



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Escenario:

La institución educativa AditaEdu tiene un servidor web que aloja el dominio adita.com, siendo desarrollada la página web con JavaScript, y para el almacenamiento de los datos, MySQL. Primero se presenta el código fuente completo del desarrollo web con los datos del módulo académico. Luego, se presenta el paso a paso del desarrollo web, del backend como del frontend.

Desarrollo web local

Backend

databaseCon.js

```
JS databaseCon.js X
src > config > JS databaseCon.js > <unknown> > module.exports
1  const mysql = require('mysql');
2
3  module.exports = () => [
4    return mysql.createConnection({
5      host: "localhost",
6      user: 'root',
7      password: '11235Dist',
8      database: 'espol',
9    });
10   ]
11
```

estudiante.controller.js

```
JS estudiante.controller.js ●
src > controller > JS estudiante.controller.js > [e] getEstudiante
1  const dbConnection = require('../config/databaseCon');
2  const connection = dbConnection();
3  let getEstudiante = async (req,res)=>[
4    await connection.query("select * from estudiantes", (err,result)=>{
5      if (result)
6        res.send(result);
7      else
8        res.status(500).send(err);
9    });
10   ]
11
12   module.exports = {
13     getEstudiante
14   }
15
16
```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

estudiante.js

```
JS estudiante.js X
src > routes > JS estudiante.js > ...
1  const {Router} = require("express");
2  const router = Router();
3  const{ getEstudiante, addEstudiante } =  require("../controller/estudiante.controller");
4
5  router.get('/', getEstudiante);
6  //router.post("/",addEstudiante);
7
8  module.exports = router;
9
```

app.js

```
JS app.js X
src > JS app.js > ...
1  const express = require('express');
2  |
3  const app = express();
4  const cors= require('cors')
5  // body-parser
6  app.use(cors())
7  app.use(express.json());
8  app.use(express.urlencoded({extended:false}));
9
10 app.set('port', process.env.PORT | 8080);
11
12 const{ getEstudiante, addEstudiante } =  require("../controller/estudiante.controller");
13
14 app.get('/', getEstudiante);
15
16 app.use("/api/estudiante",require("./routes/estudiante") );
17
18
19 app.listen(app.get('port'));
20 console.log(`Server on port ${app.get('port')}`)
21
```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Frontend

ListEstudiantes.js

```
JS ListEstudiantes.js •
frotnend > src > Estudiantes > JS ListEstudiantes.js > ListEstudiantes > render
  1 import React, {Component} from 'react'
  2 import axios from 'axios'
  3 import './boton.css'
  4
  5 export default class ListEstudiantes extends Component{
  6   state={
  7     users:[],
  8     id_estudiante:'',
  9     nombre_est:'',
10     apellidos_est:'',
11     edad:'',
12     id_carrera:''
13   }
14   async componentDidMount(){
15     const res = await axios.get('http://localhost:8080');
16     this.setState({users:res.data});
17   }
18   render(){
19     const alerta=(msj)=>{
20       alert(msj);
21     }
22     return(
23       <center>
24         <table border="1">
25           <tbody>
26             <tr>
27               <td width="180"><h3 align="center">ID del estudiante</h3></td>
28               <td width="250"><h3 align="center">Nombre del estudiante</h3></td>
29               <td width="200"><h3 align="center">Código de la carrera</h3></td>
30               <td width="80"><h3 align="center">Edad</h3></td>
31             </tr>
32             <td align="center" bgcolor="#E5E5E5">
```

```
33               {this.state.users.map( (user)=>(
34                 <tr key={user.nombre_est}>
35                   <td>{user.id_estudiante}</td>
36                 </tr>
37               )
38             )
39           )
40         </td>
41         <td bgcolor="#FFEA87E">
42           {this.state.users.map( (user)=>(
43             <tr key={user.nombre_est}>
44               <td>{user.nombre_est+" "+user.apellidos_est}</td>
45             </tr>
46           )
47         )
48       )
49     </td>
50     <td align="center" bgcolor="#E5E5E5">
51       {this.state.users.map( (user)=>(
52         <tr key={user.nombre_est}>
53           <td>{user.id_carrera}</td>
54         </tr>
55       )
56     )
57   )
58 </td>
59 <td align="center" bgcolor="#FFEA87E">
60   {this.state.users.map( (user)=>(
61     <tr key={user.nombre_est}>
62       <td>{user.edad}</td>
63     </tr>
```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

```

64      )
65      )
66      }
67      </td>
68      </tbody>
69  </table>
70  <br><br>
71  <br><br>
72  <br><br>
73  <div align="center">
74  <button class="btn" onClick={()=>alerta("Existen "+this.state.users.length+" estudiantes en la
75  </div>
76  </center>
77  }
78  }
79 }

