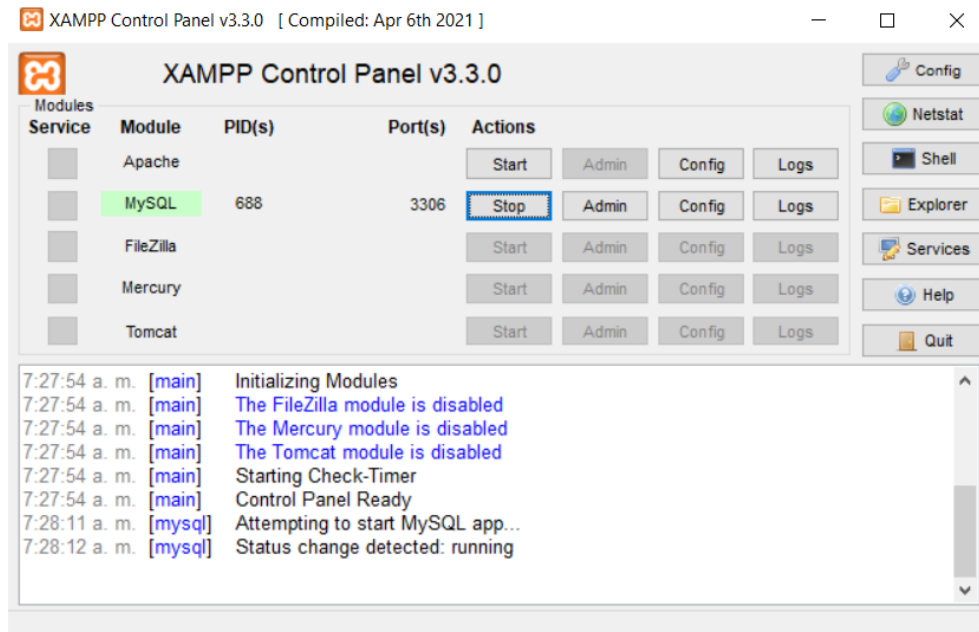
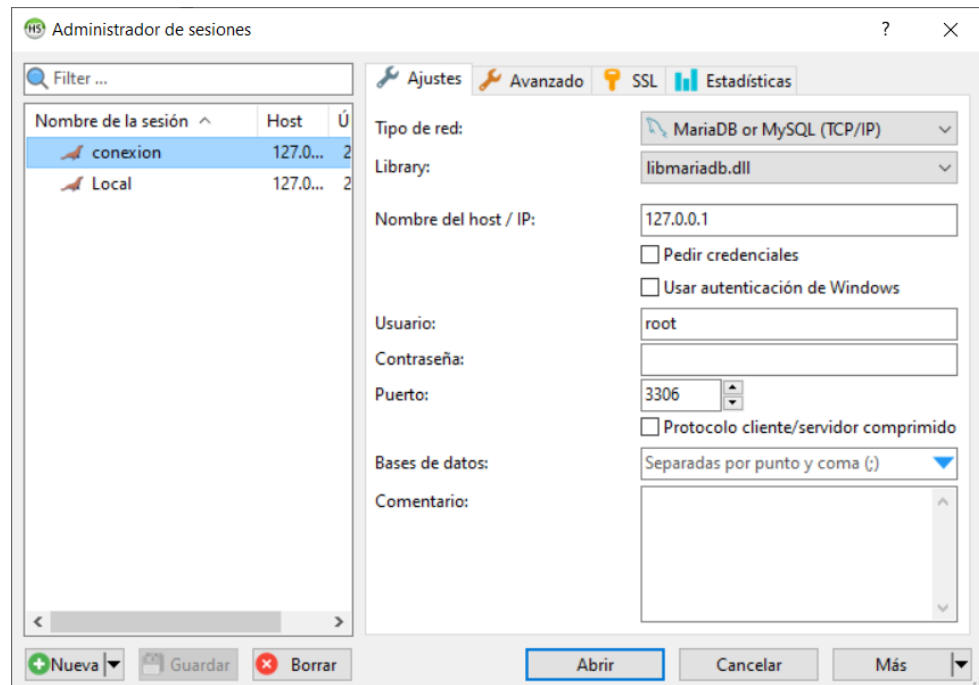


