

## **Grado en Ingeniería Informática**

*Proyectos de Programación*

*Centros de Supercomputación*

Entrega FINAL (v2.0)

Alumno/s:

MIRALLES MARTÍNEZ, ALFONSO ( [alfonsomiralles1@gmail.com](mailto:alfonsomiralles1@gmail.com) )

MUÑOZ MUÑOZ, JAIME ( [jaimegradoviu@gmail.com](mailto:jaimegradoviu@gmail.com) )

PEREIRO SECO, JOSE FRANCISCO ( [jpereirs@gmail.com](mailto:jpereirs@gmail.com) )

ENERO, 2021

# Índice

1. Introducción .....	3
2. Requerimientos del Proyecto .....	4
2.1. Requerimientos Funcionales .....	4
2.2. Requerimientos No-Funcionales .....	5
3. Roles y Organización del Trabajo .....	6
3.1. Roles Técnicos .....	6
3.2. Roles Transversales .....	7
4. Herramientas de Trabajo .....	8
4.1. Herramientas de Colaboración y Coordinación .....	8
4.2. Herramientas de Desarrollo .....	9
5. Diseño de la aplicación .....	10
5.1. Esquema de la Base de Datos .....	10
5.2. Diagrama de Clases .....	11
5.3. Casos de Uso .....	12
6. Desarrollo de la Aplicación .....	13
6.1. Front-End – Lógica de Presentación .....	13
6.2. Back-End – Lógica de Negocio .....	20
6.3. Base de Datos .....	25
7. Anexos .....	29
7.1. Anexo I: Creación de base de datos MariaDB en AWS .....	29
7.2. Anexo II: Creación de tablas con HeidiSQL .....	32
7.3. Anexo III: Video Explicación Front-End .....	38

# 1. Introducción

El presente documento tiene por objeto describir la conceptualización, diseño y desarrollo del proyecto denominado “Centros de Supercomputación”. Este se trata del desarrollo de un software para la gestión de centros de supercomputación y las cargas de trabajo asociadas a los mismos.

A través de esta aplicación podrán gestionarse cada uno de los centros, teniendo la posibilidad de crearlos, eliminarlos o modificarlos, y de igual forma, podrán realizarse operaciones equivalentes sobre los usuarios de la aplicación y las cargas de trabajo de los centros. Además, podremos asignar trabajos a los centros, hacer simulaciones y obtener una visual completa del estado de los trabajos y los centros.

Para el desarrollo del proyecto se ha empleado el lenguaje de programación Java, un lenguaje de alto nivel, orientado a objetos y propósito general desarrollado por Sun Microsystems en 1995. Una de las características principales de Java es que es multiplataforma dado que se ejecuta sobre una máquina virtual (JVM).

Se ha definido una arquitectura de software de 3 capas: lógica de presentación, lógica de negocio y datos. Esta aproximación permite aislar las diferentes lógicas y de esta forma facilitar la división del trabajo y aumentar la modularidad del programa. Por ejemplo, podría cambiarse el aspecto del interfaz gráfico para adaptarlo a una aplicación móvil sin tener que cambiar ni la lógica de negocio ni la base de datos.

Este proyecto es el resultado del trabajo en equipo de Alfonso Miralles Martínez, Jaime Muñoz Muñoz y José Francisco Pereiro Seco para la asignatura de proyectos de programación del Grado en Ingeniería Informática. Quedamos a su disposición para cualquier duda que pudiera tener sobre el mismo.

## 2. Requerimientos del Proyecto

A continuación, procedemos a describir los principales requerimientos de la aplicación, tanto los funcionales como los no-funcionales.

### 2.1. Requerimientos Funcionales

- La aplicación permitirá la gestión de diversos centros de supercomputación, las cargas de trabajo asociadas a cada una de ellas y los usuarios.
- A través de la aplicación podrán crearse, eliminarse y modificarse los centros de supercomputación, los usuarios y los trabajos.
- La aplicación permitirá asignar trabajos a los centros.
- La aplicación permitirá visualizar la situación actual de: 1) las colas de trabajo; 2) trabajos sin asignar; 3) trabajos en proceso. Además, podrá visualizarse el histórico de trabajos procesados.
- Los centros tendrán un nombre de identificador de 30 caracteres máximo, su capacidad de procesamiento medida en teraflops/s (de 1 a 1.000.000,00), tamaño máximo de la cola de trabajos y usuario administrador.
- Los trabajos tendrán un nombre de identificador de 30 caracteres máximo, cantidad de operaciones necesarias para su ejecución (de 1 a 1.000.000.000.000,00) y usuario propietario. Cada usuario solo podrá operar sus trabajos. El administrador del centro solamente podrá dar de baja los trabajos que estén en la cola de su centro esperando para la ejecución. El administrador podrá realizar cualquier operación.
- La asignación de trabajos deberá realizarse mediante una cola FIFO (First-in-First-Out) y el tiempo de ejecución estimado menor. Se considerarán todos los trabajos que no lleguen al máximo de trabajos en cola. Tiempo de ejecución estimado (segundos) =  $((\text{Operaciones faltantes del trabajo actual} + \text{operaciones totales de los trabajos en cola} + \text{operaciones del trabajo}) / \text{Capacidad de procesamiento del centro}) + ((\text{cantidad de trabajos en cola} + 1) * 10)$ . En caso de empate entre diferentes centros no importa a cuál de ellos se asigne. Esta funcionalidad estará restringida a los administradores.
- La funcionalidad de “Simulación Segundos” pedirá la cantidad de segundos a simular. Cada centro “ejecutará” el trabajo actual y de terminarse los adicionales que están en cola hasta que pasen los segundos indicados. Cada vez que se cambia de trabajo se consumen 5 segundos para guardar los resultados y 5 segundos para poner en ejecución el siguiente trabajo. En la primera ejecución de la simulación también debemos considerar los 5 segundos para ponerlo en ejecución, así como cuando la cola se vacíe del todo. A nivel de simulación el tiempo es “continuo”, no importa cuantas veces se interrumpa.

## 2.2. Requerimientos No-Funcionales

- Seguridad:
  - El acceso a la aplicación será autenticado mediante usuario y contraseña.
  - Existirán 3 niveles de privilegio, estos son: administrador, administrador Centro y usuario.
  - Cada usuario tendrá asociado un nivel de privilegio.
  - Cada funcionalidad del sistema tendrá asociado un nivel de privilegio, solo los usuarios con dicho nivel de privilegio podrán ejecutar la funcionalidad.
  - Los identificadores de usuario serán de 1 a 10 caracteres
  - La calidad de la contraseña se establece en un mínimo de 4 caracteres y un máximo de 8.
- Documentación:
  - La aplicación deberá estar documentada mediante Javadocs.
- Lenguaje y metodología de programación:
  - El lenguaje de programación será Java
  - La metodología de programación será orientada a objetos.
- Arquitectura del Software:
  - Se desarrollará una aplicación de 3 capas.

### 3. Roles y Organización del Trabajo

Para una organización eficiente del trabajo se han establecido 3 roles técnicos y 3 roles transversales que han sido distribuidos entre los miembros del equipo. Dicho esto, se ha trabajado en equipo, compartiendo información y conocimientos entre todos los miembros y apoyándose mutuamente cuando ha sido necesario.

#### 3.1. Roles Técnicos

Se han establecido 3 roles técnicos, paralelos a cada una de las 3 capas que conforman la arquitectura del proyecto, estos son:

- **Desarrollador Front-End.** El desarrollador Front-End será el encargado del desarrollo del código fuente correspondiente al interfaz gráfico y la lógica de presentación. Esto incluye el diseño e implementación de ventanas, elementos gráficos para la captura de datos y presentación de información, así como la programación de eventos y botones del interfaz gráfico.
- **Desarrollador Back-End.** El desarrollador back-end se encargará de la lógica de negocio, es decir del conjunto de algoritmos, clases y métodos encargados de implementar la funcionalidad del programa, así como del tratamiento y control de errores de los datos. De igual forma se encargará de las funciones para conectar el Front-end e invocar la base de datos.
- **Desarrollador Base de Datos.** El desarrollador de base de datos será el responsable de definir e implementar el modelo de datos dentro del sistema de gestión de base de datos, establecer los conectores, los datos de prueba y dar soporte al desarrollador back-end para la conexión de ambos entornos.

Los roles descritos han sido asignados a los miembros del equipo conforme a la siguiente tabla.

Rol Técnico	Miembro
Desarrollador Front-End	José Francisco Pereiro Seco
Desarrollador Back-End	Alfonso Miralles Martínez
Desarrollador Base de Datos	Jaime Muñoz Muñoz

## 3.2. Roles Transversales

Adicionalmente a los roles anteriormente descritos, se han establecido 3 roles transversales, encargados de desarrollar funciones de coordinación, diseño y operativas dentro del proyecto, estos son:

- **Jefe de Proyecto y Documentación.** Este será el responsable de coordinar las actividades de los diferentes miembros del equipo, convocar reuniones de seguimiento y supervisar que se alcancen los plazos comprometidos. Además, liderará el desarrollo de la memoria de proyecto, diseñando el índice, asignando tareas e integrando la versión final.
- **Responsable de Integración y Pruebas.** El responsable de integración y pruebas tiene la misión de integrar los diferentes módulos de código para formar un único programa completamente funcional. Además, será el encargado de realizar las pruebas necesarias para comprobar el correcto funcionamiento del programa.
- **Analista Funcional y de Datos.** El analista funcional y de datos será el encargado de diseñar el programa a alto nivel, incluyendo el esquema de base de datos, el diagrama de clases y los casos de uso. Además, definirá y asesorará al resto de miembros del equipo sobre los tipos de datos más adecuados.