```

Header.js

```

JS Header.js ×
frotnend > src > Estudiantes > JS Header.js > [o] Header
1 import React from 'react'
2
3 const Header = ({title}) => [
4
5   return (
6     <>
7       <h1 align="center">{title}</h1>
8
9     </>
10  )
11 ]
12
13 export default Header
14

```

EstudianteLayout.js

```

JS EstudianteLayout.js •
frotnend > src > Estudiantes > JS EstudianteLayout.js > ...
1 import "./boton.css"
2 import Header from "./Header"
3 import ListEstudiantes from "./ListEstudiantes"
4 import React from 'react'
5
6
7 const EstudianteLayout = () => {
8
9   return (
10
11   <React.Fragment>
12
13     <Header title="Listado de Estudiantes"/>
14
15     <ListEstudiantes/>
16
17   </React.Fragment>
18
19 }
20
21 export default EstudianteLayout
22

```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

index.js

```
JS index.js ●
frtend > src > JS index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import EstudianteLayout from './Estudiantes/EstudianteLayout.js';
5 import reportWebVitals from './reportWebVitals';
6
7 ReactDOM.render(
8
9   <React.StrictMode>
10    <EstudianteLayout/>
11
12  </React.StrictMode>,
13  document.getElementById('root')
14 );
15
16 // If you want to start measuring performance in your app, pass a function
17 // to log results (for example: reportWebVitals(console.log))
18 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
19 reportWebVitals();
```

La hoja de estilos (CSS) usada se muestra a continuación:

```
# boton.css ●
frtend > src > Estudiantes > # boton.css > ...
1 .btn {
2   background: #eb94d0;
3   /* Crear degradados */
4   background-image: -webkit-linear-gradient(top, #eb94d0, #2079b0);
5   background-image: -moz-linear-gradient(top, #eb94d0, #2079b0);
6   background-image: -ms-linear-gradient(top, #eb94d0, #2079b0);
7   background-image: -o-linear-gradient(top, #eb94d0, #2079b0);
8   background-image: linear-gradient(to bottom, #eb94d0, #2079b0);
9   /* Dar bordes curvados a btn */
10  -webkit-border-radius: 28;
11  -moz-border-radius: 28;
12  border-radius: 28px;
13  text-shadow: 3px 2px 1px #9daef5;
14  -webkit-box-shadow: 6px 5px 24px #666666;
15  -moz-box-shadow: 6px 5px 24px #666666;
16  box-shadow: 6px 5px 24px #666666;
17  font-family: Arial;
18  color: #fafafa;
19  font-size: 27px;
20  padding: 19px;
21  text-decoration: none;
22 }
23
```



[NT4] NOTA TÉCNICA N°. 4
DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

La base de datos del “Sistema Académico” se muestra a continuación:

	id_estudiante	nombre_est	apellidos_est	edad	id_carrera
▶	19870981	Carolina	Ormaza	18	INGMCT
	19870982	José	Patiño	21	INGMCT
	19870983	Stalin	Vera	21	INGELCT
	19870984	Andrés	Patiño	18	INGCVL
	19870985	Itati	Orellana	22	INGCVL
	19870986	Ammy	Bastidas	21	ADMEMP
	19870987	Amanda	Worms	25	ADMEMP



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Paso 1. Conexión con la base de datos.

1. Se debe usar al módulo de MYSQL, mediante la siguiente línea de código, presente en el archivo "databaseCon.js":

```
1 const mysql = require('mysql');
2
```

2. Se crea la conexión especificando la base de datos (`mysql.createConnection()`), el usuario, host contraseña de nuestra base de datos creada. Y se lo exporta como módulo, para luego poder usarlo en cualquier parte de nuestro programa, en este caso en el archivo `estudiante.controller.js`.

```
3 module.exports = () => {
4   return mysql.createConnection({
5     host: "localhost",
6     user: 'root',
7     password: '112    ',
8     database: 'espol1',
9   });
10 }
11
```

Paso 2. Consulta a la base de datos

1. Se importa el módulo creado en el archivo `databaseCon.js` y se llama al método `dbConnection()` que nos ayudará instanciando al objeto `connect`, para luego realizar la consulta.

```
1 const dbConnection = require('../config/databaseCon');
2 const connection = dbConnection();
```

2. Se crea la función asincrónica `getEstudiante()`, se especifica que se obtendrán todos los registros de la tabla creada utilizando "*" y se valida que exista una respuesta, caso contrario se envía un mensaje de error.

```
const dbConnection = require('../config/databaseCon');
const connection = dbConnection();
let getEstudiante = async (req,res)=>{
  await connection.query("select * from estudiantes", (err,result)=>[
    if (result)
      res.send(result);
    else
      res.status(500).send(err);
  ]);
}
```

