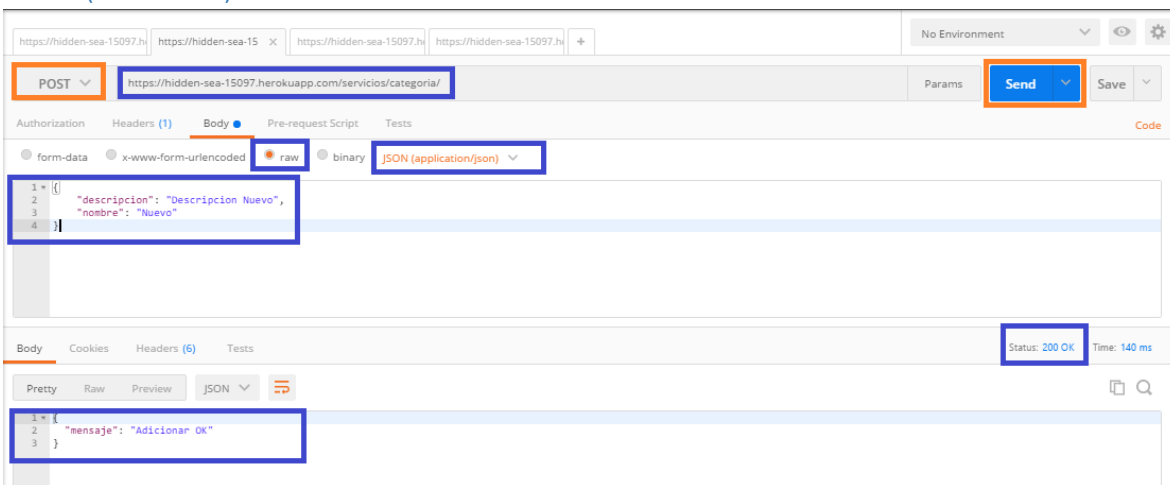
Para probar los servicios web se utilizará Postman, la url de la aplicación es:

<https://hidden-sea-15097.herokuapp.com/servicios/categoria/>

GET (listar)



POST (adicionar)



PUT (modificar)

The screenshot shows a REST client interface with the following details:

- Method:** PUT (highlighted with a blue box)
- URL:** `https://hidden-sea-15097.herokuapp.com/servicios/categoria/2` (highlighted with a blue box)
- Body Type:** raw (highlighted with a blue box)
- Content Type:** JSON (application/json) (highlighted with a blue box)
- Request Body:**

```
1 {
2   "descripcion": "Descripción Modificada",
3   "nombre": "Categoría Cambiada"
4 }
```

 (highlighted with a blue box)
- Status:** 200 OK (highlighted with a blue box)
- Time:** 476 ms
- Response Body:**

```
1 {
2   "mensaje": "Modificar OK"
3 }
```

 (highlighted with a blue box)

DELETE (eliminar)

The screenshot shows a REST client interface with the following details:

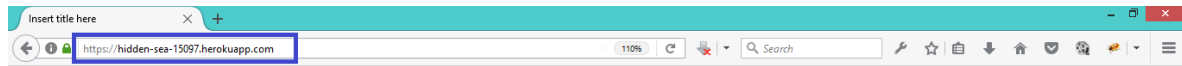
- Method:** DELETE (highlighted with a blue box)
- URL:** `https://hidden-sea-15097.herokuapp.com/servicios/categoria/6` (highlighted with a blue box)
- Authorization:** No Auth
- Status:** 200 OK (highlighted with a blue box)
- Time:** 169 ms
- Response Body:**

```
1 {
2   "mensaje": "Eliminar OK"
3 }
```

 (highlighted with a blue box)

PRUEBA DE LA APLICACIÓN

Para probar la aplicación web en heroku, abra un navegador (Browser) y digite la url del proyecto: <https://hidden-sea-15097.herokuapp.com>