Los roles descritos han sido asignados a los miembros del equipo conforme a la siguiente tabla.

Rol Técnico	Miembro
Jefe de Proyecto y Documentación	José Francisco Pereiro Seco
Responsable de Integración y Pruebas	Alfonso Miralles Martínez
Analista Funcional y de Datos	Jaime Muñoz Muñoz

## 4. Herramientas de Trabajo

A continuación, procedemos a describir las principales herramientas de trabajo empleadas por el equipo durante el proyecto:

### 4.1. Herramientas de Colaboración y Coordinación

- **Whatsapp:** Aunque inicialmente se utilizó como herramienta para establecer un foro de debate y coordinación entre los miembros del equipo su uso fue creciendo hasta convertirse en la principal herramienta de trabajo, incluyendo el intercambio de código fuente y documentación de proyecto. Su principal ventaja estriba en que se trata de una aplicación móvil, permitiendo a los miembros del equipo estar conectados desde cualquier ubicación, además su versión web permite muy fácilmente intercambiar ficheros y documentos desde el ordenador. El equipo es plenamente consciente de que trata de una herramienta de uso doméstico y que no escala ni dispone de las funcionalidades de plataformas como Sourceforge, pero para el alcance del proyecto y necesidades del equipo ha sido más que suficiente.
- **Trello:** se trata de una herramienta de gestión de proyectos de tipo SaaS. Es increíblemente sencilla de usar, con una curva de aprendizaje de apenas unos minutos. Permite crear grupos de actividades (llamados tableros) y dentro de los mismos definir las actividades, asignarlas a miembros del equipo y supervisar su estado. Además, es trivial insertar documentos y materiales multimedia en la misma. Si bien no la hemos explotado intensivamente, este proyecto si nos ha permitido familiarizarnos con la misma.
- **Google Meet:** Esta ha sido la herramienta usada para la realización de las reuniones de coordinación del equipo, está integrada de forma nativa con la agenda y email de Google y permite establecer video conferencias de calidad.
- **Otras:** En menor medida hemos utilizado otras herramientas de colaboración y coordinación, a destacar: 1) Screenbits, para la grabación de videos explicativos sobre el proyecto; 2) Youtube (en modo oculto), para publicar videos explicativos entre nosotros; 3) email, para el intercambio de alguna información; 4) Weshare, para compartir algún fichero.



## 4.2. Herramientas de Desarrollo

La aplicación está desarrollada por el lado de base de datos en AWS (Amazon Web Services) con **MariaDB**, que es un sistema de gestión de bases de datos derivado de MySQL con licencia Publica.

Para crear las tablas se utiliza **HeidiSql**, un software libre y de código abierto que permite conectarse a servidores MySQL (y sus derivaciones como MariaDB y Percona Server), así como Microsoft SQL Server y PostgreSQL.

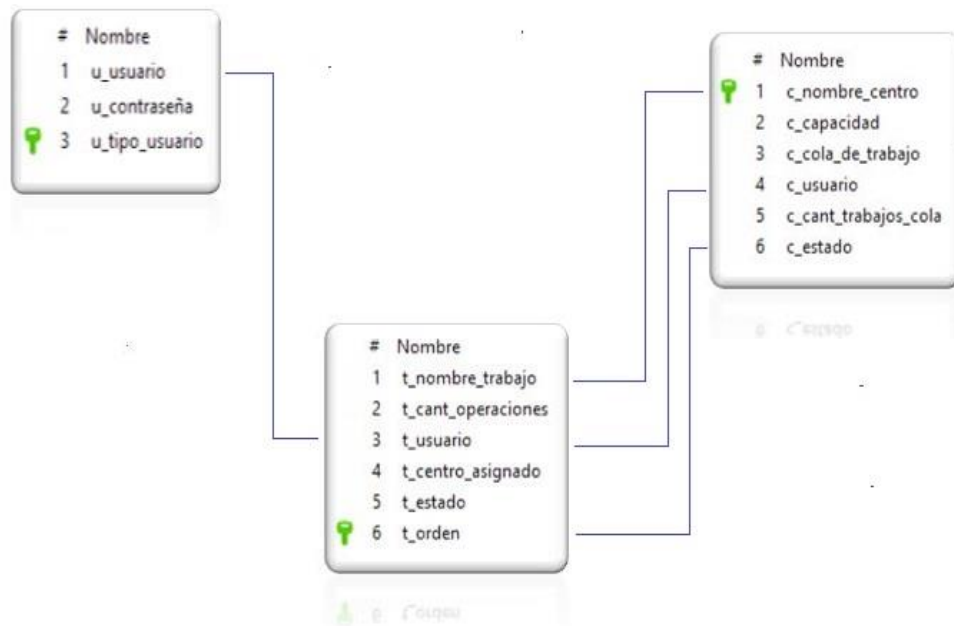
Para la conexión entre la base de datos y el código Java, se utiliza el driver SQL workbench también libre.

Los JFrame son creados a partir del asistente del editor **NetBeans**, con el que también se ha generado el código Java para el proyecto.

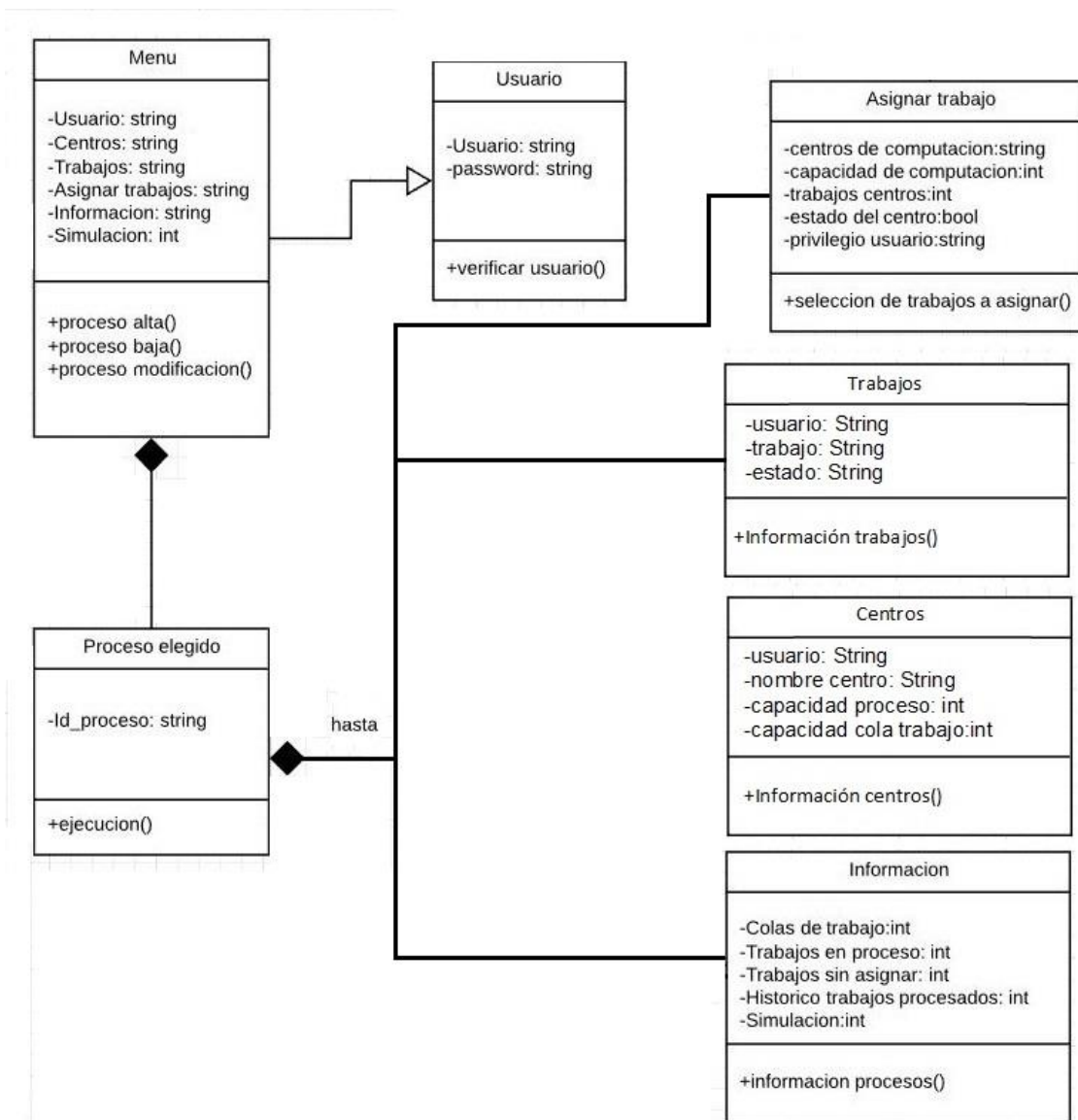
## 5. Diseño de la aplicación

Detalle de los principales elementos de la aplicación a nivel funcional.