3. Se exporta el módulo creado con la función `getEstudiante()`.

```
module.exports = {
  getEstudiante
}
```

Paso 3. Mostrar la página del backend

4. Se configura el puerto en el cual se mostrará el backend, se obtiene la función `getEstudiante()` y se hace uso de la función.

```
app.set('port', process.env.PORT || 8080);
const{getEstudiante} = require("../src/controller/estudiante.controller");
app.get('/', getEstudiante);
app.use("/api/estudiante",require("./routes/estudiante") );
```

5. Se publica el backend y se lo imprime un mensaje de confirmación por consola.

```
app.listen(app.get('port'));
console.log(`Server on port ${app.get('port')}`)
```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Paso 4. Desarrollo del código del Frontend.

1. Se crea la cabecera con el título de la página parametrizado. Los tags `<h1></h1>` indican un tamaño de letra utilizado para los títulos, debido a su altura. Con `export default` permitimos que otro archivo pueda usar este componente.

```
import React from 'react'

const Header = ({title}) => {

    return (
        <>
            <h1 align="center">{title}</h1>
        </>
    )
}

export default Header
```

2. Se declara la clase `ListEstudiantes` que extiende de `Component` y se la habilita para poder ser usada por otros archivos.

```
export default class ListEstudiantes extends Component{
```

3. Se crea la lista `users`, y se especifican los datos que guardará esta lista, tales como: id del estudiante, nombre del estudiante, apellidos, edad e id de la carrera.

```
state={
    users:[],
    id_estudiante:'',
    nombre_est:'',
    apellidos_est:'',
    edad:'',
    id_carrera:''
}
```

4. Se obtienen los datos del backend haciendo uso del URL de la página y se los almacena en la lista.

```
async componentDidMount(){
    const res = await axios.get('http://localhost:8080');
    this.setState({users:res.data});
}
```

5. Crear el mensaje de alerta parametrizado mediante `"msj"`.

```
render(){
    const alerta=(msj)=>{
        alert(msj);
    }
}
```

6. Comenzamos la creación de una tabla centrada (`<center>`) con un borde fino `border="1"`, se le da títulos a las columnas, se centra el texto de las celdas y ajustamos el ancho correspondiente a los pixeles especificados.

```
render(){
    const alerta=(msj)=>{
        alert(msj);
    }
    return(
        <center>
            <table border="1">
                <tbody>
                    <tr>
                        <td width="180"><h3 align="center">ID del estudiante</h3></td>
                        <td width="250"><h3 align="center">Nombre del estudiante</h3></td>
                        <td width="200"><h3 align="center">Código de la carrera</h3></td>
                        <td width="80"><h3 align="center">Edad</h3></td>
                    </tr>
                </tbody>
            </table>
        </center>
    )
}
```

7. Se crean filas con su contenido centrado y colores definidos, se recorre la lista como mapa de acuerdo al dato que se quiera obtener. Por ejemplo, en la primera fila se recorre el id de los estudiantes; luego, se realiza el mismo método para las otras columnas de la base de datos.



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

```
<td align="center" bgcolor="#E5E5E5">
  {this.state.users.map( (user)=>(
    <tr key={user.nombre_est}>
      | {user.id_estudiante}
    </tr>
  )
)
}
</td>
<td bgcolor="#FEA87E">
  {this.state.users.map( (user)=>(
    <tr key={user.nombre_est}>
      | {user.nombre_est} "+user.apellidos_est}
    </tr>
  )
)
}
</td>
<td align="center" bgcolor="#E5E5E5">
  {this.state.users.map( (user)=>(
    <tr key={user.nombre_est}>
      | {user.id_carrera}
    </tr>
  )
)
}
</td>
<td align="center" bgcolor="#FEA87E">
  {this.state.users.map( (user)=>(
    <tr key={user.nombre_est}>
      | {user.edad}
    </tr>
  )
)
}
```

8. Se cierran los tags de la tabla, se espacian tres líneas y se crea un botón asociado a la función de alerta mediante:

onClick={()=>alert("Existen "+this.state.users.length+" estudiantes en la lista")}

El botón posee formato (class="btn"), obtenido de un archivo .css, el cual será explicado posteriormente

```

        )
      )
    </td>
  </tbody>
</table>

<br><br>
<br><br>
<div align="center">
  <button class="btn" onClick={()=>alert("Existen "+this.state.users.length+
  " estudiantes en la lista")}>Cantidad de Estudiantes</button>
</div>
</center>
}
}
```

9. Desarrollar la cabecera parametrizada de la página, la cual constará de un título.

```
import React from 'react'
const Header = ({title}) => {
  return (
    <>
      <h1 align="center">{title}</h1>
    </>
  )
}
export default Header
```

10. Realizar las importaciones respectivas. Definir el contenido del Layout, para esto se utilizan las etiquetas correspondientes a los archivos trabajados con anterioridad.