## Guía rápida de cómo iniciar el proyecto de la gestión de vacaciones

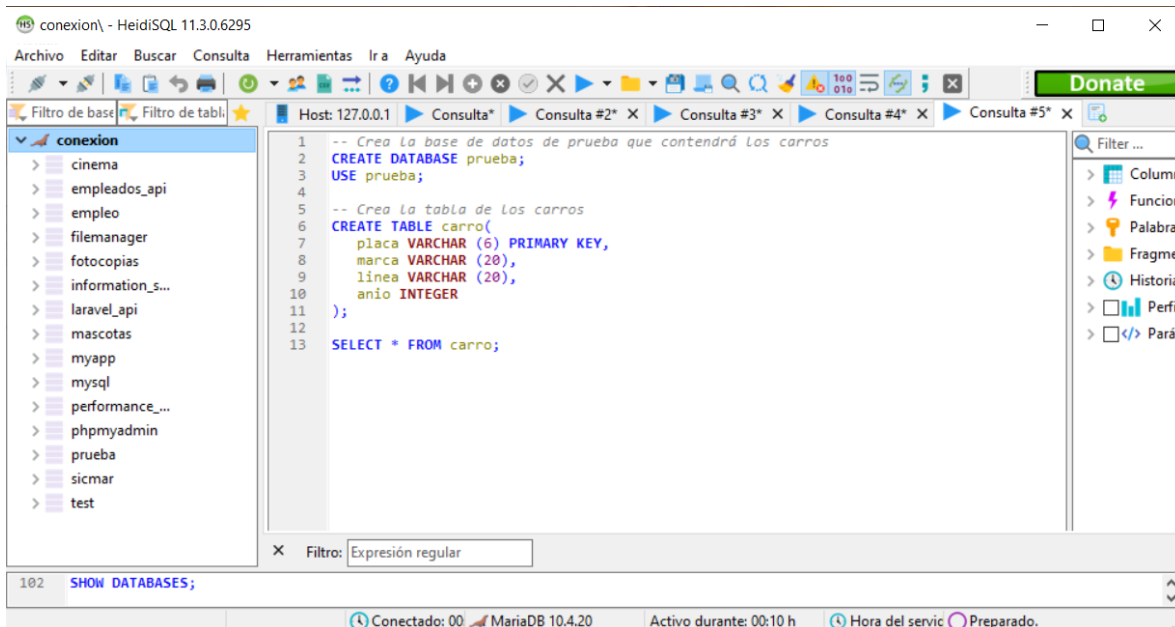
Después de tener Xampp instalado y en este caso HeidiSQL que nos permite conectarnos al servidor de base de datos MySQL, procedemos a arrancar MySQL desde Xampp, como se muestra a continuación.



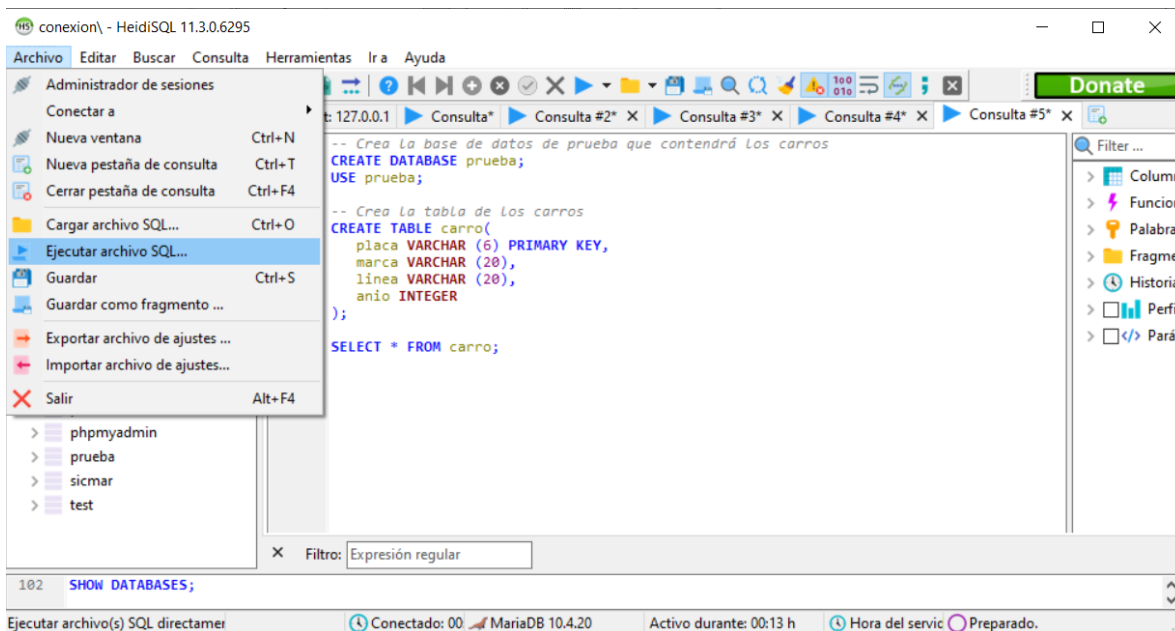
Posterior a esto procedemos a iniciar Heidi desde donde nos vamos a poder conectar a MySQL, en este toca tener en cuenta las credenciales de inicio de sesión en el servidor.



