

MANUAL TÉCNICO

PARADIGMA DE PROGRAMACIÓN

El paradigma de programación utilizado en este proyecto fue “Programación del lado del Cliente”, ya que se espera que el cliente reciba la mejor experiencia al utilizar esta página web.

REQUERIMIENTOS O FUNCIONALIDADES ESPECIFICAS

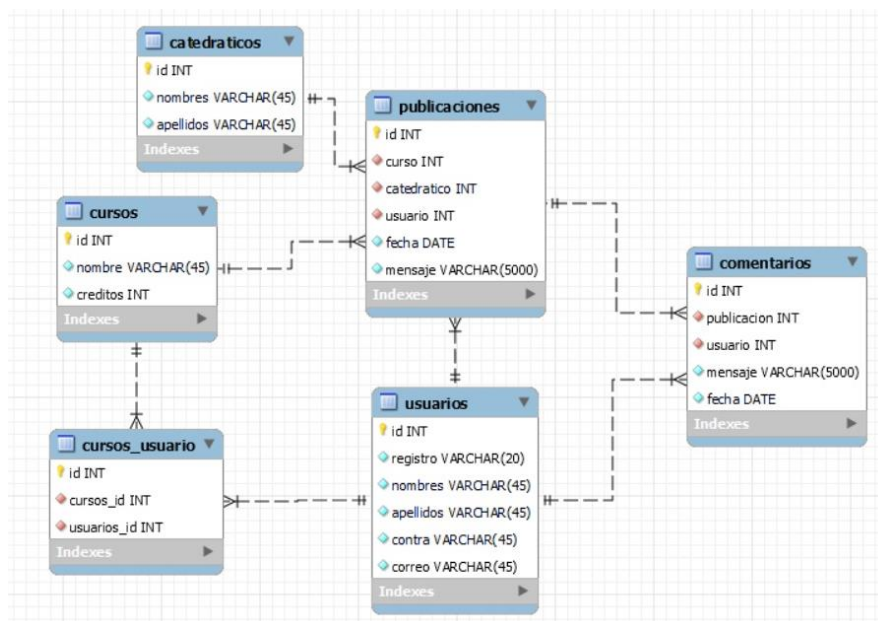
Para la creación de este proyecto se utilizo MySQL para la creación y administración de la base de datos. Para la creación del sitio web se utilizo Node Js y el lenguaje Javascript. Se utilizó React como Framework de la parte Frontend dirigida al usuario.

BACKEND

BASE DE DATOS

En este Proyecto se utiliza una base de datos Relacional en la que se almacenan los datos de los usuarios registrados, publicaciones y comentarios, cursos y catedráticos que los imparten, con el propósito de hacer más dinámico el manejo de la información.

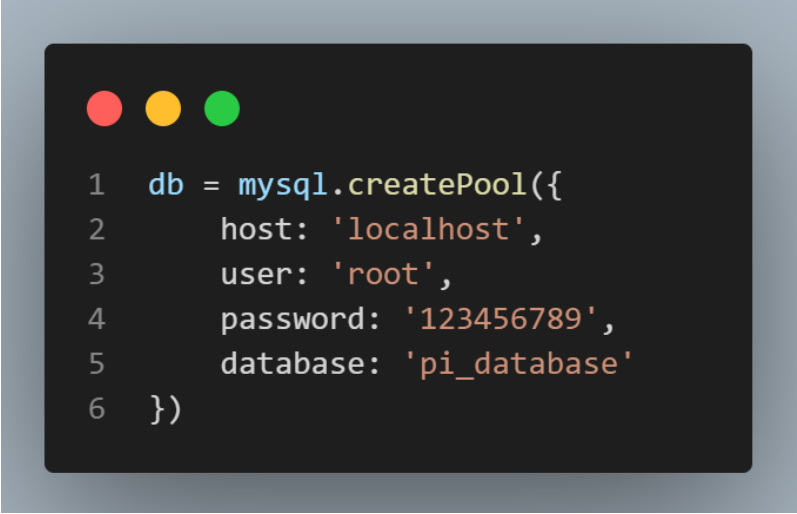
DIAGRAMA RELACIONAL DE LA BASE DE DATOS:



FUNCIONES IMPORTANTES:

CONEXIÓN CON LA BASE DE DATOS:

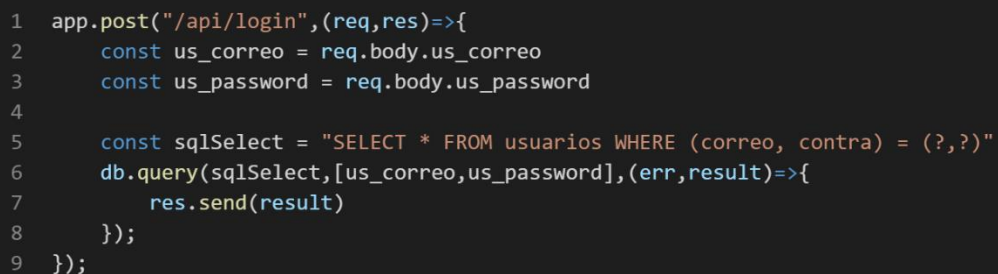
Con el siguiente código creamos la conexión con la base de datos local "pi_database".