<Header title="Listado de Estudiantes"/>: Sirve para definir el título de la página.

<ListEstudiantes/>: Contiene a la tabla con todos los registros presentes en la base de datos y al botón que muestra el mensaje de alerta con el número de estudiantes que han sido registrados.

<React.Fragment>: Un patrón común en React es que un componente devuelva múltiples elementos. Con Fragment se puede agrupar una lista de elementos hijos sin agregar nodos adicionales al DOM.

```
import Header from "./Header.js"
import ListEstudiantes from "./ListEstudiantes"
import React from 'react'

const EstudianteLayout = () => {

  return (
    <React.Fragment>
      <Header title="Listado de Estudiantes"/>
      <ListEstudiantes/>
    </React.Fragment>
  )
}
export default EstudianteLayout
```

11. Asegurarse que en el archivo "index.js" se encuentre referenciada la etiqueta <Estudiante.Layout>

```
ReactDOM.render(
  <React.StrictMode>
    <EstudianteLayout/>
  </React.StrictMode>,
  document.getElementById('root')
);
```



[NT4] NOTA TÉCNICA 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Se presenta el backend ingresando a localhost:8080, como se muestra:

```
[{"id_estudiante":19870981,"nombre_est":"Carolina","apellidos_est":"Ormaza","edad":18,"id_carrera":"INGMCT"}, {"id_estudiante":19870982,"nombre_est":"José","apellidos_est":"Patiño","edad":21,"id_carrera":"INGMCT"}, {"id_estudiante":19870983,"nombre_est":"Stalin","apellidos_est":"Vera","edad":21,"id_carrera":"INGELCT"}, {"id_estudiante":19870984,"nombre_est":"Andrés","apellidos_est":"Patiño","edad":18,"id_carrera":"INGCVL"}, {"id_estudiante":19870985,"nombre_est":"Itati","apellidos_est":"Orellana","edad":22,"id_carrera":"INGCVL"}, {"id_estudiante":19870986,"nombre_est":"Ammy","apellidos_est":"Bastidas","edad":21,"id_carrera":"ADMEMP"}, {"id_estudiante":19870987,"nombre_est":"Amanda","apellidos_est":"Worms","edad":25,"id_carrera":"ADMEMP"}]
```

Al ingresar a localhost:3000, se muestra el resultado del frontend:

Listado de Estudiantes

ID del estudiante	Nombre del estudiante	Código de la carrera	Edad
19870981	Carolina Ormaza	INGMCT	18
19870982	José Patiño	INGMCT	21
19870983	Stalin Vera	INGELCT	21
19870984	Andrés Patiño	INGCVL	18
19870985	Itati Orellana	INGCVL	22
19870986	Ammy Bastidas	ADMEMP	21
19870987	Amanda Worms	ADMEMP	25

Cantidad de Estudiantes



[NT4] NOTA TÉCNICA N°. 4
DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Al presionar el botón, se muestra el número de estudiantes en la tabla:

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000". A modal dialog box is displayed, containing the text "localhost:3000 dice" and "Existen 7 estudiantes en la lista". Below this, there is a table with four columns: "ID del estudiante", "Nombre del estudiante", "Código de la carrera", and "Edad". The table data is as follows:

ID del estudiante	Nombre del estudiante	Código de la carrera	Edad
19870981	Carolina Ormaza	INGMCT	18
19870982	José Patiño	INGMCT	21
19870983	Stalin Vera	INGELCT	21
19870984	Andrés Patiño	INGCVL	18
19870985	Itati Orellana	INGCVL	22
19870986	Ammy Bastidas	ADMEMP	21
19870987	Amanda Worms	ADMEMP	25

At the bottom of the modal, there is a blue button with the text "Cantidad de Estudiantes".



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Desarrollo web en la nube

Para el desarrollo de esta sección se necesitarán realizar ciertos cambios, tanto al backend como al frontend. Además, se utilizarán páginas que ofrecen servicio de hosting gratuito (Heroku y Netlify). Además, se alojará la base de datos remota en db4free.net. A continuación, se mostrarán los pasos y cambios que se deben realizar para lograr el despliegue remoto de nuestra página:

1. Registrarse en la página mostrada a continuación (<https://signup.heroku.com/>).

The screenshot shows the Heroku sign-up page. It features a purple header with the Heroku logo and a 'Sign up for free and experience Heroku today' call-to-action. Below the header, there are two main sections: 'Free account' and 'Your app platform'. The 'Free account' section describes it as a place to 'Create apps, connect databases and add-on services, and collaborate on your apps, for free.' The 'Your app platform' section describes it as a 'platform for apps, with app management & instant scaling, for development and production.' To the right of these sections is a registration form with fields for First name, Last name, Email address, Company name, and Role.