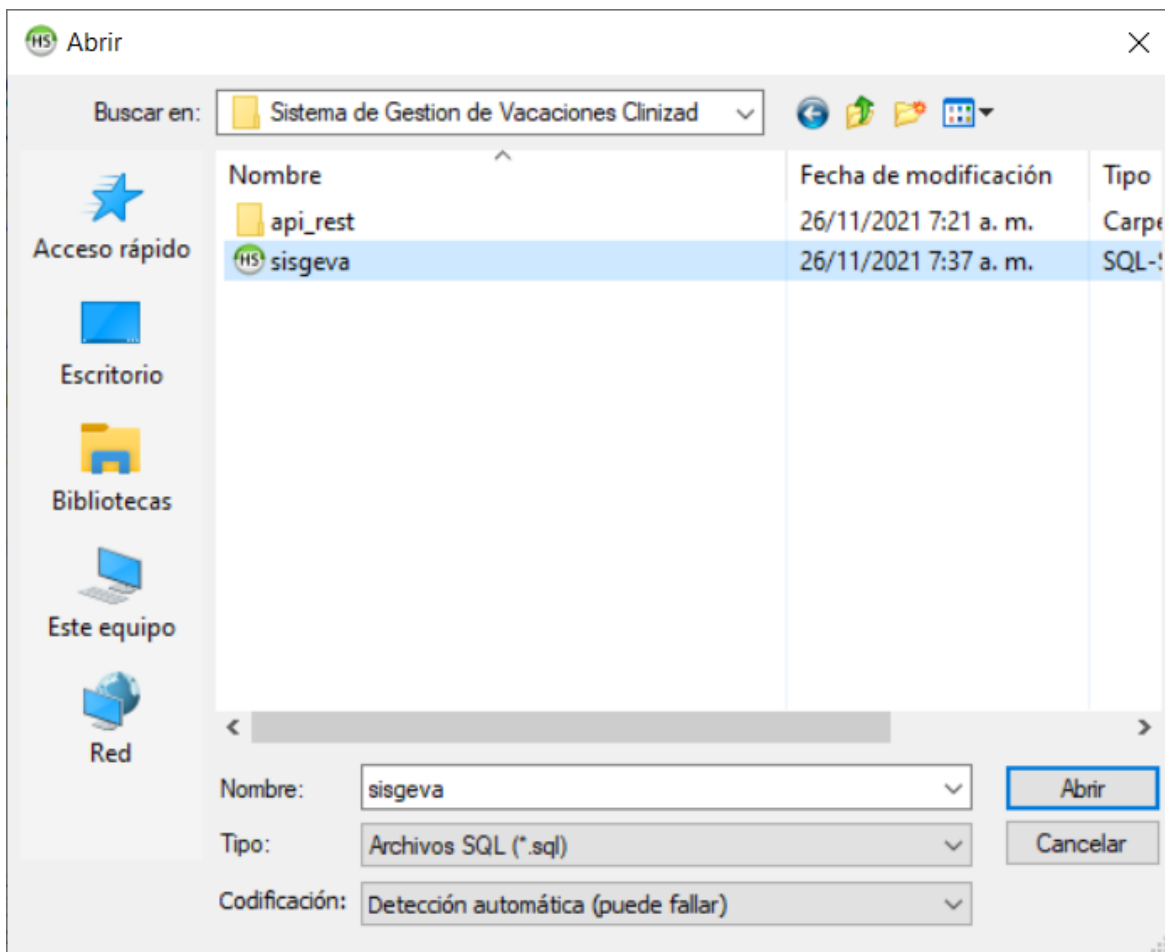
En este caso se ha creado una conexión la cual se conecta a MySQL, el nombre del host es localhost (127.0.0.1), el usuario será root y en este caso no tiene contraseña, su puerto será el de MySQL que viene por defecto (3306) y finalmente le damos en abrir.



