



Unidad 2: Evaluación de la Unidad

Asignatura: Taller de Aplicaciones Empresariales

Docente: Iván Bilbao

Alumno: Eliú Martínez

Solución del Siguiente Caso

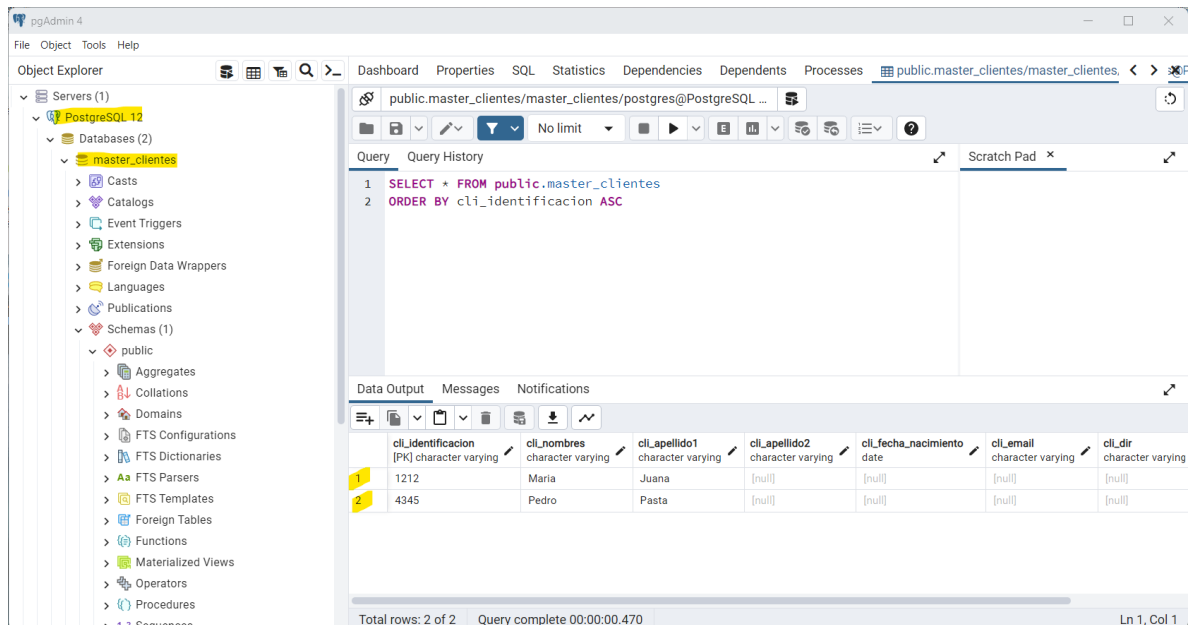
Instrucciones:

1. Crear un proyecto Java del tipo Web Application.
2. Inicializar el proyecto con el sistema de control de versiones git. (git init)
3. Crear una rama en Bitbucket para este proyecto.
4. Subir el código fuente del proyecto local al repositorio de Bitbucket.
5. Crear una tabla (libre elección) en la base de datos PostgreSQL que contenga 5 campos con una llave primaria.
6. Crear Entity y DAO para la tabla creada.
7. Eliminar la página HTML y crear una Página JSP con un formulario que reciba los campos de la tabla creada
8. Crear un servlet, recibir los parámetros del formulario.
9. Crear un objeto Entity y asignarle los valores provenientes del JSP.
10. Crear instancia DAO e invocar método create(), edit() y destroy() , pasándole como parámetro el objeto Entity y verificar que haya realizado cambios en la base de datos.

Crear captura de pantalla del JSP ejecutándose en el navegador y también de la evidencia en base de datos de inserción edición y eliminación, posteriormente los adjunta en un documento Word.