### 5.1. Esquema de la Base de Datos



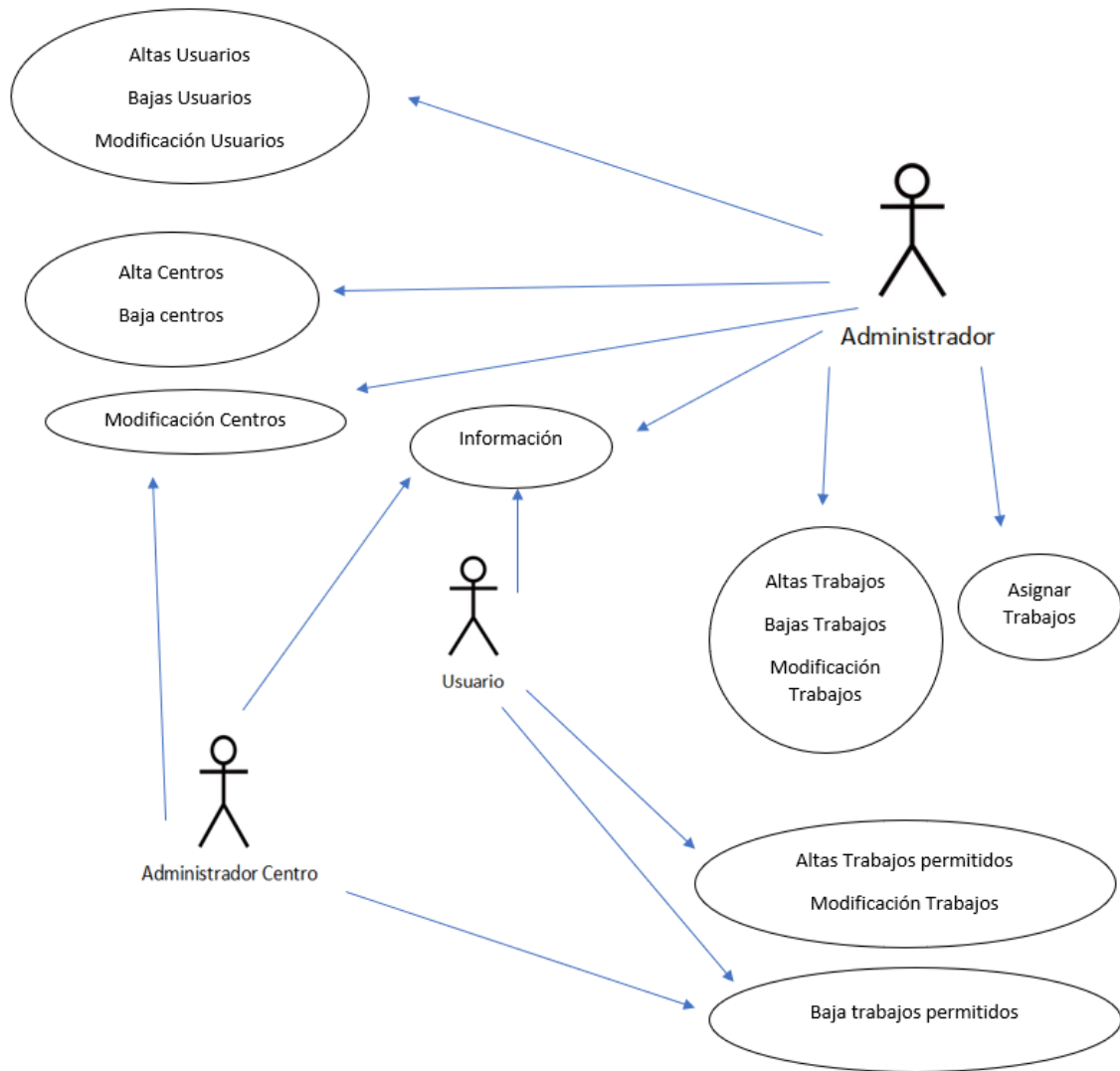
## 5.2. Diagrama de Clases



### 5.3. Casos de Uso

A continuación, se presentan los casos de uso por usuario en el siguiente diagrama.

Cada usuario, dependiendo de sus privilegios, solo podrá realizar ciertas operaciones definidas por programa.



## 6. Desarrollo de la Aplicación

A continuación, procederemos a destacar los elementos y aspectos principales en el desarrollo de las 3 capas de la aplicación.

### 6.1. Front-End – Lógica de Presentación

El elemento base para el desarrollo del Front-end ha sido la clase **JFrame**, esta define una ventana en la que podremos insertar otras clases para conformar el aspecto y contenido de estas. En total, se han tenido que crear 14 ventanas para la realización del proyecto. Resumamos las clases principales que han sido empleadas para diseñar el contenido de las ventanas:

- **JLabel**: Esta clase permite insertar etiquetas y texto dentro de la ventana. Admite tanto texto plano como HTML.
- **JButton**: esta clase nos permite insertar un botón dentro de la ventana, dentro de este botón se podrá insertar el código fuente que debe ejecutarse cuando se pulsa el botón.
- **JTextField**: esta clase permite definir un campo que podremos utilizar para capturar información introducida por el usuario o mostrar el resultado de alguna de nuestras variables. Al ser un campo texto, si capturamos datos de otro tipo tendremos que convertirlos.
- **JPasswordField**: similar a `JTextField` pero con la diferencia que está especialmente diseñado para la captura de contraseñas, mostrando asteriscos para proteger la seguridad de esta. Presenta la particularidad de que no devuelve un tipo `String`, sino un array de caracteres.
- **Choice**: esta clase muestra un menú desplegable que permite al usuario elegir un valor dentro de una lista cerrada.
- **JTextArea**: esta clase permite mostrar textos extensos por pantallas, incorporando barras de desplazamiento horizontal y vertical cuando es necesario.

Cada ventana se trata de un objeto independiente y por tanto tiene sus datos encapsulados y no son visibles a otros objetos. Sin embargo, era necesario pasar uno de los campos (el campo usuario) cuando se navegaba de algunas ventanas a otras. La razón de ello es que era necesario saber que usuario estaba ejecutando la funcionalidad para conocer si tenía privilegios para poder ejecutarla. Esto se ha conseguido insertando el dato en un campo `jTextField` desde la ventana origen. Por ejemplo, mediante el siguiente código de la ventana de Login cargamos el dato en la ventana `Menú_Principal`:

```
Menu_Principal.jTextField1.setText(usuario);
```

A lo largo del programa, ha sido necesario lanzar mensajes al usuario cuando cometía un error o para confirmar que una operación se ha realizado con éxito. Esto se ha realizado mediante el uso de ventanas emergentes pertenecientes a la clase `JOptionPane`. Estas admiten un parámetro de tipo `int` para mostrar el tipo de mensaje (error, aviso, informativo, etc.). Mostremos un ejemplo de cómo se invocan estas ventanas emergentes:

```
JOptionPane.showMessageDialog(this, "Usuario o Contraseña Incorrectos", "AVISO!", 0);
```

Las acciones de usuario se ejecutan tras hacer “*clic*” en los botones del interfaz, esto invoca a una función que ejecuta el código fuente asociado a la misma. Dentro de estas se inserta el código de backend con la funcionalidad de negocio. Por ejemplo, en la ventana Login, al pulsar el botón se invoca a la función:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
```

La captura de datos entrada de usuario se realiza a través del método `getText()` aunque para el campo tipo password es necesario emplear un método diferente, `getPassword()` que devuelve una cadena de caracteres en lugar de un tipo `String`. De esta forma, por ejemplo, para capturar el usuario y contraseña de la ventana Login, esto se realiza de la siguiente forma:

```
usuario = this.jTextField1.getText();  
contrasena = this.jPasswordField1.getPassword();
```

No obstante, como el método `getText()` devuelve una cadena de caracteres, cuando el dato a capturar es de un tipo diferente es necesario hacer un casting en la captura. Por ejemplo, si se trata de un tipo `int`, esto se hace de la siguiente forma:

```
capacidadProceso = Integer.parseInt(this.jTextField2.getText());  
tamanoColaTrabajos = Integer.parseInt(this.jTextField3.getText());
```

Todas las clases tienen una clase llamada “constructor”, que tiene el mismo nombre que la clase y que se invoca una única vez al comienzo de la clase. Esta se utiliza habitualmente para inicializar variables y parámetros. Por ejemplo, en la clase `Menu_Principal` hemos utilizado el constructor para inicializar los valores de entrada de los menús de tipo `Choice`:

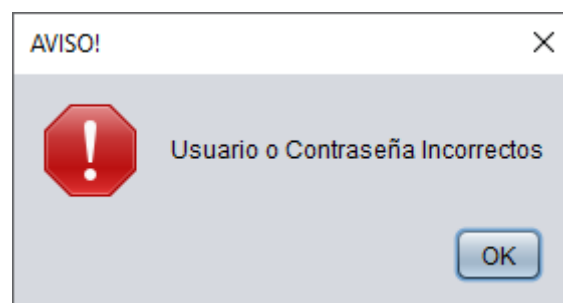
```
public Menu_Principal() {  
    initComponents();  
  
    choice1.add("Usuarios");  
    choice1.add("Centros");  
    choice1.add("Trabajos");  
    choice1.add("Asignar Trabajos");  
    choice1.add("Información");  
    choice1.add("Simulación");  
    choice1.add("Salir");  
  
    add(choice1);  
  
    choice2.add("Alta");  
    choice2.add("Baja");  
    choice2.add("Modificación");  
  
    add(choice2);  
}
```

Además, para cada objeto de la pantalla es posible establecer lo que se llaman “eventos”, es decir acciones que se ejecutarán cada vez que haya algún suceso o cambio de contexto en el interfaz. Por ejemplo, dentro de la clase `Menu_Principal` hay dos menús desplegables de tipo `Choice`. El contenido del segundo menú depende del valor que se haya introducido en el primero. Para resolver esto vamos a emplear el evento:

```
private void choice1ItemStateChanged(java.awt.event.ItemEvent evt) {
```

Este evento se va a invocar cada vez que el usuario haga algún cambio en el primer menú y por tanto dentro de esta función vamos a meter la lógica para definir el contenido del segundo menú en función del primero.

Finalicemos esta sección mostrando algunas de las ventanas diseñadas:





**CENTROS DE SUPERCOMPUTACIÓN**

Seleccione la Operación a Realizar

Usuarios

Alta

**EJECUTAR**

Usuario:

**NUEVO CENTRO**

Nombre Centro:

Capacidad Proceso:  Teraflops/s

Tamaño Cola Trabajos:

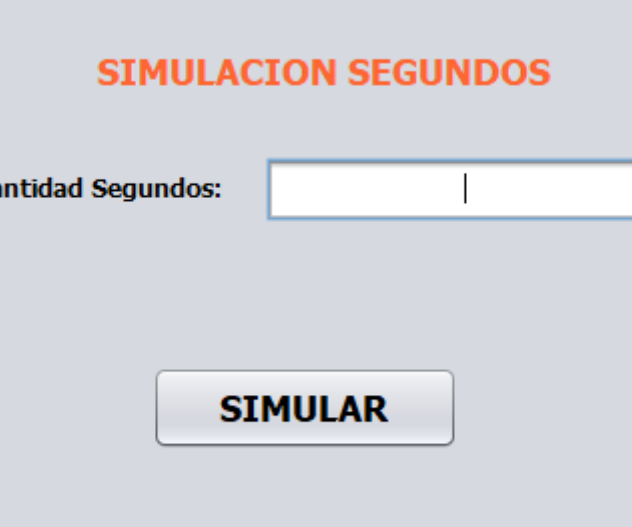
Usuario Administrador:

**CREAR**

INFORMACIÓN ESTADO ACTUAL DEL SISTEMA

Colas de Trabajo en cada Centro	Trabajos sin Asignar
<p>Centro-&gt;'centro1'      Cola-&gt;'2'</p> <p>Centro-&gt;'centro2'      Cola-&gt;'2'</p>	
Trabajos en Proceso en cada Centro	Histórico de Trabajos Procesados
<p>Trabajo-&gt;'atope2'      Centro-&gt;'centro1'</p>	<p>Trabajo-&gt;'atope'      Centro-&gt;'centro1'</p> <p>Trabajo-&gt;'trabajo2'      Centro-&gt;'centro2'</p>

SALIR



**SIMULACION SEGUNDOS**

Cantidad Segundos:

**SIMULAR**



A screenshot of a web application window titled "MODIFICAR TRABAJO". The window has a light gray background and a white title bar with standard minimize, maximize, and close buttons. The main content area contains the following elements: a title "MODIFICAR TRABAJO" in bold orange text; a label "Nombre Trabajo:" followed by a white text input field; a label "Cantidad Operaciones:" followed by a white text input field; a large, rounded rectangular button with the text "MODIFICAR" in bold black text; and a label "Usuario:" followed by a white text input field.

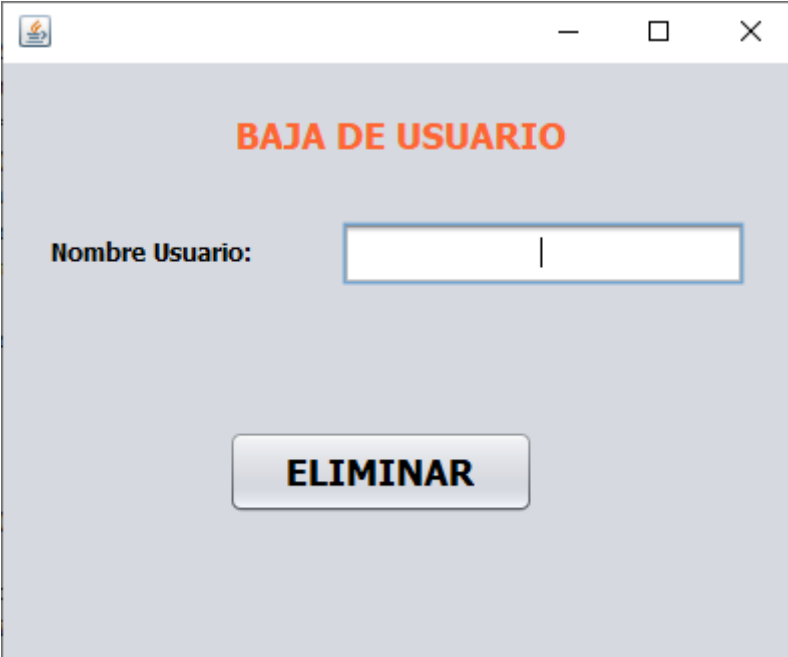
**MODIFICAR TRABAJO**

Nombre Trabajo:

Cantidad Operaciones:

**MODIFICAR**

Usuario:



A screenshot of a web application window titled "BAJA DE USUARIO". The window has a light gray background and a white title bar with standard minimize, maximize, and close buttons. The main content area contains the following elements: a title "BAJA DE USUARIO" in bold orange text; a label "Nombre Usuario:" followed by a white text input field; and a large, rounded rectangular button with the text "ELIMINAR" in bold black text.

**BAJA DE USUARIO**

Nombre Usuario:

**ELIMINAR**

## 6.2. Back-End – Lógica de Negocio

El elemento clave para desarrollar el Back-End del proyecto ha sido conectar el código lógico con la parte desarrollada de Front-End y con la Base de Datos, de modo que se pudiese insertar y extraer la información necesaria. Para ello, ha sido necesario en primera instancia, tener acceso a datos clave de la Base de Datos como:

- DB instance identifier: centros
- Master username: admin
- Master password: cencentros1
- Port: 3306
- Endpoint: centros.ckcro2r2me.us-east-1.rds.amazonaws.com

Con esta información, es posible conectarse a HeidiSQL, desde el código de netbeans, para poder visualizar las tablas diseñadas previamente.

Para realizar la conexión entre netbeans y HeidiSQL, se ha utilizado la librería java.sql.\*, que ha sido necesaria importarla en todas y cada una de las clases del proyecto.

Primero se crea la siguiente variable, al principio de cada clase para que sea accesible de forma global en toda la clase:

```
Connection conn = null;
```

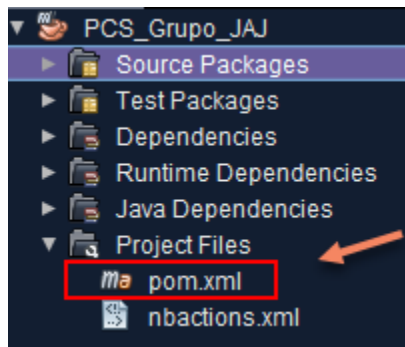
Después podemos implementar la conexión a través de un método try catch. Como ejemplo, se muestra la conexión para verificar el login:

```
try //comprobar las credenciales de usuario
{
    conn = DriverManager.getConnection("jdbc:mysql://centros.ckcro2r2me.us-east-1.rds.amazonaws.com:3306/centrosdb","admin","cencentros1");
    Statement stmt=conn.createStatement();
    ResultSet rs = stmt.executeQuery("select * from usuarios_db where u_usuario='"+usuario+"' and u_contraseña='"+Arrays.toString(contrasena)+"'");
    if (rs.next()){
        new Menu_Principal().setVisible(true);
        Menu_Principal.jTextField1.setText(usuario);
        this.setVisible(false);
    }else{
        JOptionPane.showMessageDialog(this, "Usuario o Contraseña Incorrectos", "AVISO!", 0);
    }
    conn.close();
}
catch(Exception e)
{
    System.out.println(e);
}
```

Lo más importante, es conectar java con la Base de Datos de Amazon (ésta a su vez está conectada con HeidiSQL), introduciendo el endpoint, puerto, usuario y contraseña. Una vez hecho esto, lanzamos un executeQuery que permite realizar consultas a la BBDD a través de código SQL. En este caso, se verifica que el usuario y la contraseña introducidos, estén dentro de la tabla usuarios\_db. Si esto es correcto, entonces el programa da acceso al usuario al Menú Principal. En caso contrario, o si se introducen caracteres no permitidos, el programa lanzará un error o un mensaje indicando que el usuario y contraseñas introducidos no son correctos.

Uno de los problemas principales a la hora de realizar la conexión JDBC con mysql, es que netbeans en primera instancia no conectaba correctamente. Por un lado, se instaló mysql tal y como se informa en la guía del proyecto, pero seguía fallando la conexión. Una de las soluciones posibles era añadir el archivo de conexión .jar de <https://dev.mysql.com/downloads/connector/j/> y después agregarlo al proyecto dentro de una de sus carpetas. Esto se realizó y seguía sin conectar.

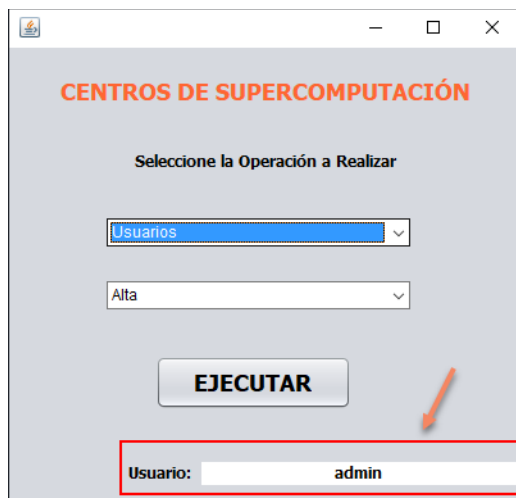
Finalmente, la solución al problema fue introducir código dentro del pom.xml del proyecto.