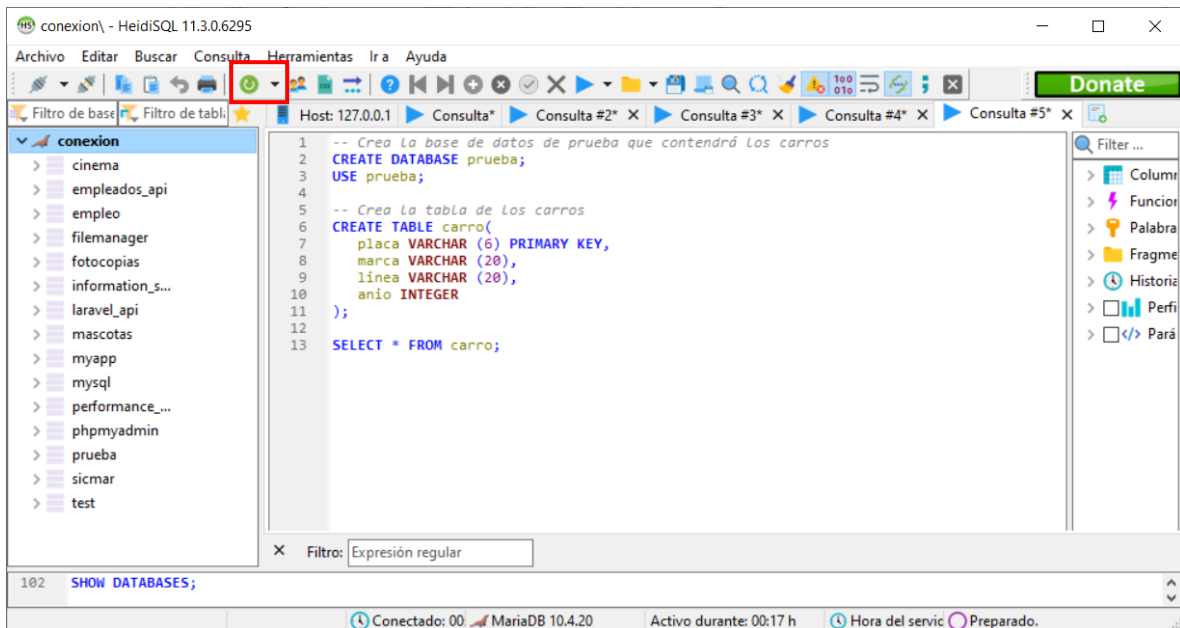
Esta es la interfaz inicial de HeidiSQL, en esta parte nos dirigimos a archivo y le damos en ejecutar archivo SQL, como se muestra a continuación.