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains a JavaScript code snippet for creating a MySQL connection pool.

```
1 db = mysql.createPool({
2   host: 'localhost',
3   user: 'root',
4   password: '123456789',
5   database: 'pi_database'
6 })
```

LOGIN:

El siguiente Código es de un Post Request que obtiene la información ingresada por el usuario en la ruta /api/login y la compara con los almacenados en la base de datos. Luego de comparar devuelve un resultado si los datos coinciden con algún registro dentro de la base de datos o devuelve un error si ningún dato coincide. Se utilizó código parecido para la opción de mostrar publicaciones.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains a JavaScript code snippet for a POST request handler for the login endpoint.

```
1 app.post("/api/login", (req, res) => {
2   const us_correo = req.body.us_correo
3   const us_password = req.body.us_password
4
5   const sqlSelect = "SELECT * FROM usuarios WHERE (correo, contra) = (?, ?)"
6   db.query(sqlSelect, [us_correo, us_password], (err, result) => {
7     res.send(result)
8   });
9 });
```

REGISTER:

El siguiente Código es de un Post Request que obtiene la información ingresada por el usuario en la ruta /api/register y la almacena en la base de datos como un nuevo usuario. Se utilizó un código parecido para la opción de crear publicación y crear comentarios.

```
1 app.post("/api/register",(req,res)=>{
2   const us_carnet = req.body.us_carnet
3   const us_nombres = req.body.us_nombres
4   const us_apellidos = req.body.us_apellidos
5   const us_password = req.body.us_password
6   const us_correo = req.body.us_correo
7
8   const sqlInsert = "INSERT INTO usuarios (registro, nombres, apellidos, contra, correo) VALUES (?, ?, ?, ?, ?);"
9   db.query(sqlInsert, [us_carnet, us_nombres, us_apellidos, us_password, us_correo], (err,result)=>{
10
11   });
12 });
```


GET:

Este código muestra una consulta en la que se solicitarán los nombres de todos los catedráticos para mostrarlos en alguna de las partes de la página. Se utilizó un código parecido para crear las opciones de mostrar cursos.

```
1 app.get("/api/todosCat",(req,res)=>{
2   const sqlSelect = "SELECT * FROM catedraticos"
3   db.query(sqlSelect,(err,result)=>{
4     res.send(result)
5   });
6 });
```

LISTEN:

Este código genera una comunicación con el puerto 3001.




```
1 app.listen(3001,() =>{
2   console.log("running in port 3001");
3 });
```

FRONTEND (CLIENTE)

API GET

El siguiente código representa la API que se encarga de obtener los datos proporcionados por el usuario para luego ser utilizados en el backend en donde se utilizará para ser validaciones o para almacenar los datos.



```
1 Axios.get("http://localhost:3001/api/todosCat").then((res)=>{
2   var data = res.data;
3   for(var i=0;i<data.length;i++){
4     catedraticos.push({id:data[i].id,nombre: data[i].nombres+ " "+data[i].apellidos});
5   }
6 });
```

API POST

El siguiente código representa la API que se encarga de mostrar cierta información al usuario, en el caso de este código se muestra la información de una publicación recientemente creada. Se utilizó un código similar para mostrar los comentarios creados en las publicaciones.

```
1  Axios.post("http://localhost:3001/api/publicar",{
2      us_usuario: usuario,
3      us_catedratico: cat,
4      us_titulo: titulo,
5      us_mensaje: mensaje,
6      us_fecha: date,
7      us_curso: curso,
8  }).then(()=>{
9      alert("se ha creado la publicacion: "+titulo);
10  })
```

HTML

Este código representa la forma en que los elementos en las diferentes pantallas, que se muestran al usuario, se dividen. Se utilizó este formato para mostrar las publicaciones y comentarios y cada uno de sus elementos.

```
1  <div className="form">
2      <label className="guia"> Catedratico: </label>
3      <DropDownList
4          dataKey="id"
5          textField="nombre"
6          onChange={(seleccion) => {setCat(seleccion.id)}}
7          className="dropD"
8          defaultValue=""
9          data={catedraticos}/>
10 </div>
```

CSS

Se utilizó CSS para dar formato y estilo a los diferentes elementos dentro de la página web y así los usuarios pudieran tener una interfaz clara u agradable.