```
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
    <version>8.0.22</version>
  </dependency>
</dependencies>
```

Con esto, se solucionó el problema de la conexión.

Para bloquear el acceso a usuarios según su rol o permitirse, se han creado una consulta que detecta el rol de usuario según el usuario que ha hecho login y que ya está conectado al software. Esto se ha hecho comprobándolo con el campo de texto que se ha establecido en todas las ventanas del programa.



A continuación, se muestra el código:

```
try //comprobar las credenciales de usuario
{
    conn = DriverManager.getConnection("jdbc:mysql://centros.ckcrope2r2me.us-east-1.rds.amazonaws.com:3306/centrosdb","admin","cencentros1");
    Statement stmt=conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT u_tipo_usuario FROM usuarios_db where u_usuario='"+usuario+"'");
    while (rs.next()){
        String tipo=rs.getString("u_tipo_usuario");
        if (null != tipo)switch (tipo) {
            case "Administrador":
                rt=1;
                break;
            case "Administrador Centro":
                rt=2;
                break;
            case "Usuario":
                rt=3;
                break;
            default:
                break;
        }
    }
    conn.close();
}
catch(Exception e)
{
    System.out.println(e);
}
return rt;
```

Este try catch, se encuentra ubicado dentro de un método que al final retorna un valor según el tipo de usuario.

Este valor, es el que nos va a permitir dar o denegar el acceso a las clases siguientes. Por ejemplo, si el usuario es administrador, tendrá un valor de 1:

```
if (acceso == 1){
    new Usuarios_Alta().setVisible(true);
} else {
    JOptionPane.showMessageDialog(this, "El usuario no tiene privilegios para ejecutar esta acción", "AVISO!", 0);
}
```

Para mostrar la información por pantalla que se pide en la guía, se realiza una consulta a la BBDD y si se cumple ejecutamos un bucle while, extraemos la información en las variables que necesitemos y luego se añaden con un append al JTextArea definido en el Front-End. Como, por ejemplo:

```
jTextArea1.append("Centro->" + cnombre + "\tCola->" + ccola + "\n\n");
```

Para asignar trabajos a centros, se ha tenido en cuenta el orden de creación de los trabajos. Para esto, se ha creado una columna en la BBDD que es autoincremental. Esto quiere decir, que cada vez que se crea un nuevo trabajo en el sistema, se le asigna un número nuevo de forma ascendente. Gracias a esto, si en la consulta se pide que se ordenen los datos de menor a mayor, podemos obtener directamente el dato que fue creado primero.

El código de asignar trabajos se encuentra dentro de la clase Menu\_Principal, ya que esta opción se ejecuta directamente desde el menú.

Aquí hay que tener en cuenta varios aspectos, por un lado, que el centro se encuentre en estado “Libre”, que el trabajo sea de orden FIFO, hay que tener en cuenta la capacidad de procesamiento del Centro y la cantidad de operaciones del trabajo a asignar.

Una vez se asigna un trabajo a un centro, la cola de trabajo del centro se incrementa y se comprueba si la cola de trabajo ha llegado al máximo. En el caso de que la cola de trabajo llegue al máximo, el estado del centro de trabajo pasaría a “Ocupado”. Se puede ver más claramente en HeidiSQL. En cuanto a los estados de los trabajos, pueden pasar de estar “Sin Asignar”, a “En cola”.

```
conn = DriverManager.getConnection("jdbc:mysql://centros.ckcropo2r2me.us-east-1.rds.amazonaws.com:3306/centrosdb","admin","cencentros1");
Statement stmt=conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM trabajos_centros WHERE t_estado='Sin Asignar' ORDER BY t_orden ASC LIMIT 1");
if (rs.next()){
    ResultSet ri = stmt.executeQuery("SELECT * FROM centros_computacion WHERE c_estado='Libre' ORDER BY c_capacidad ASC LIMIT 1");
    if (ri.next()){
        stmt.executeUpdate("UPDATE trabajos_centros SET t_estado='En cola' WHERE t_estado='Sin Asignar' ORDER BY t_orden ASC LIMIT 1");
        ResultSet rk = stmt.executeQuery("SELECT * FROM centros_computacion WHERE c_estado='Libre' ORDER BY cCola_de_trabajo DESC LIMIT 1");
        if (rk.next()){
            int cCola = rk.getInt("cCola_de_trabajo");
            int ctrabajos=rk.getInt("c_cant_trabajosCola");
            String centro=rk.getString("c_nombre_centro");
            stmt.executeUpdate("UPDATE trabajos_centros SET t_centro_asignado='"+centro+"' WHERE t_centro_asignado='vacio' ORDER BY t_orden ASC LIMIT 1");
            ctrabajos=ctrabajos+1;
            stmt.executeUpdate("UPDATE centros_computacion SET c_cant_trabajosCola='"+ctrabajos+"' WHERE c_estado='Libre' ORDER BY cCola_de_trabajo DESC LIMIT 1");
            if (ctrabajos>=cCola){
                stmt.executeUpdate("UPDATE centros_computacion SET c_estado='Ocupado' WHERE c_estado='Libre' ORDER BY c_capacidad LIMIT 1");
            }
        }
    }
}
```

Para realizar la simulación, se debe tener principalmente en cuenta el estado del trabajo, si se encuentra en proceso o si se encuentra en cola. Una vez se comprueba esta información se obtienen los datos necesarios para realizar el cálculo, en este caso nos interesa la cantidad de operaciones del trabajo y la capacidad del centro de trabajo. El bucle para que empiece a realizar los cálculos según los segundos introducidos, se va a iniciar con un contador de tiempo en el segundo 6.

Esto es debido a que el programa consume 5 segundos para iniciar la simulación y se le añade un segundo más porque para la lógica se ha utilizado que el resultado inicial sea igual al resultado menos la capacidad de operaciones del centro. Tal y como se puede ver a continuación en esta parte de código.

```
ResultSet re = stmt.executeQuery("SELECT * FROM trabajos_centros WHERE t_estado='En proceso' ORDER BY t_orden ASC LIMIT 1");
if (re.next()) {
    cant_op = re.getInt("t_cant_operaciones");
    ResultSet rs = stmt.executeQuery("SELECT * FROM centros_computacion ORDER BY c_capacidad DESC LIMIT 1");
    if (rs.next()) {
        cap_op = rs.getInt("c_capacidad");
        resultado = cant_op;
        for (int tiempo = 6; tiempo <= segundos; tiempo++) {
            resultado = resultado - cap_op;
        }
    }
}
```

Luego si el resultado es menor a 0, el trabajo se habrá ejecutado y deberá cambiar a estado “Finalizado”. Y se seguirá ejecutando la acción de simulación mientras sigan quedando segundos.

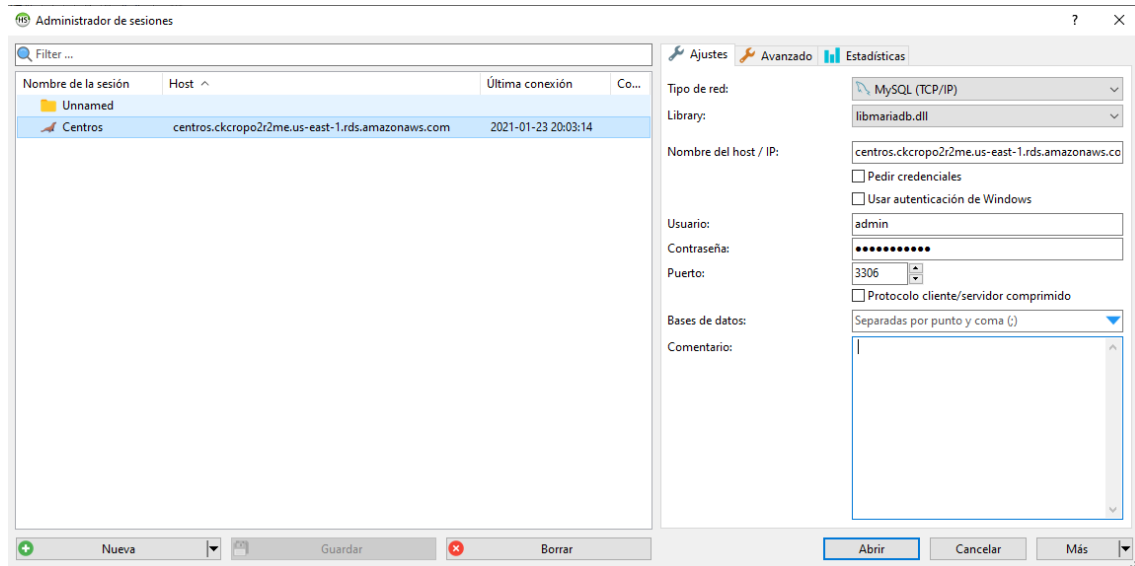
Esta parte, sin duda ha sido la más difícil, ya que hay que tener en cuenta muchos factores, muchos estados y estar continuamente cambiándolos a la vez que se van consumiendo segundos de simulación.



## 6.3. Base de Datos

Para el proyecto se ha creado una base de datos con MariaDB en el servidor Amazon Web Services (AWS).

Una vez creada la base de datos, hemos procedido a crear las tablas con HeidiSql.



Después de instalar HeidiSql hemos de proporcionarle el endpoint y el puerto facilitado en AWS al crear la base de datos con MariaDB.

A su vez proporcionamos el usuario y contraseña de nuestra base de datos creada, “centrosDB” que ya introducimos al crearla en AWS para asegurarnos de su privacidad, así pues, estas credenciales, usuario y password han de introducirse también para la conexión desde HeidiSql a nuestra base de datos, en este caso, como mencionamos anteriormente con MariaDB.

Una vez introducidos todos los datos correctamente hacemos la conexión y comenzamos a crear las tablas/ficheros con los campos que necesitamos para nuestro proyecto.

En este caso hemos creado 3 tablas como se muestra en el siguiente gráfico.

Nombre	Filas	Tamaño	Creado	Actualizado	Motor	Comentario	Tipo
centros_computacion	2	48,0 KiB	2021-01-23 11:04:20	2021-01-23 15:20:44	InnoDB		Table
trabajos_centros	6	48,0 KiB	2021-01-23 11:25:47	2021-01-23 15:20:30	InnoDB		Table
usuarios_db	10	48,0 KiB	2021-01-22 16:56:12	2021-01-23 19:04:23	InnoDB		Table