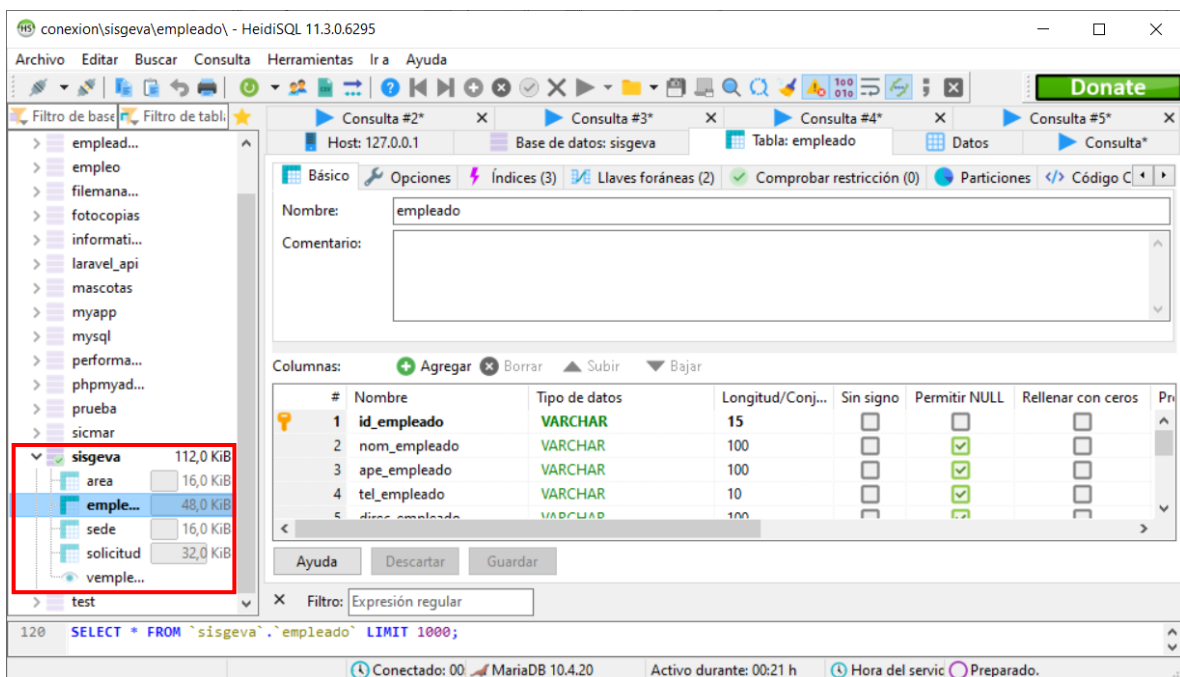
Después de esto se nos abre la siguiente ventana, en la cual tenemos que seleccionar el archivo **sisgeva.sql**, la cual esta en la carpeta raíz del proyecto, como se muestra a continuación.



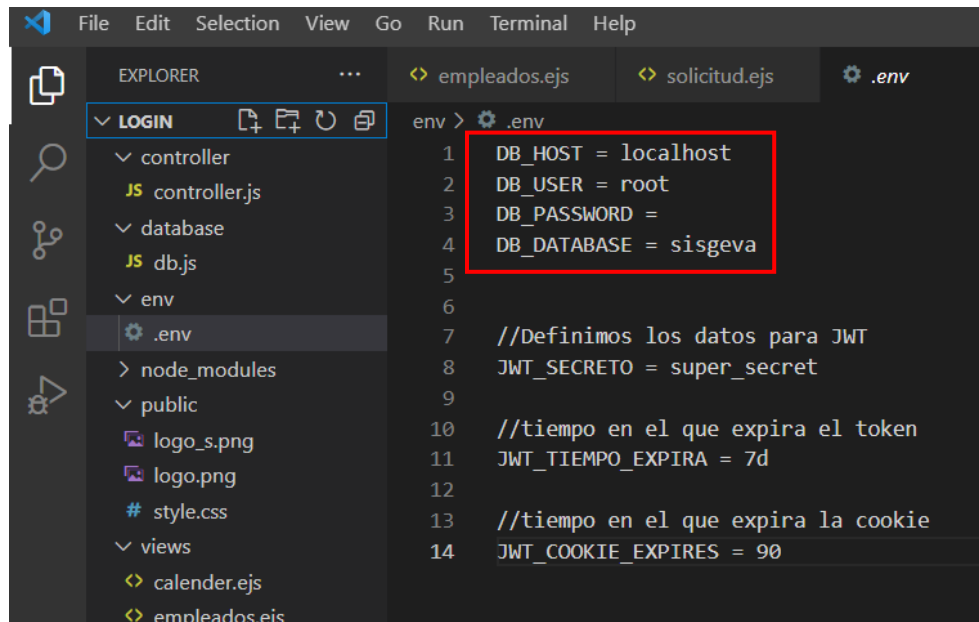
Después de esto le damos en abrir y se ejecutará el script con la creación y inserción de los datos de prueba con los que cuenta este script. Posterior a esto seleccionamos la conexión y le damos en recargar, como se muestra a continuación.



De este modo ya hemos importado la base de datos del proyecto, como se muestra a continuación.