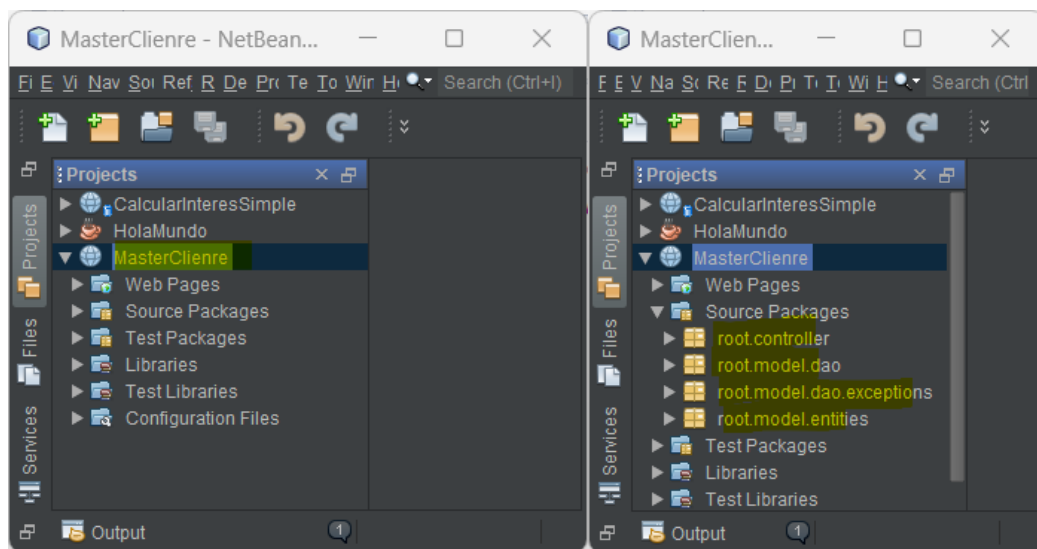
En base a lo investigado, se dará solución a la conexión de BD PostgreSQL, donde se ingresará a través de la interfaz gráfica JSP, un registro que simulará los datos de un cliente.