2. Establecer una contraseña para la cuenta.

The screenshot shows the 'Set your password' page from Heroku. The title is 'Set your password' and the sub-instruction is 'Create your password and log in to your Heroku account.' The page contains a form with fields for 'Create a new password' (New password and Password confirmation) and 'Confirm new password'. Below the form are 'Password requirements': 'Must be a minimum of 8 characters.', 'Must contain letters, numbers, and symbols.', and 'Passwords must match.' At the bottom is a blue 'SET PASSWORD AND LOG IN' button.



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

3. Aceptar todos los términos de servicio condicionadas por la página.

The screenshot shows a web browser window for the Heroku dashboard. The URL is dashboard.heroku.com/terms-of-service. At the top, there's a header with the Salesforce Platform logo and the Heroku logo. Below the header, there's a search bar labeled "Jump to Favorites, Apps, Pipelines, Spaces...". A note states: "use, in which case such provider's terms or use govern the use or purchase of the service. [Terms](#) apply to credit card customers of the Heroku Services." There are two sections: "Heroku Marketplace Providers" and "Italian Customers". Under "Italian Customers", there's a question "Are you domiciled in Italy?" followed by a toggle switch set to "No.". At the bottom right is a large blue "Accept" button with a hand icon pointing to it.

4. Seleccionar la opción “Create a new app”.

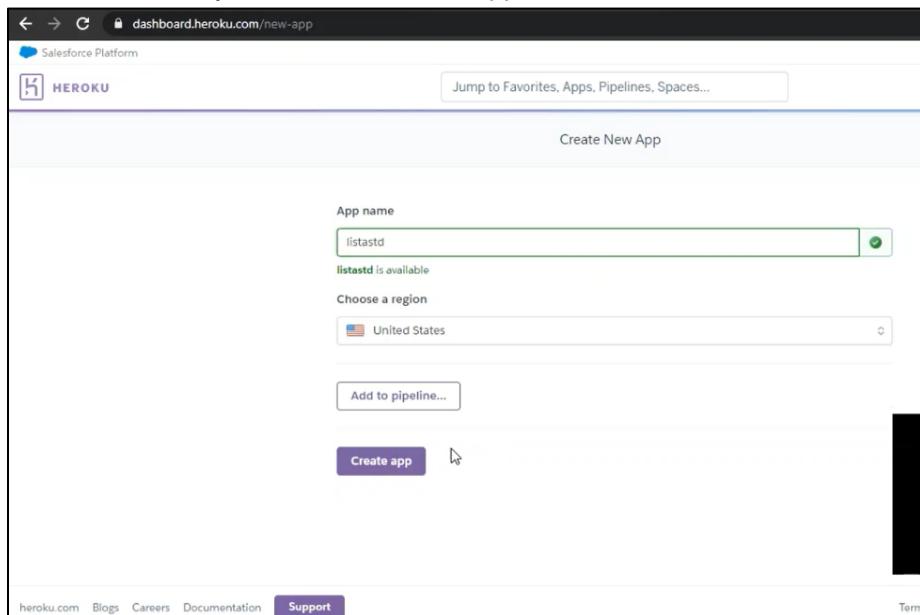
The screenshot shows a web browser window for the Heroku dashboard. The URL is dashboard.heroku.com/apps. The page has a purple header with the Heroku logo and "Personal". Below the header, there's a banner with the text "Welcome to Heroku" and "Now that your account has been set up, here's how to get started.". There are two main buttons: "Create a new app" (with a single user icon) and "Create a team" (with a group of users icon). Below these buttons, there's a link "Looking for help getting started with your language? Get started by reading one of our language guides in the Dev Center". At the bottom left is a link "https://dashboard.heroku.com/new-app".