GESTION DE CATEGORIAS

Adicionar Modificar Eliminar

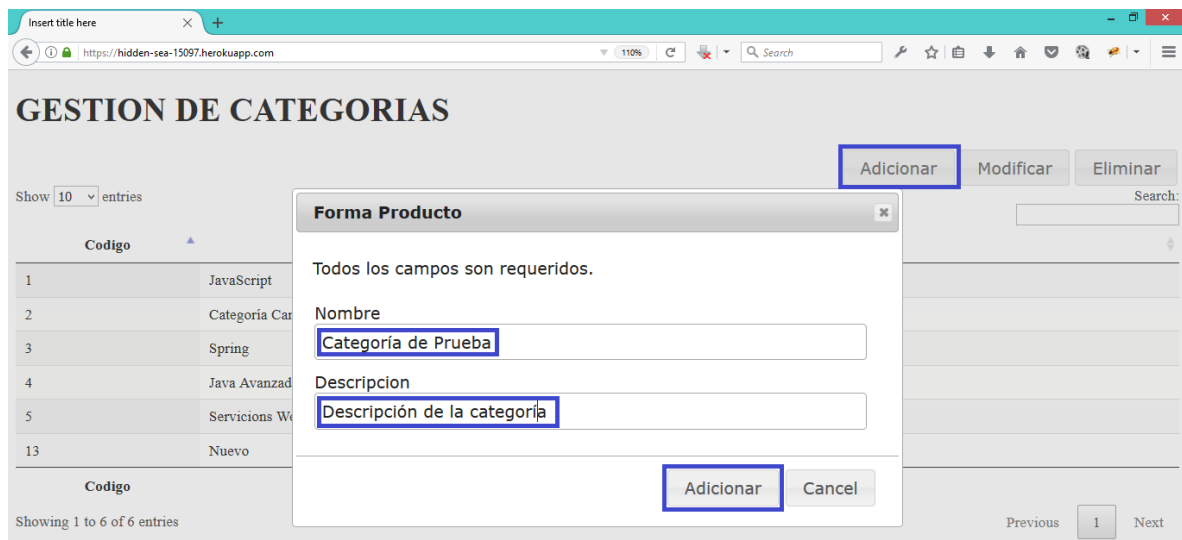
Show 10 entries Search:

Codigo	Nombre	Descripcion
1	JavaScript	JavaScript sin el uso de frameworks.
2	Categoria Cambiada	Descripción Modificada
3	Spring	Framework de Java.
4	Java Avanzado	Funciones lambda, streams, collections, etc.
5	Servicios Web	Servicios Web de tipo Restful
13	Nuevo	Descripcion Nuevo

Showing 1 to 6 of 6 entries Previous 1 Next

Adicionar

Para adicionar una categoría (registro) haga click arriba en el boton [Adicionar], digite la información (nombre, descripción) y finalmente haga click en el boton [Adicionar].



GESTION DE CATEGORIAS

Adicionar Modificar Eliminar

Show 10 entries Search:

Codigo	Nombre	Descripcion
1	JavaScript	JavaScript sin el uso de frameworks.
2	Categoria Cambiada	Descripción Modificada
3	Spring	Framework de Java.
4	Java Avanzado	Funciones lambda, streams, collections, etc.
5	Servicios Web	Servicios Web de tipo Restful
13	Nuevo	Descripcion Nuevo

Showing 1 to 6 of 6 entries Previous 1 Next

Forma Producto

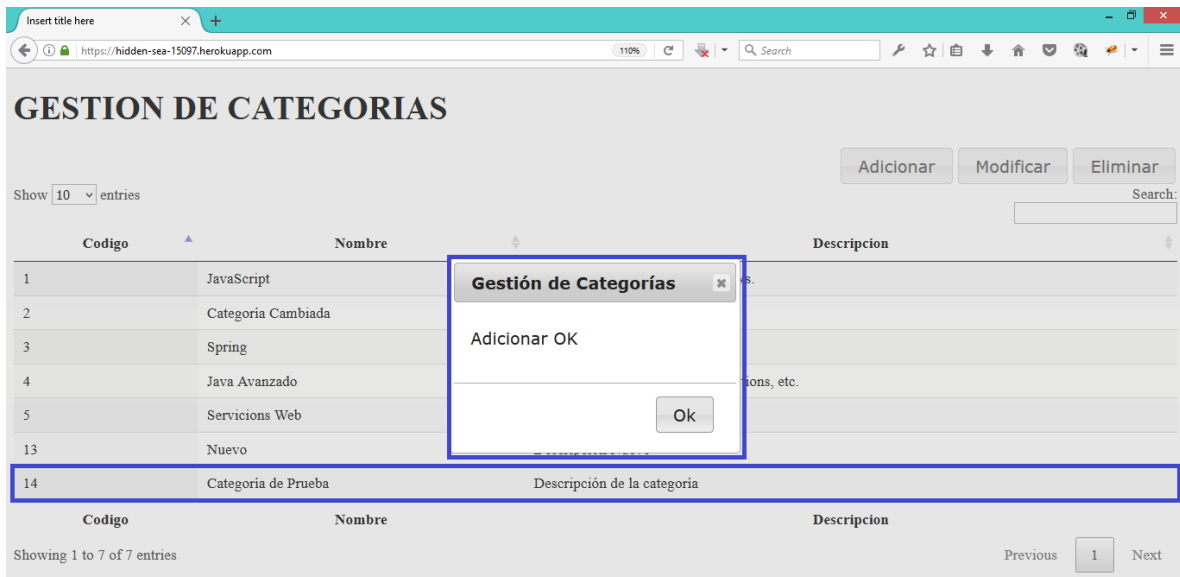
Todos los campos son requeridos.