```

14 SHOW TABLE STATUS FROM `centrosdb`;
15 SHOW FUNCTION STATUS WHERE `Db`='centrosdb';
16 SHOW PROCEDURE STATUS WHERE `Db`='centrosdb';
17 SHOW TRIGGERS FROM `centrosdb`;
18 SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.`EVENTS` WHERE `EVENT_SCHEMA`='centrosdb';

```

La base de datos es centroDB y las tablas las siguientes:

Tabla centros\_computacion, con los siguientes campos.

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con ceros	Predeterminado
1	c_nombre_centro	CHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
2	c_capacidad	INT	7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
3	cCola_de_trabajo	INT	2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
4	c_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
5	c_cant_trabajosCola	INT	3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
6	c_estado	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'Libre'

```

27 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='centros_computacion' AND REFE
28 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='centros_computacion' AND REFERENCED_TABLE
29 SHOW ENGINES;
30 SHOW COLLATION;
31 SHOW CREATE TABLE `centrosdb`.`centros_computacion`;

```

Tabla trabajos\_centro, con los siguientes campos.

Centros\centrosdb\trabajos\_centros\ - HeidiSQL 11.1.0.6116

Host: centros.ckcropo2r2me.us-east-1.rds.amazonaws.com Base de datos: centrosdb Tabla: trabajos\_centros

Nombre: trabajos\_centros

Comentario:

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con zeros	Predeterminado
1	t_nombre_trabajo	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
2	t_cant_operaciones	INT	13	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
3	t_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
4	t_centro_asignado	CHAR	30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'vacio'
5	t_estado	CHAR	30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'Sin Asignar'
6	t_orden	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT

Columnas: + Agregar -x- Borrar ▲ Subir ▼ Bajar

Ayuda Descartar Guardar

Filtro: Expresión regular

```

32 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' ORDER BY ORDINAL_POSITION;
33 SHOW INDEXES FROM `trabajos_centros` FROM `centrosdb`;
34 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' AND REFERENCED_TABLE_NAME='trabajos_centros';
35 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' AND REFERENCED_TABLE_NAME='trabajos_centros';
36 SHOW CREATE TABLE `centrosdb`.`trabajos_centros`;

```

Conectado: 00:04 MariaDB 10.4.8 Activo durante: 3 días, 03:23 Hora del servidor: Preparado.

Tabla usuarios\_db, con los siguientes campos.

Centros\centrosdb\usuarios\_db\ - HeidiSQL 11.1.0.6116

Host: centros.ckcropo2r2me.us-east-1.rds.amazonaws.com Base de datos: centrosdb Tabla: usuarios\_db

Nombre: usuarios\_db

Comentario:

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con zeros	Predeterminado
1	u_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
2	u_contraseña	CHAR	24	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado
3	u_tipo_usuario	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado

Columnas: + Agregar -x- Borrar ▲ Subir ▼ Bajar

Ayuda Descartar Guardar

Filtro: Expresión regular