Base de Datos PostgreSQL 12

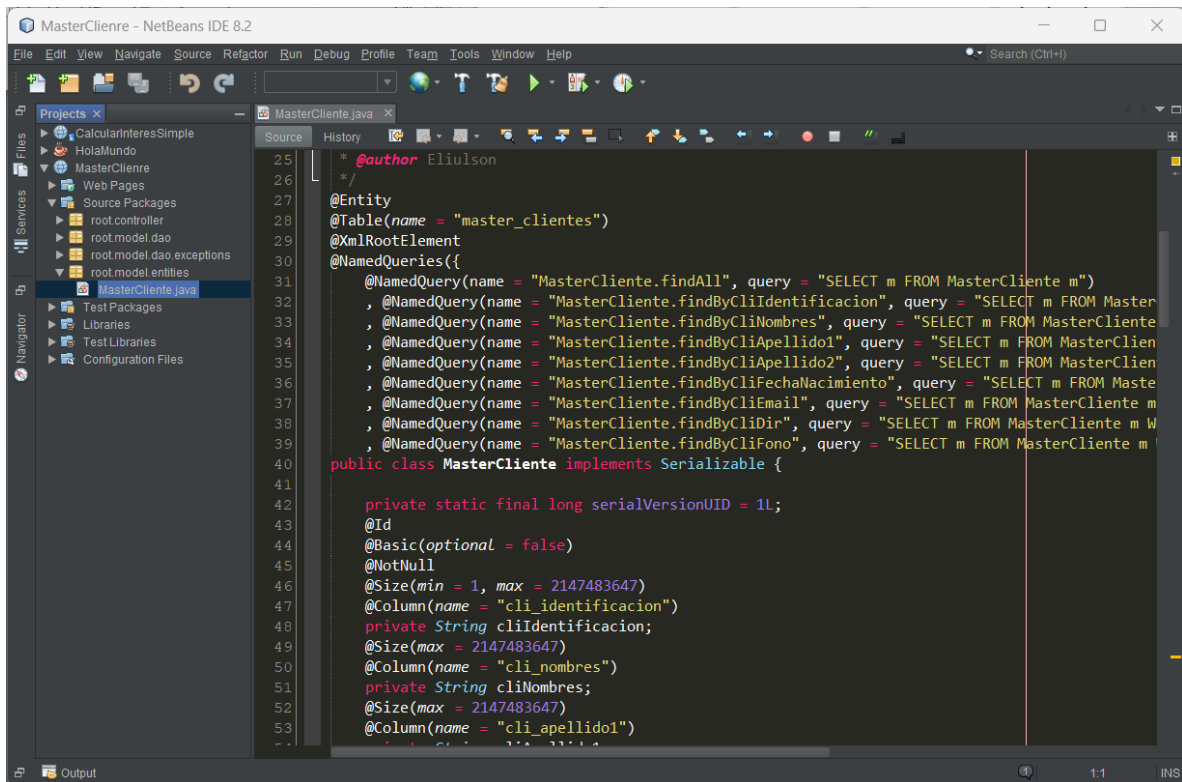


Acá podemos percatarnos que, creé la BD **master_clientes**, así mismo, la tabla con el mismo nombre, y donde se agregaron los campos que se ven en la imagen. Anteriormente había realizado una prueba de conexión a la BD, por lo tanto, he agregado 2 registros, vamos a ver cual fue el proceso que realice y agregaremos más clientes.