Nombre
Categoría de Prueba

Descripcion
Descripción de la categoría

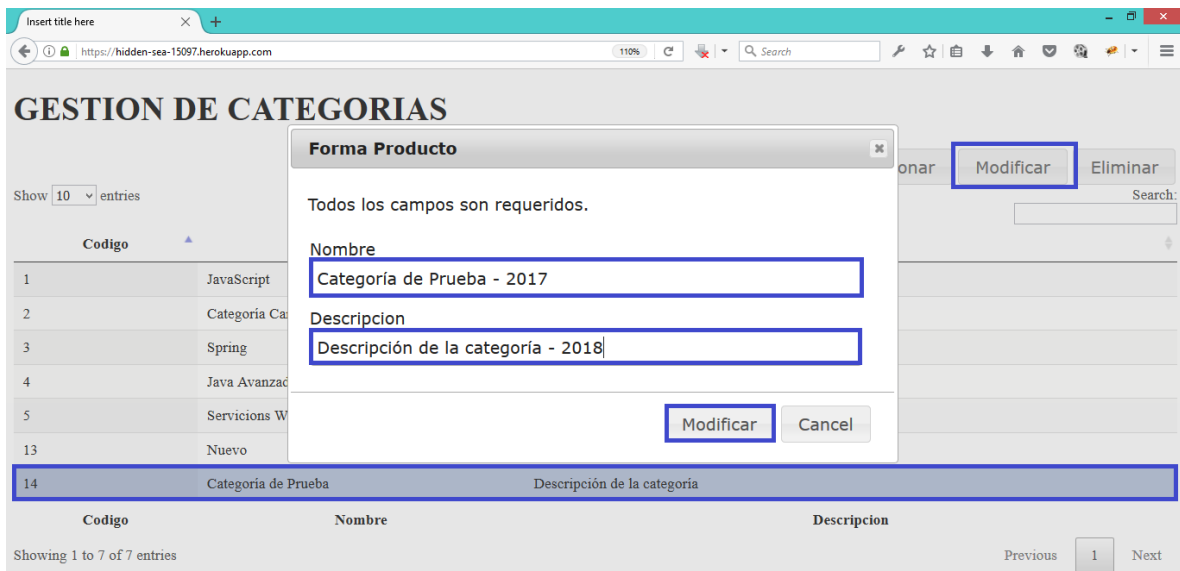
Adicionar Cancel

El sistema mostrará un cajon de dialogo con un mensaje acerca del resultado de la adición, para cerrarlo haga click en el boton [OK].