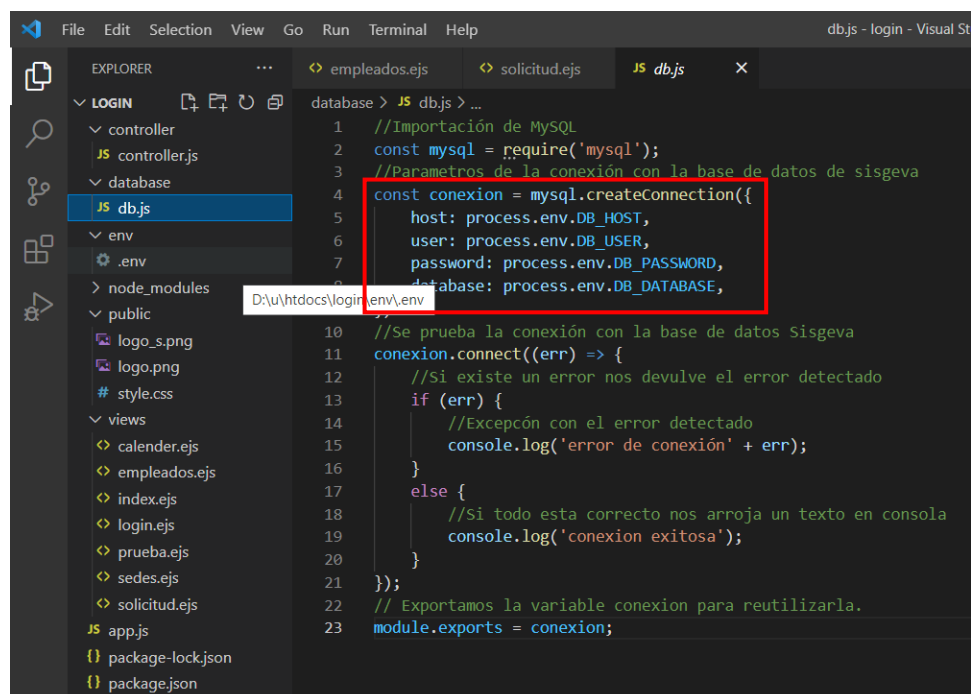


Posterior a esto pasamos al archivo **.env**, el cual se encuentra en la carpeta **env** y valiéndonos de **Visual Studio Code**, como se muestra a continuación.

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project structure with folders like LOGIN, controller, database, env, node\_modules, and public. The file db.js is selected under the database folder. The main editor window displays the content of the .env file. A red rectangle highlights the database connection configuration lines: DB\_HOST = localhost, DB\_USER = root, DB\_PASSWORD =, and DB\_DATABASE = sisgeva. Other lines in the file include JWT\_SECRET = super\_secret and JWT\_COOKIE\_EXPIRES = 90.

```
1 DB_HOST = localhost
2 DB_USER = root
3 DB_PASSWORD =
4 DB_DATABASE = sisgeva
5
6
7 //Definimos los datos para JWT
8 JWT_SECRETO = super_secret
9
10 //tiempo en el que expira el token
11 JWT_TIEMPO_EXPIRA = 7d
12
13 //tiempo en el que expira la cookie
14 JWT_COOKIE_EXPIRES = 90
```

En este archivo tenemos que asociar los datos de la conexión con la base de datos, en este caso el host será **localhost** (127.0.0.1), el usuario será **root**, el **password** lo dejamos vacío porque en este caso no se le asignó contraseña a este usuario y finalmente le digitamos el nombre de la base de datos, en este caso será **sisgeva**, de esta forma ya hemos asociado nuestra base de datos con el proyecto. Posteriormente a esto pasamos a asociar estos datos anteriores en el archivo **db.js**, el cual se encuentra en la carpeta **database**, como se muestra a continuación.

A screenshot of the Visual Studio Code editor showing the db.js file. The Explorer sidebar shows the database folder selected, with db.js highlighted. The main editor window shows the code for db.js. A red rectangle highlights the mysql.createConnection function call, which uses process.env variables for host, user, password, and database. A tooltip for process.env.DB\_PASSWORD is visible, showing its path as D:\u\htdocs\login\env\env.

```
1 //Importación de MySQL
2 const mysql = require('mysql');
3 //Parametros de la conexión con la base de datos de sisgeva
4 const conexion = mysql.createConnection({
5   host: process.env.DB_HOST,
6   user: process.env.DB_USER,
7   password: process.env.DB_PASSWORD,
8   database: process.env.DB_DATABASE,
9 });
10 //Se prueba la conexión con la base de datos Sisgeva
11 conexion.connect((err) => {
12   //Si existe un error nos devuelve el error detectado
13   if (err) {
14     //Excepción con el error detectado
15     console.log('error de conexión' + err);
16   }
17   else {
18     //Si todo esta correcto nos arroja un texto en consola
19     console.log('conexion exitosa');
20   }
21 });
22 // Exportamos la variable conexion para reutilizarla.
23 module.exports = conexion;
```

Igualmente, que en caso anterior asociamos los datos de conexión como esta en la figura y procedemos a ejecutar el archivo **app.js** desde una consola de Visual Studio dándole con el comando **ctrl + ñ** y ejecutamos el comando **node app** como se muestra a continuación.