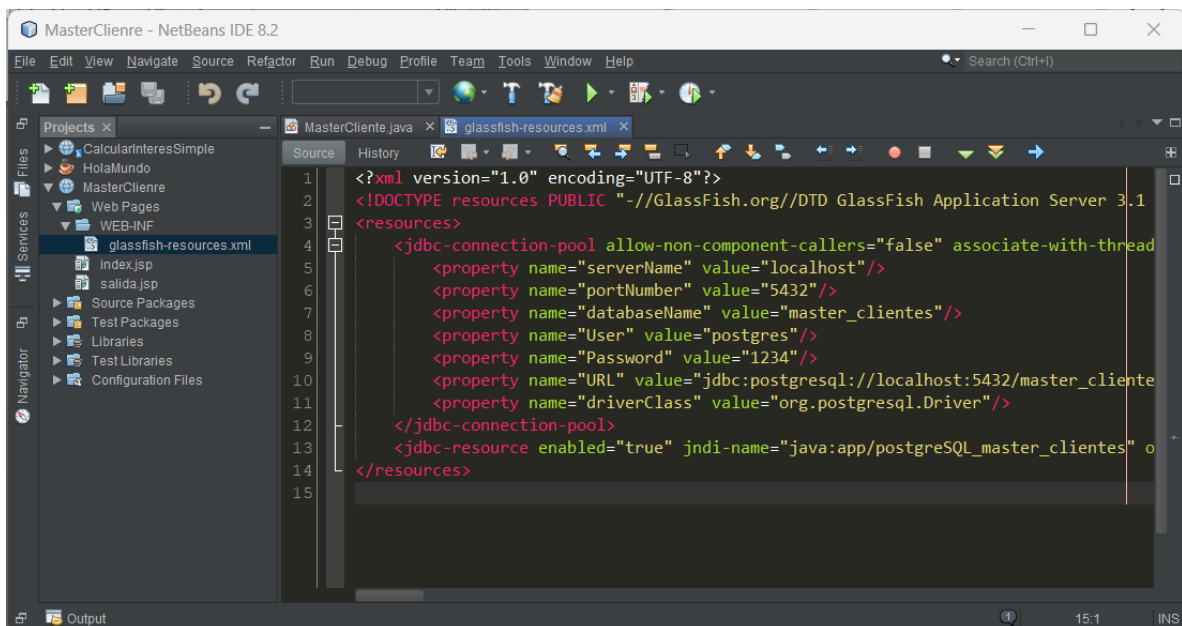
NetBeans 8.2 / Web Application



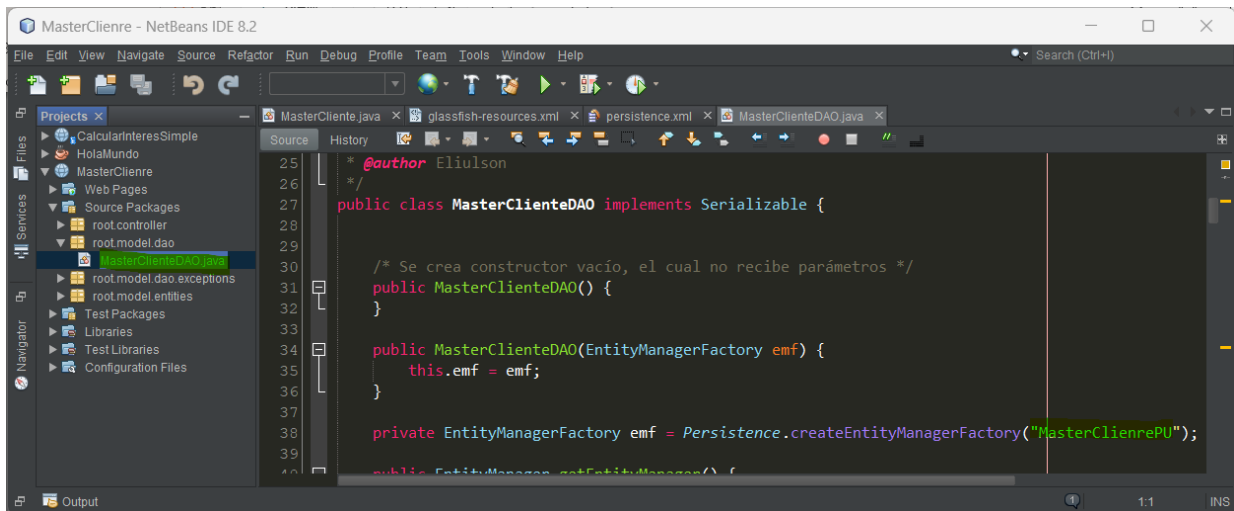
Se creó la app: **MasterClienre**, y además los packages: **root.controller**, **root.model.dao**, **root.exceptions**, **root.model.entities**.



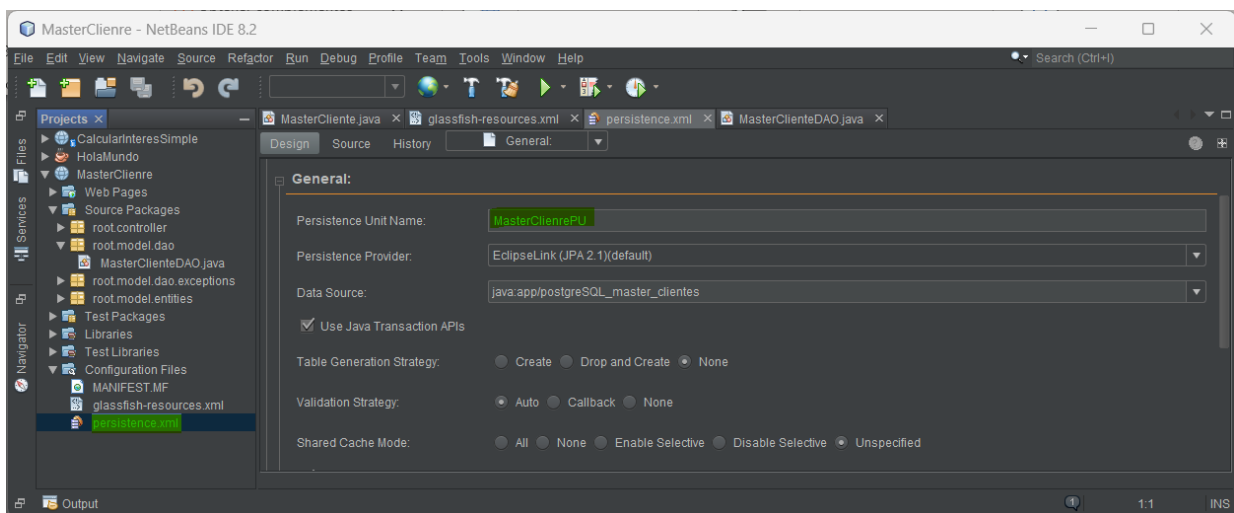
Con ayuda de la interfaz del IDE creamos la clase **Entity**, que refleja, un registro de la tabla: “**master_clientes**”.