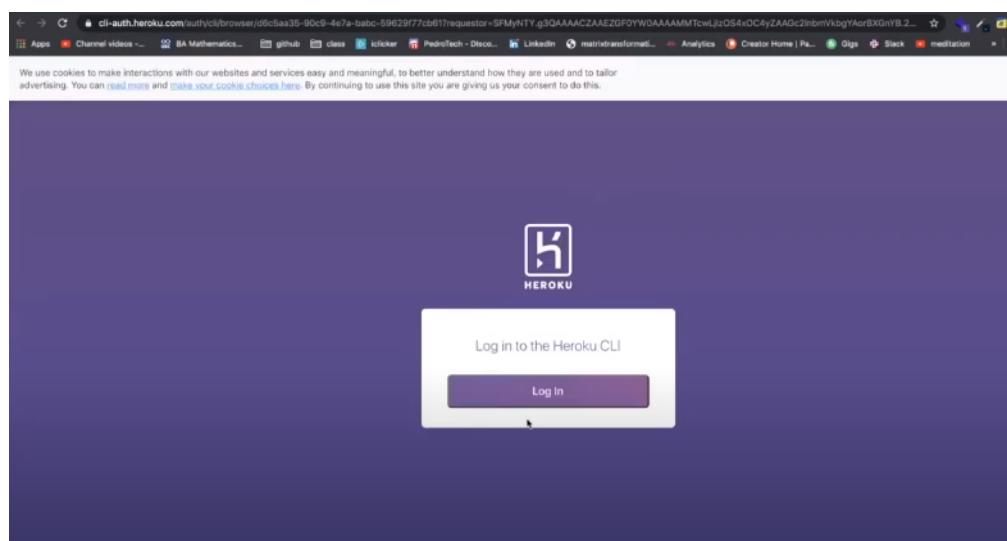
[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

5. Darle nombre a la aplicación a desarrollar y dar click en “Create app”.



6. Abrir la terminal y ubicarse en el directorio de la carpeta en la que se encuentre el backend e ingresar el siguiente comando para instalar el cliente de Heroku: `npm i -g heroku`
 7. Ingrese a la cuenta de Heroku desde la terminal de VSC, mediante el siguiente comando: `heroku login`
 8. Se abrirá automáticamente la siguiente página de login, ingresar los respectivos datos pedidos por la página:





[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

9. Ingresar las siguientes líneas para iniciar el repositorio vacío remoto en Heroku.

```
git init  
heroku git:remote -a listastds
```

10. Ejecutar los commandos ordinarios para subir archivos a un repositorio:

```
git add .  
git commit -am "make it better"  
git push heroku master
```

11. Asegurarse de que en el archivo “index.js”, la siguiente línea tenga al operador lógico OR (||), como se muestra en la siguiente figura:

```
app.set('port', process.env.PORT || 8082);
```

12. Asegurarse de que en el archivo “package.js” se especifique que el archivo de inicio es el “index.js”.

```
"scripts": {  
  "start": "node src/index.js"  
},
```

Ahora nos enfocaremos en el alojamiento de la base de datos remota.

13. Ingresar a <https://www.db4free.net/signup.php>, y digitar todos los datos requeridos por la página para el registro.

The screenshot shows the registration form for db4free.net. It includes fields for the database name (MySQL), user name, password, email, and a checkbox for accepting terms and conditions. There is also a donation button and a note about the service being for testing only.

db4free.net sólo proporciona una base de datos MySQL, pero no espacio en la web (no hay lugar para cargar ningún archivo)

Además:

- db4free.net es un servicio para pruebas, no para hosting. Las bases de datos que almacenan más de 200 MB de datos se borrarán a intervalos irregulares y sin notificación
- Por favor, elimine los datos que ya no necesita, o borre su cuenta si ya no es necesaria. Esto hace que sea más fácil de recuperar si se produce una caída del servidor.

Nombre de la base de datos MySQL: _____
Nombre de usuario MySQL: _____
Contraseña de usuario MySQL: _____
Verifique contraseña de usuario MySQL: _____

Correo electrónico: _____
Ingresá tu correo electrónico
Email addresses of certain domains are not allowed!

He leído las [condiciones de uso](#) y estoy de acuerdo con ellas.

El usuario de base de datos y el nombre de la base de datos puede contener letras minúsculas, números y guión bajo, y deben tener entre 6 y 16 caracteres de longitud. Usted no debe usar [palabras reservadas](#)!



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

14. Revisar el correo enviado por db4free.net, entrar al link de confirmación e ingresar con su usuario y contraseña.

Su registro de base de datos con db4free.net

* db4free.net es un servicio para pruebas, no para hosting. Las bases de datos que almacenan más de 200 MB de datos se borrarán a intervalos irregulares y sin notificación

* Por favor, elimine los datos que ya no necesite, o borre su cuenta si ya no es necesaria (<https://www.db4free.net/delete-account.php>). Esto hace que sea más fácil de recuperar si se produce una caída del servidor.

<https://www.db4free.net/confirm.php?create=aeaacbbf05b1dcee94aa50891e6accf7>

¿Puede ayudar a traducir el sitio web de db4free.net? Por favor, vaya a <https://www.db4free.net/translate.php>

Si no fue usted quien ha registrado una cuenta de base de datos en la página web de db4free, ¡por favor ignore este correo electrónico!

¡Esperamos que disfrute trabajar con su base de datos!