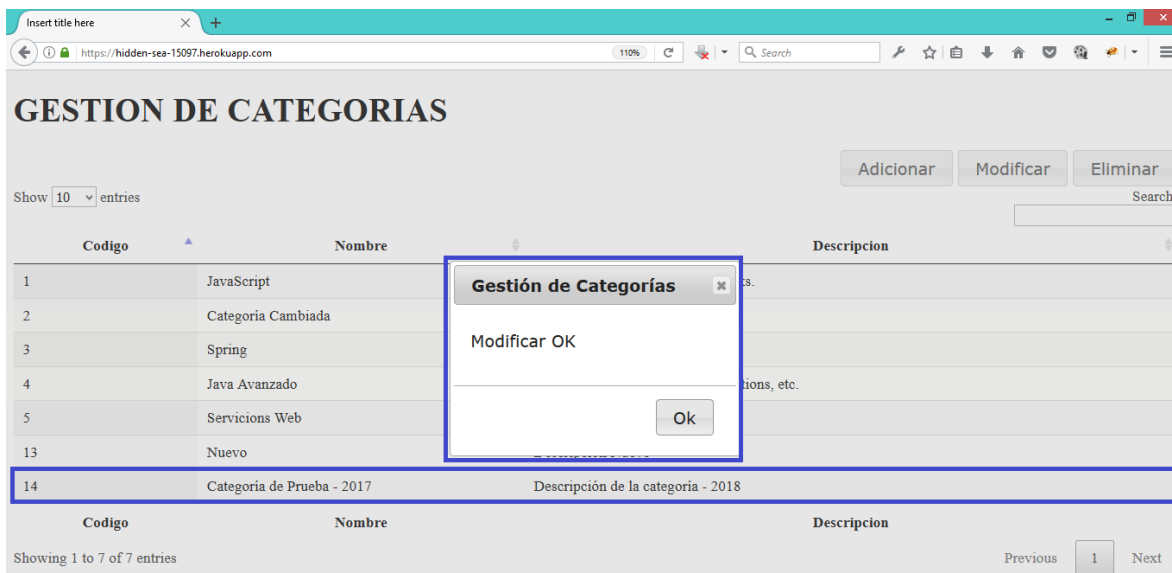


Modificar

Para modificar una categoría seleccione el registro (la fila) que desea cambiar, modifique los datos y haga click en el boton de arriba de [Modificar]. Digite la nueva información y finalmente haga click en el boton de [Modificar].

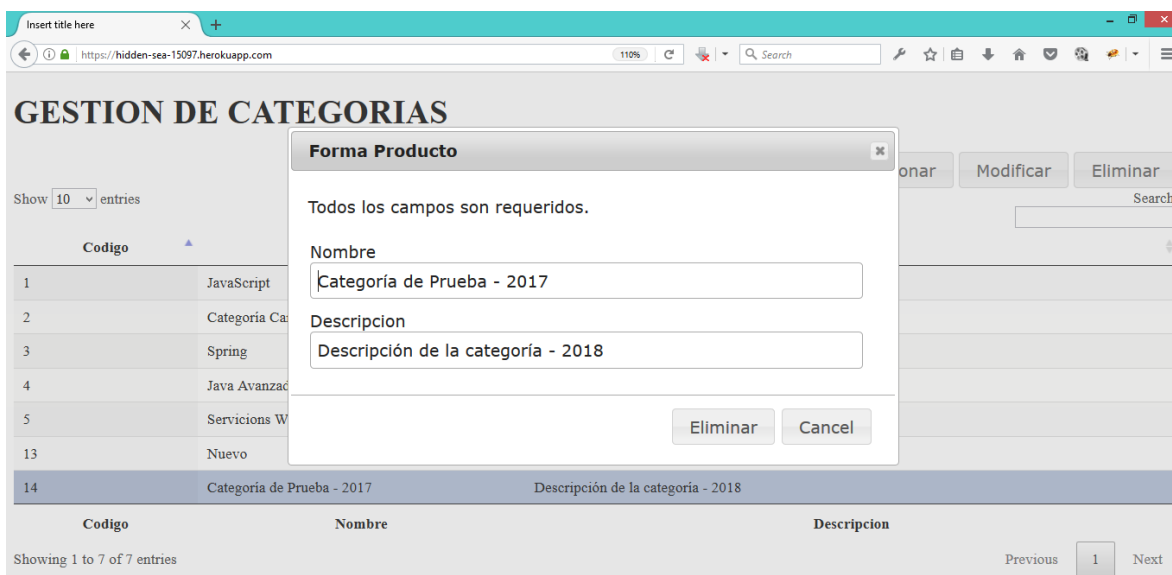


El sistema mostrará un cajon de dialogo con un mensaje acerca del resultado de la modificación.



Eliminar

Para eliminar una categoría (registro), seleccione la fila que desea eliminar y haga click en el boton [Eliminar], aparecerá una ventana con la información de la categoria seleccionada, para confirmar que desea borrar el registro haga click en el boton [Eliminar].



El sistema mostrará un cajon de mensaje con la información del resultado de la operación.

The screenshot shows a web browser window with the URL `https://hidden-sea-15097.herokuapp.com`. The page title is "GESTION DE CATEGORIAS". At the top right, there are buttons for "Adicionar", "Modificar", and "Eliminar". Below these, there is a search bar and a "Show 10 entries" dropdown. The main content area contains a table with columns "Codigo", "Nombre", and "Descripcion". The table lists several categories, including "JavaScript", "Categoria Cambiada", "Spring", "Java Avanzado", "Servicions Web", and "Nuevo". A modal dialog box is open in the center of the screen, titled "Gestión de Categorías", with the text "Eliminar OK" and an "Ok" button. The dialog box is highlighted with a blue border. At the bottom of the page, there is a pagination bar showing "Showing 1 to 6 of 6 entries" and "Previous 1 Next".

Codigo	Nombre	Descripcion
1	JavaScript	
2	Categoria Cambiada	
3	Spring	
4	Java Avanzado	
5	Servicions Web	
13	Nuevo	