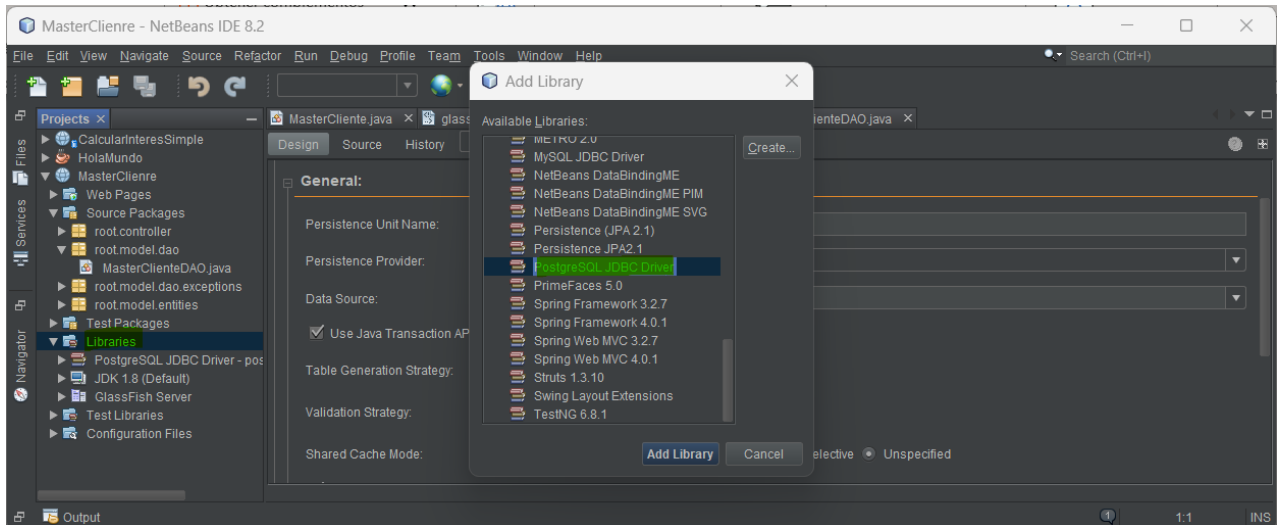
En este caso, yo estoy usando el servidor GlassFish, que permitirá configurar el data source de la BD.