El equipo de db4free.net

<https://www.db4free.net>

[Responder](#) | [Reenviar](#)

15. Configurar la estructura de la tabla de la base de datos y guardar.



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

16. Ingresar cada uno de los registros que contendrá la tabla y presionar el botón “continuar” para ejecutar los comandos MySQL.

Si el proceso ha sido realizado correctamente, se debería tener una tabla como la siguiente:



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

17. Se realizarán los siguientes cambios al archivo “databaseCon.js”:

- El host ya no será “localhost”, será “db4free.net”.
- El usuario, contraseña y base de datos deben ser los registrados cuando se creó la cuenta en la página db4free.net.

```
const mysql = require('mysql');

module.exports = () => {
    return mysql.createConnection({
        host: "db4free.net",
        user: 'pstlistajospat1',
        password: '112_____',
        database: 'pstlista1',
    });
}
```

18. Repetir el paso 10.

Si es que todo se ha realizado correctamente, se deben mostrar los datos de nuestra tabla en la nube.

A screenshot of a web browser window displaying a JSON response from a Heroku application. The URL is 'liststds.herokuapp.com'. The JSON data lists several student records:

```
[{"id_estudiante":183,"nombre_est":"Stalin","apellidos_est":"Vera","edad":24,"id_carrera":"INGELCT"}, {"id_estudiante":1235,"nombre_est":"Carolina","apellidos_est":"Ormaza","edad":18,"id_carrera":"MCTG"}, {"id_estudiante":12322,"nombre_est":"Jose4","apellidos_est":"Patiño","edad":21,"id_carrera":"MCTG"}, {"id_estudiante":12348,"nombre_est":"Andrés","apellidos_est":"Patiño","edad":19,"id_carrera":"INGCVL"}, {"id_estudiante":18273,"nombre_est":"Ammy","apellidos_est":"Bast","edad":34,"id_carrera":"EHPGET"}]
```

Desde este momento, lo único que falta es mostrar nuestro frontend. Para lograrlo, seguir los siguientes pasos:

19. Asegurarse que en el archivo “ListEstudiantes.js” se esté obteniendo la información del backend desde la página donde se encuentra actualmente.

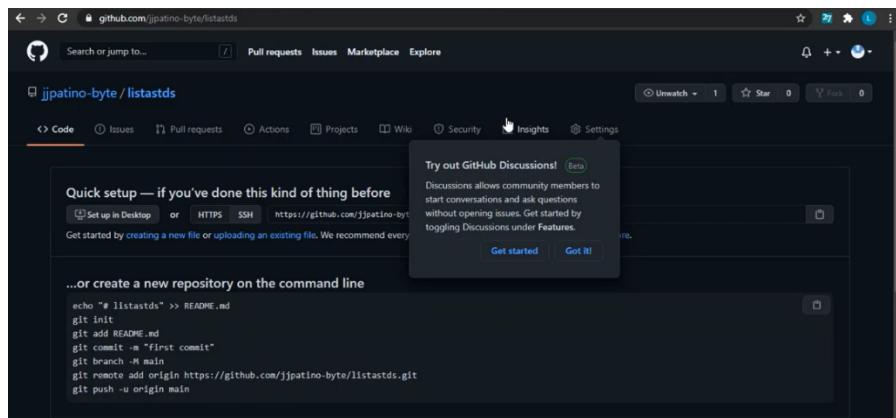
```
async componentDidMount(){
    const res = await axios.get('https://liststds.herokuapp.com/');
    this.setState({users:res.data});
}
```



[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

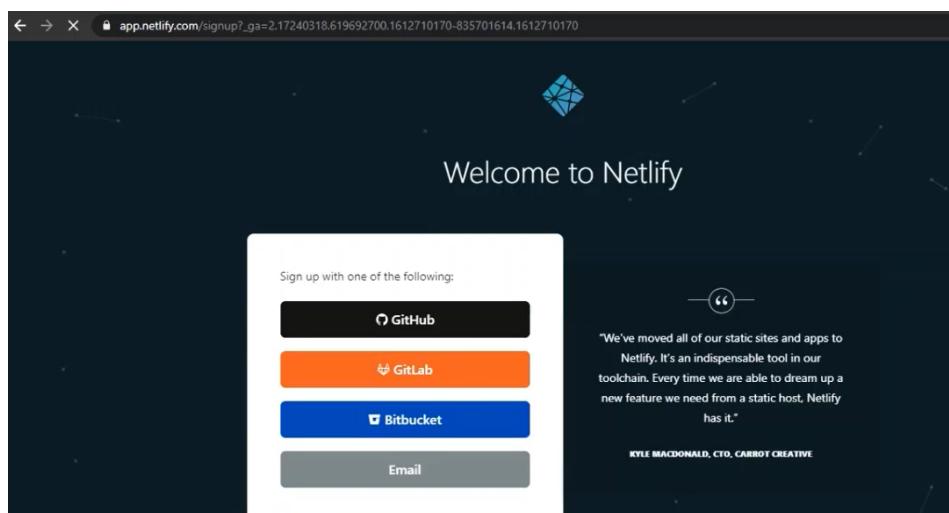
20. Crear un nuevo repositorio público en su cuenta de GitHub.