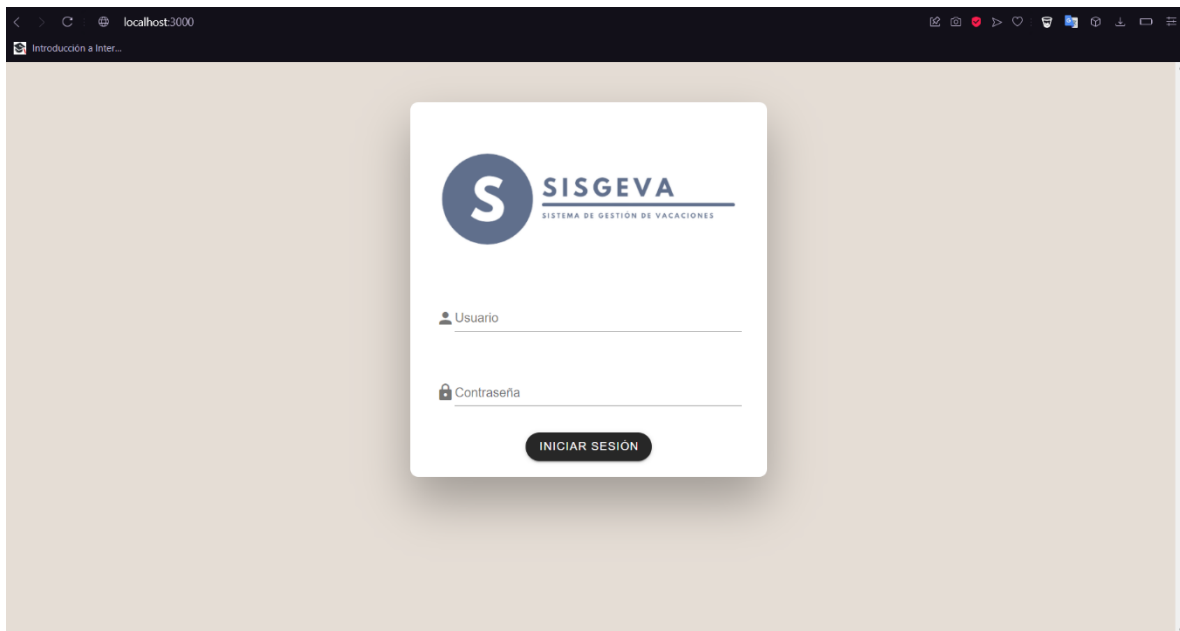
```
57 //Función para mostrar una sola área de Clinizad con su código
58 app.get('/api/area/:id_area',(req,res)=>{
59 //Consulta a la base de datos usando la sentencia sql que viene por parámetro con el id del área
60 conexion.query('SELECT * FROM area WHERE id_area = ?', [req.params.id_area], (error, fila)=>{
61 //Si existe un error nos devuelve el error detectado
62 if(error){
63 //Excepción con el error detectado
64 throw error;
65 }else{
66 //Si todo está correcto nos arroja el área con el id buscado

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL**

```
PS C:\Users\JHORLEN\Desktop\Sistema de Gestion de Vacaciones Clinizad\api_rest> node app
El servidor esta corriendo correctamente en el puerto: 3000
Conexión exitosa con la base de datos.
```

Al darle enter este nos arroja un mensaje en consola el cual nos indica que el servidor está corriendo e indicando porque puerto lo está haciendo, en este caso por el puerto **3000** y de la misma forma nos dice que la conexión con la base de datos esta correcta, de este modo ya hemos ejecutado el proyecto del sistema de vacaciones de Clinizad, para visualizarlo nos dirigimos a un navegador y digitamos **localhost:3000** en la barra de dirección, lo cual nos arroja el **login** de la aplicación.



Para iniciar sesión tenemos dos roles, administrador y no administrador, en este caso se ha iniciado sesión con un **administrador** y nos arroja la siguiente interfaz.