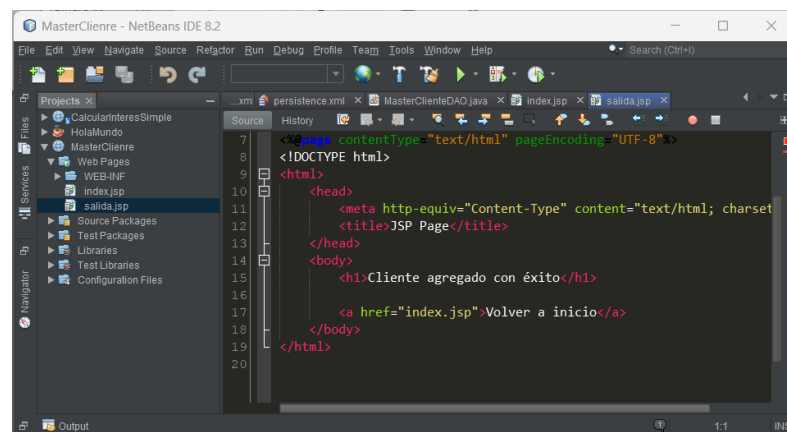
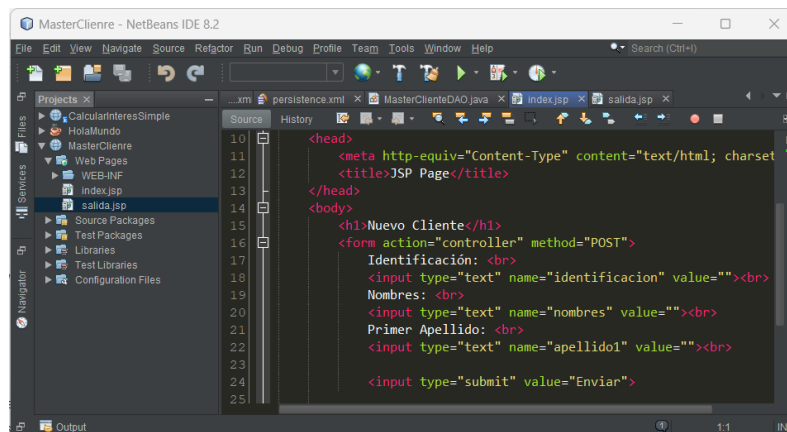
Luego creamos la clase **DAO** (Data Access Object), que nos permitirá, comunicarnos con la BD, a través de comandos **CRUD**, está nuevamente la creamos con la ayuda del IDE.



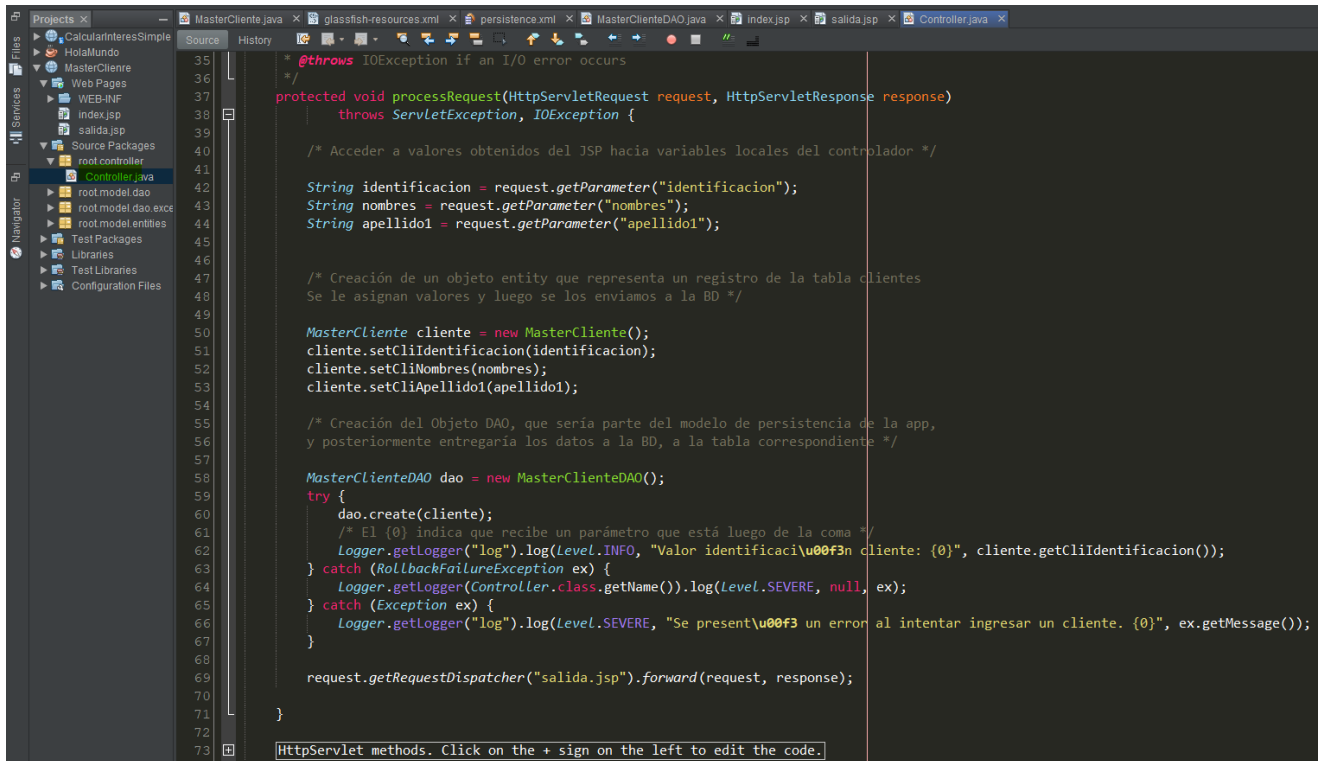
Si nos fijamos en archivo **persistence.xml**, la unidad de persistencia es: **MasterClienrePU**, el mismo nombre que debemos asignar cuando creamos un contexto de persistencia. Esta unidad de persistencia, hace referencia a nuestra base de datos, este se creó automáticamente cuando creamos la clase Entity y le ingresamos la URL de BD, recordar que una clase Entity refleja el registro de una tabla de la base de datos.