```

37 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='usuarios_db' ORDER BY ORDINAL_POSITION;
38 SHOW INDEXES FROM `usuarios_db` FROM `centrosdb`;
39 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='usuarios_db' AND REFERENCED_TABLE_NAME='usuarios_db';
40 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='usuarios_db' AND REFERENCED_TABLE_NAME='usuarios_db';
41 SHOW CREATE TABLE `centrosdb`.`usuarios_db`;

```

Conectado: 00:04 MariaDB 10.4.8 Activo durante: 3 días, 03:24 Hora del servidor: Preparado.

Nota:

Es importante comentar, que para que NetBeans conecte con nuestra base de datos y poder intercambiar dichos datos con nuestro proyecto Java, hemos de añadir el driver sql en el fichero pom.xml de nuestro proyecto, tal y como se observa en el siguiente gráfico:

## 7. Anexos

### 7.1. Anexo I: Creación de base de datos MariaDB en AWS

Nota: Procedimiento/Guía interna (no formal) del equipo de trabajo.

Una vez creamos nuestro usuario podemos entrar en la página de Amazon Web Services.

Os recomiendo que sigáis los pasos para crear la vuestra en vuestro usuario AWS, ya que si entramos en la que estuve trabajando para crear nuestra base de datos consumiremos tiempo que nos puede hacer falta para cambiar algún parámetro si fuera necesario por algún problema con el servidor.

Recordad que me tire 5 días para ver porque no funcionaba y solo eran cambio en la configuración respecto a AWS con MySQL, así que es mejor no gastar créditos de una sola cuenta y mejor de otras cuentas si las tenéis.

De todos modos, todo está bien y si necesitáis crearla, pues no pasa nada, se crea una nueva y después intentáis conectarla desde HeidiSQL, así cogéis práctica.

Total, lo peor que podría pasar es que consumiéramos el tiempo y poner un par de euros cada uno y entrar a cambiar cosas.

A vuestro criterio. También podéis descargar HeidiSQL y metéis los datos que os proporciono en el manual de HeidiSQL que os adjunto y ya os podéis conectar, crear una tabla nueva, campos, claves... etc. Eso sí, mejor borrarlas luego, ¡¡¡y sobre todo no cargaros las que están hechas para el proyecto...!

Si usáis mi usuario, es el siguiente:

Podéis entrar a ver la base de datos con los que tengo propios de acceso. Solo os pedirá mi email y mi password:

email --> jaimegadoviu@gmail.com

pass ---> Proyectoajj99#

Nombre de cuenta AWSjaime

\*Importante: Intentad no estar mucho tiempo, pues es tiempo que se consume y se desactivaría la cuenta, es por eso por lo que espere al último momento para crear la base de datos.

Bien, podéis entrar desde el siguiente enlace  
<https://www.awseducate.com/signin/SiteLogin?ec=302&startURL=%2Fstudent%2Fs%2F>

Para crear la base de datos en MariaDB, crearemos una nueva y como tarda al menos 20 minutos en crearla, mostrare como crearla y el resto os paso una explicación del resto de parámetros a configurar para su funcionamiento.

Una vez dentro, lo siguiente es entrar en AWS Account en la parte superior derecha.

Después entramos en AWS Educate Starter Account pinchando en el botón naranja.

Después en Your AWS Account Status aparecerá debajo un acceso a la consola: AWS Console, pinchamos y entramos.

Dentro de la consola, crearemos una base de datos con la herramienta RDS marcada en recuadro rojo.

Una vez marcamos RDS, nos presenta la nueva página de la consola donde podemos ver en el recuadro rojo, que existe una instancia creada, ya que indica DB Instances 1/40.

Para crear una nueva base de datos pincharíamos en el botón naranja dentro del recuadro verde:

Vamos a crear una nueva. Seleccionamos Easy create, seleccionamos la base de datos MariaDB y la cuenta Free tier que es la más económica y nos deja hacer diseños con una capacidad suficiente y de forma gratuita, aunque por tiempo limitado.

Roleamos y sigue pidiendo parámetros en los que hemos de asignar un nombre a nuestra base de datos, usuario y contraseña y marcar que será una base de datos publica para poder acceder cualquiera.

Después de unos 10-30 minutos la base de datos esta creada con el nombre que pusimos.

Después, si ponemos seguridad en la ventana de búsqueda arriba, seleccionamos grupo de seguridad o simplemente como en la imagen he puesto la palabra "seguridad" y me sale todo lo que lleve la palabra. Después selecciono grupos de seguridad.

Esto es necesario, pues la base de datos debe tener un grupo de seguridad para poder implantar restricciones o no al acceso y es donde le diremos que podemos acceder desde el punto de acceso que nos dará posteriormente de forma automática.

Pinchamos en grupos de seguridad, lo he traducido al español, vendrá en inglés.

Creemos un grupo de seguridad en nuestra base de datos, en este caso, en la ventana ya viene creado uno que tenía anteriormente, se puede utilizar el mismo para tantas bases de datos creemos después.

En la parte de abajo, vemos una serie de apartados, detalles, reglas de entrada, salida y etiquetas. Nos interesan las reglas de entrada, son las que nos darán el acceso desde cualquier punto, pero para ello hay que configurar una IP global y una máscara Global, así se podrá acceder desde cualquier punto del mundo.

Por supuesto, esto no se haría si se trata de una base de datos de producción o cualquier empresa, pero en nuestro caso, si, pues todos hemos de tener acceso.

Quedando de esta manera:

Como veis, las reglas de entrada tienen el protocolo TCP/IP, el puerto 3306 que usaremos y la nomenclatura de redes adecuada para el acceso desde cualquier sitio.

El tipo MYSQL/Aurora, no es más que la conexión que será a través de SQL, no importaría otro tipo mientras sea SQL, que siempre será el motor de datos.

Una vez terminado esto, podemos volver a la consola, donde pinchamos en RDS para que nos muestre nuestra base de datos.

Una vez allí, si pinchamos un par de veces, nos aparecen los parámetros que nos interesan para poder conectarnos como el host:

- Endpoint o host: centrosdb.c0k3tpt4ujzu.us-east-1.rds.amazonaws.com
- Puerto: 3306
- Estado: Publico

Con el host es como entramos a la base de datos ya creada y hacemos nuestro mantenimiento de entrada y salida de datos.

Observad el resumen en el grafico a continuación:

Creo que no me dejo nada, pero por si acaso, probar a crear una base de datos en vuestro usuario AWS y consultar el PDF que dejó Roger, por si me salte algo.

Os recomiendo usar el navegador en la parte frontal arriba de AWS y no cerrar las pestañas que se vayan abriendo, pues os ahorrarán tiempo para localizar lo que necesitéis.

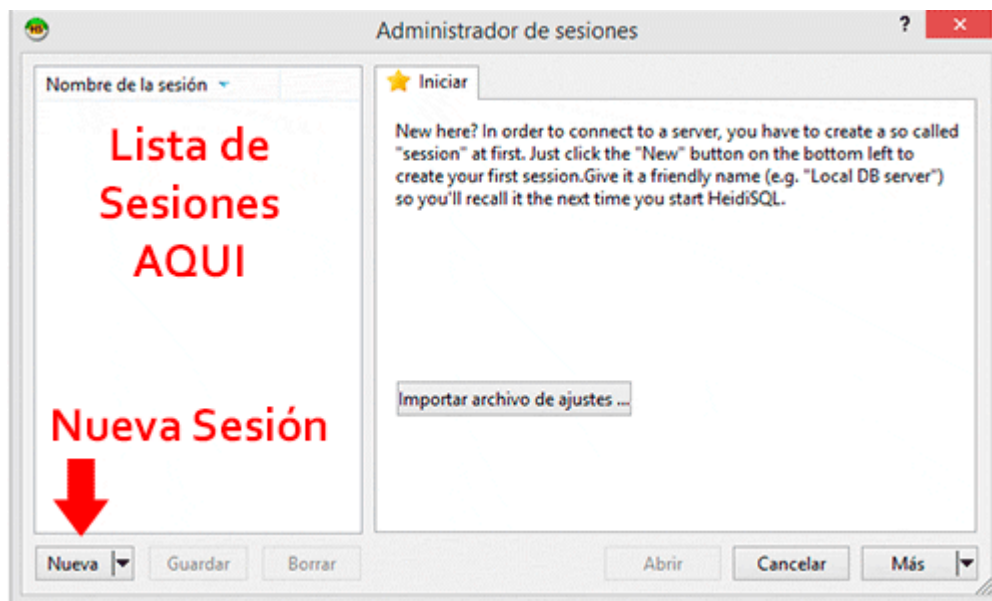
## 7.2. Anexo II: Creación de tablas con HeidiSQL

Nota: Procedimiento/Guía interna (no formal) del equipo de trabajo.

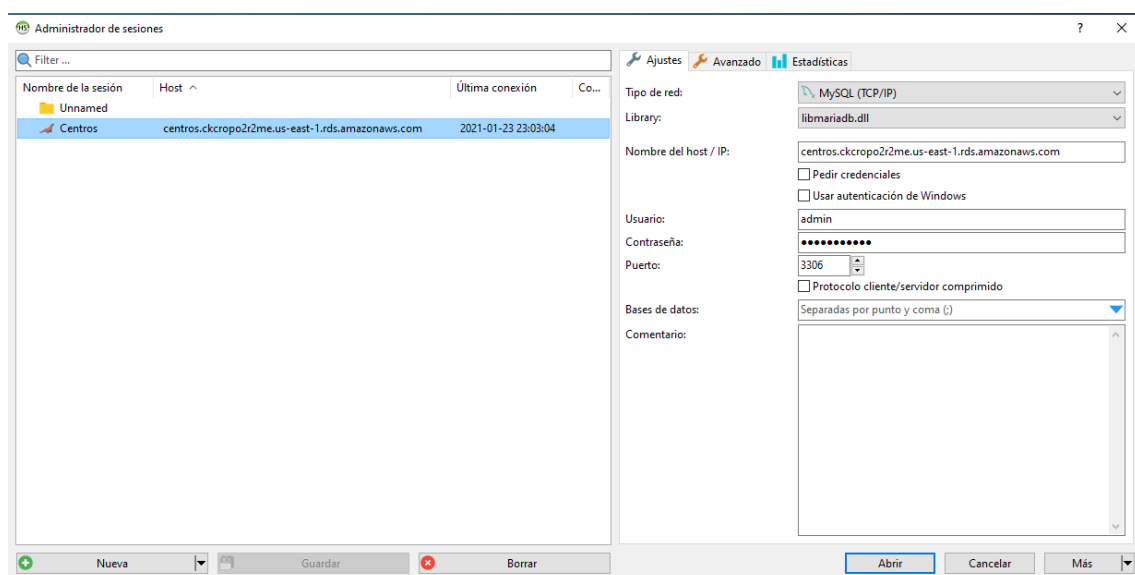
Instalamos el programa desde la página <https://www.heidisql.com/download.php>

La ventana que nos aparece es la siguiente:

Sesión en Heidi



HeidiSQL cuenta un sistema de sesiones para gestionar el trabajo. Cuando iniciamos el programa, debemos seleccionar con que sesión queremos trabajar o en su defecto, crear una nueva sesión presionando el botón «**Nueva**»





En este caso, añadimos una nueva, le damos un nombre cualquiera, en este caso nuestra base de datos se llama “centrosdb” y al nombre de la conexión le he dado “BDcentros”, es un nombre identificativo de la conexión con AWS, no sirve para más.

Seleccionamos en la parte posterior derecha el tipo de conexión, en este caso “MySQL (TCP/IP).

La dll, de la librería se añade por defecto.

- El usuario de la base de datos es: admin
- La contraseña es: cencentros1
- El host o Endpoint: centros.ckcro2r2me.us-east-1.rds.amazonaws.com
- Puerto: 3306

Estos datos nos sirven para acceder a la base de datos en la nube (AWS).

Importante poner el nombre del Host que es la dirección que nos proporciona el AWS cuando creamos la base de datos de MariaDB, tal y como se muestra en la imagen.

El puerto de comunicación es el 3306.

Una vez hecho esto ya podemos pulsar en abrir y nos hace la conexión con MaríaDB de AWS.

Como veis en el siguiente gráfico, ya está conectado con AWS, nuestra base de datos “centrosdb”.

Centros\centrosdb\ - HeidiSQL 11.1.0.6116

Archivo Editar Buscar Consulta Herramientas Ira Ayuda

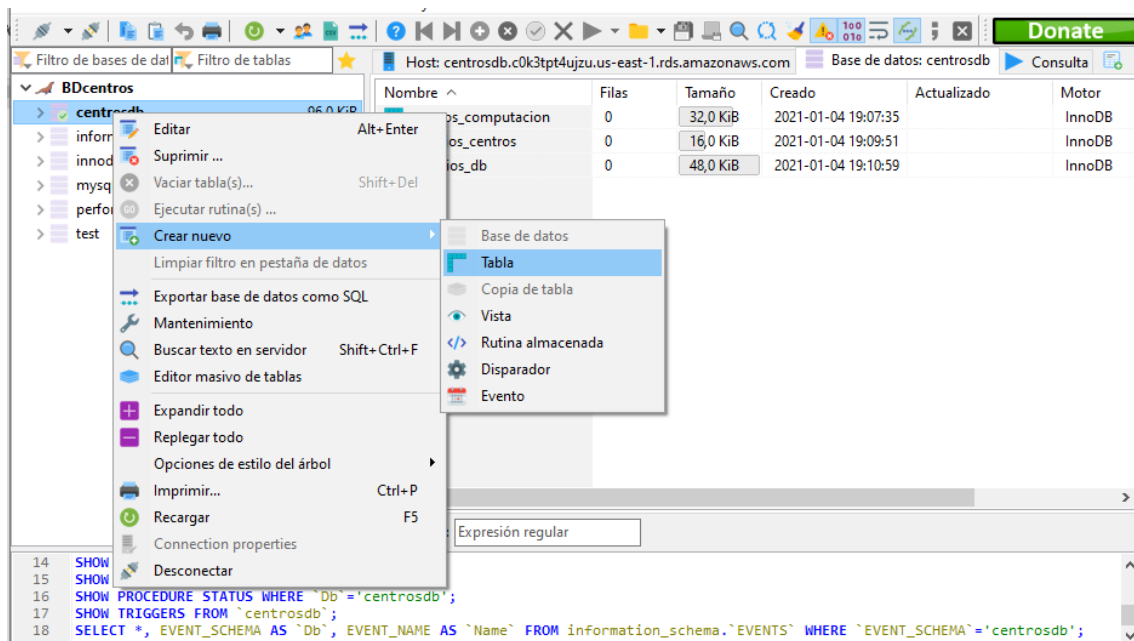
Host: centros.ckcro2r2me.us-east-1.rds.amazonaws.com Base de datos: centrosdb Consulta

Nombre	Filas	Tamaño	Creado	Actualizado	Motor	Comentario	Tipo
centros_computacion	2	48,0 KiB	2021-01-23 11:04:20	2021-01-23 15:20:44	InnoDB		Table
TablaPrueba	1	16,0 KiB	2021-01-24 11:01:29	2021-01-24 11:03:59	InnoDB	Para practicar	Table
trabajos_centros	6	48,0 KiB	2021-01-23 11:25:47	2021-01-23 15:20:30	InnoDB		Table
usuarios_db	9	48,0 KiB	2021-01-22 16:56:12	2021-01-23 19:04:23	InnoDB		Table

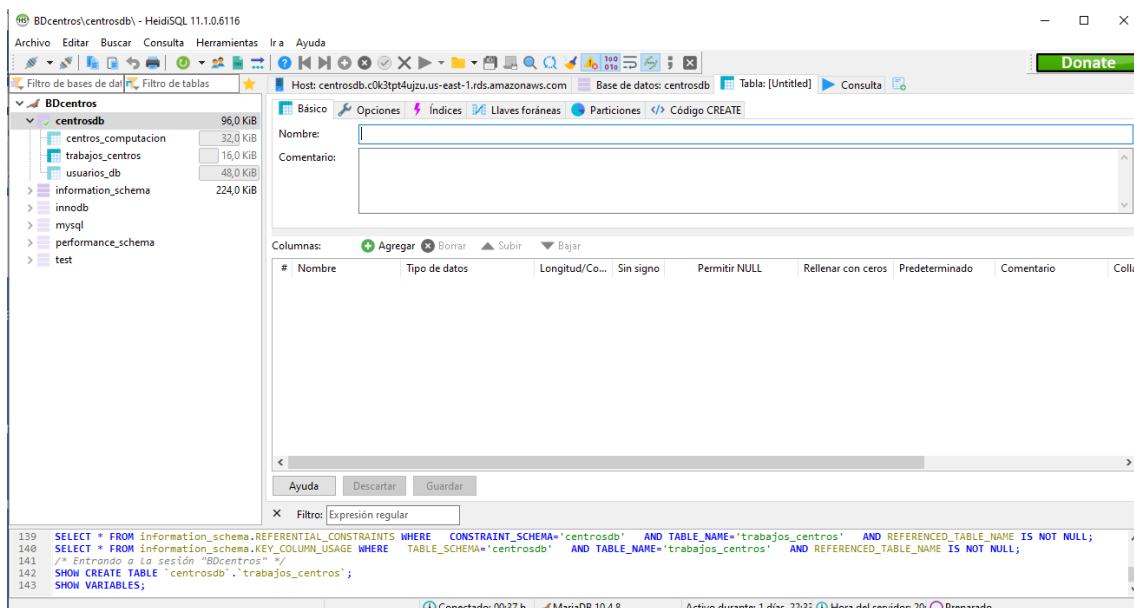
14 SHOW TABLE STATUS FROM `centrosdb`;  
 15 SHOW FUNCTION STATUS WHERE `Db`='centrosdb';  
 16 SHOW PROCEDURE STATUS WHERE `Db`='centrosdb';  
 17 SHOW TRIGGERS FROM `centrosdb`;  
 18 SELECT \*, EVENT\_SCHEMA AS `Db`, EVENT\_NAME AS `Name` FROM information\_schema.`EVENTS` WHERE `EVENT\_SCHEMA`='centrosdb';

Conectado: 00:00 MariaDB 10.4.8 Activo durante: 3 días, 16:55 Hora del servidor: Preparado.

Ahora procedemos a crear las tablas:



Al crear la nueva tabla nos proporciona los campos de entrada en blanco para poder elegir como nombrar la tabla y sus campos:



Como veis, es bastante intuitivo, arriba va el nombre de la tabla, al que se le pueden añadir comentarios sobre las funciones de esta y en la parte de abajo, pulsando el botón agregar, nominamos el campo, el tipo de dato, la longitud, es estatus que queramos que tenga y comentarios sobre el campo para aclarar cualquier duda sobre su nomenclatura.

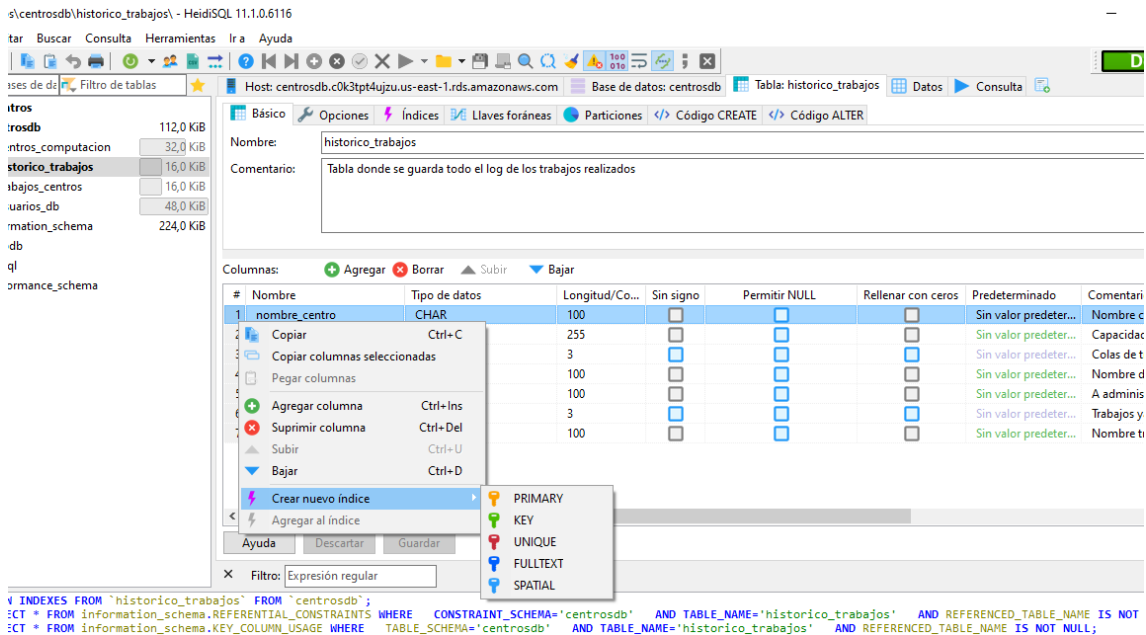
Una vez terminada la tabla, pulsamos guardar y se añade al resto de tablas que forman la base de datos.

## CLAVES

Los campos que serán clave para el acceso a la tabla se ponen una vez creada y es muy simple.

Doble clic en la tabla, buscamos el campo que queremos poner como clave y con botón derecho se despliega el tipo de clave que quieras que sea, única, primaria, etc.

Adjunto Grafico para verlo más claro aún:



Además, se pueden hacer consultas o grabar datos en las tablas ya diseñadas.

## **Agregar datos a tabla**

Para agregar datos a una tabla, seleccionamos la misma, y abrimos la pestaña «Datos»

- Para agregar o remover filas, utilizamos los botones (+) y (-)
- Para agregar datos a cada campo, doble clic y escribir su contenido

## **Consultas**

Para realizar consultas a la base de datos, contamos con la pestaña «> Consulta»

- Área donde escribir SQL
- Botón para ejecutar la consulta
- Área de resultados
- Te permite guardar la consulta en archivo con extensión \*.sql

También podemos introducir los datos en los campos a través de la pestaña «<<Datos>>» justo a la izquierda de «<<Consulta>>».

## DISEÑO FINAL DE LA BASE DE DATOS. (a revisar por si fueran necesario ajustes)

### Usuarios

The screenshot shows the HeidiSQL interface with the 'usuarios\_db' table selected. The table structure is as follows:

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con zeros	Predeterminado	Comentario
1	u_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
2	u_contraseña	CHAR	24	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
3	u_tipo_usuario	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	

The SQL console at the bottom shows the following queries:

```
27 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='usuarios_db' AND REFERENCED_TABLE_NAME IS NOT NULL;
28 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='usuarios_db' AND REFERENCED_TABLE_NAME IS NOT NULL;
29 SHOW ENGINES;
30 SHOW COLLATION;
31 SHOW CREATE TABLE `centrosdb`.`usuarios_db`;
```

### Centros de computación

The screenshot shows the HeidiSQL interface with the 'centros\_computacion' table selected. The table structure is as follows:

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con zeros	Predeterminado	Comentario
1	c_nombre_centro	CHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
2	c_capacidad	INT	7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'	
3	c_cola_de_trabajo	INT	2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'	
4	c_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
5	c_cant_trabajos_cola	INT	3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'0'	
6	c_estado	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'Libre'	

The SQL console at the bottom shows the following queries:

```
32 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='centros_computacion' ORDER BY ORDINAL_POSITION;
33 SHOW INDEXES FROM `centrosdb`.`centros_computacion`;
34 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='centros_computacion' AND REFERENCED_TABLE_NAME IS NOT NULL;
35 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='centros_computacion' AND REFERENCED_TABLE_NAME IS NOT NULL;
36 SHOW CREATE TABLE `centrosdb`.`centros_computacion`;
```

## Trabajos en centro de computación

Centros\centrosdb\trabajos\_centros - HeidiSQL 11.1.0.6116

Archivo Editar Buscar Consulta Herramientas Ira Ayuda

Host: centros.ckcro2r2me.us-east-1.rds.amazonaws.com Base de datos: centrosdb Tabla: trabajos\_centros Datos Consulta

Filtro de bases de datos: Filtro de tablas

**Centros**

- centrosdb 160,0 KiB
  - centros\_computacion 48,0 KiB
  - TablaPrueba 16,0 KiB
  - trabajos\_centros 48,0 KiB**
  - usuarios\_db 48,0 KiB
- information\_schema 224,0 KiB
- innodb
- mysql
- performance\_schema
- test

**trabajos\_centros**

Nombre: trabajos\_centros

Comentario:

Columnas:

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NULL	Rellenar con zeros	Predeterminado	Comentario
1	t_nombre_trabajo	CHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
2	t_cant_operaciones	INT	13	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'	
3	t_usuario	CHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sin valor predeterminado	
4	t_centro_asignado	CHAR	30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'vacio'	
5	t_estado	CHAR	30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'Sin Asignar'	
6	t_orden	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	

Ayuda Descartar Guardar

Filtro: Expresión regular

```

37 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' ORDER BY ORDINAL_POSITION;
38 SHOW INDEXES FROM `trabajos_centros` FROM `centrosdb`;
39 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' AND REFERENCED_TABLE_NAME IS NOT NULL;
40 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='centrosdb' AND TABLE_NAME='trabajos_centros' AND REFERENCED_TABLE_NAME IS NOT NULL;
41 SHOW CREATE TABLE `centrosdb`.`trabajos_centros`;
  
```

Conectado: 00:06 h MariaDB 10.4.8 Activo durante: 3 días, 17:02 Hora del servidor: 1 Preparado.

### 7.3. Anexo III: Video Explicación Front-End

Nota: Procedimiento/Guía interna (no formal) del equipo de trabajo.

Video explicativo desarrollo y componentes front-end, en YouTube (modo oculto):  
[https://youtu.be/Gd\\_B4XfCrL0](https://youtu.be/Gd_B4XfCrL0)