Para finalizar nuestra configuración con la BD, debemos agregar la librería de PostgreSQL, para que ésta funcione correctamente.

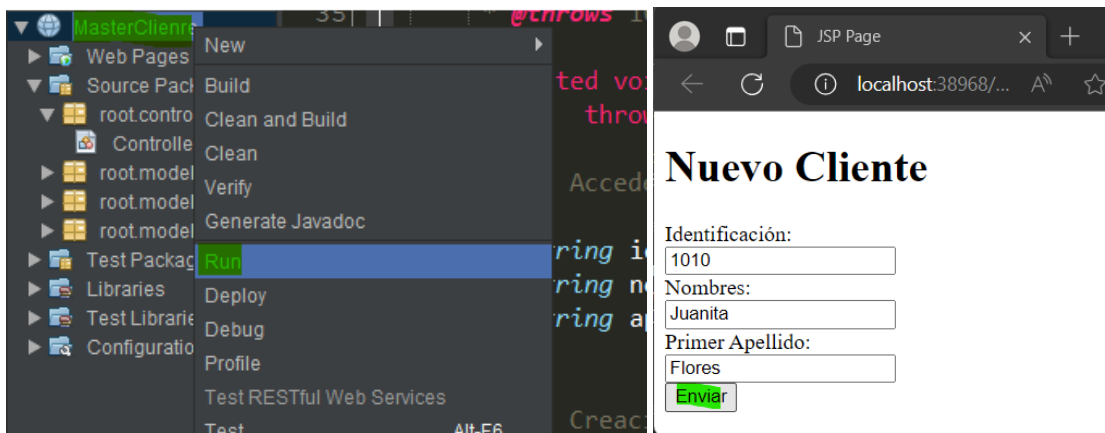


Ahora, configuramos interfaces para el ingreso de datos con formulario y salida del proceso, indicándonos que todo ha salido correctamente, con páginas JSP.

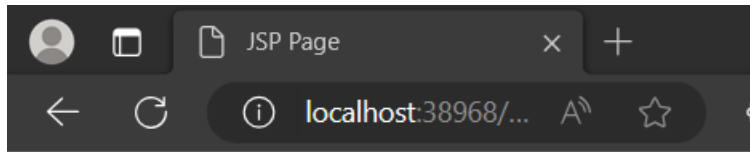


```
35  * @throws IOException if an I/O error occurs
36  */
37  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
38      throws ServletException, IOException {
39
40      /* Acceder a valores obtenidos del JSP hacia variables locales del controlador */
41
42      String identificacion = request.getParameter("identificacion");
43      String nombres = request.getParameter("nombres");
44      String apellido1 = request.getParameter("apellido1");
45
46
47      /* Creación de un objeto entity que representa un registro de la tabla clientes
48      Se le asignan valores y luego se los enviamos a la BD */
49
50      MasterCliente cliente = new MasterCliente();
51      cliente.setCliIdentificacion(identificacion);
52      cliente.setCliNombres(nombres);
53      cliente.setCliApellido1(apellido1);
54
55      /* Creación del Objeto DAO, que sería parte del modelo de persistencia de la app,
56      y posteriormente entregaria los datos a la BD, a la tabla correspondiente */
57
58      MasterClienteDAO dao = new MasterClienteDAO();
59      try {
60          dao.create(cliente);
61          /* El {} indica que recibe un parámetro que está luego de la coma */
62          Logger.getLogger("log").log(Level.INFO, "Valor identificaci\u00f3n cliente: {}", cliente.getCliIdentificacion());
63      } catch (RollbackFailureException ex) {
64          Logger.getLogger(Controllr.class.getName()).log(Level.SEVERE, null, ex);
65      } catch (Exception ex) {
66          Logger.getLogger("log").log(Level.SEVERE, "Se present\u00f3 un error al intentar ingresar un cliente. {}", ex.getMessage());
67      }
68
69      request.getRequestDispatcher("salida.jsp").forward(request, response);
70
71  }
72
73  HttpServlet methods. Click on the + sign on the left to edit the code.
```

Finalmente creamos nuestro controlador, que recibirá los parámetros, y llevará a cabo toda la lógica, ingresando un cliente a la BD, con los parámetros que le dimos por medio del formulario.



Ejecutamos la Aplicación Web y ingresamos el usuario.



Cliente agregado con éxito

[Volver a inicio](#)

Nos reenvía a la salida JSP y ahora vamos a verificar a la BD, si se agregó el cliente.

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure. The main pane displays a SQL query:

```
1 SELECT * FROM public.master_clientes
2 ORDER BY cli_identificacion ASC
```

The Data Output tab shows the results of the query:

	cli_identificacion [PK] character varying	cli_nombres character varying	cli_apellido1 character varying	cli_apellido2 character varying	cli_fecha_nacimiento date	cli_email character v
1	1010	Juanita	Flores	[null]	[null]	[null]
2	1212	Maria	Juana	[null]	[null]	[null]
3	4345	Pedro	Pasta	[null]	[null]	[null]

Total rows: 3 of 3 Query complete 00:00:00.238 Ln 1, Col 1

Como vemos se agregó el registro, que anteriormente habíamos ingresado por el formulario JSP.

Git init / Subir Proyecto a Repositorio

```
MINGW64:/d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicati...
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica
$ git init
Initialized empty Git repository in D:/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica/.git/

Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    MasterClienre/

nothing added to commit but untracked files present (use "git add" to track)

Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$
```

Iniciamos git init, donde se encuentra nuestro proyecto.

```
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$ git add MasterClienre/
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$ git commit -m "MasterClienre 1.0"
```

Agregamos los archivos y confitmamos.

```
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$ git remote add origin https://github.com/gasdar/Proyecto_MasterCliente_BD.git
```

Hacemos conexión remota, para subir los archivos

```
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime5/TallerAplicacionesE/Unidad2/videosExplicativos/practica (master)
$ git push -u origin master
```

Enviamos los archivos al servidor anteriormente conectado.

Código GIT: https://github.com/gasdar/Proyecto_MasterCliente_BD