21. Abrir la terminal en el directorio de la carpeta donde se encuentra el frontend.
22. Ejecutar en la terminal los comandos que la página ordena.



23. Registrarse con GitHub en Netlify (<https://www.netlify.com/>), llenar los formularios y otorgarle todas las autorizaciones requeridas.

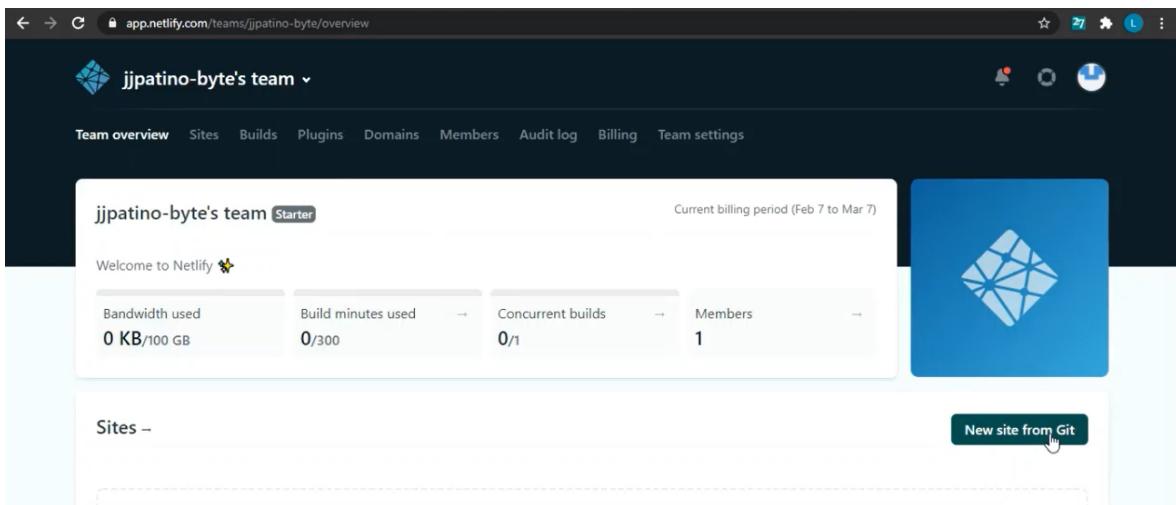




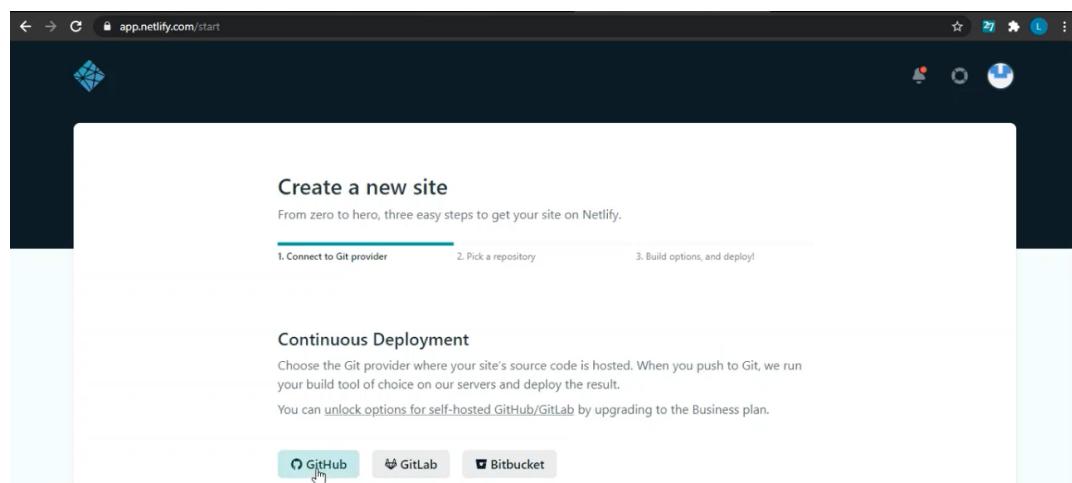
[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

24. Presionar el botón “New site from Git”.



25. Seguir los pasos de para obtener el sitio en Netlify.

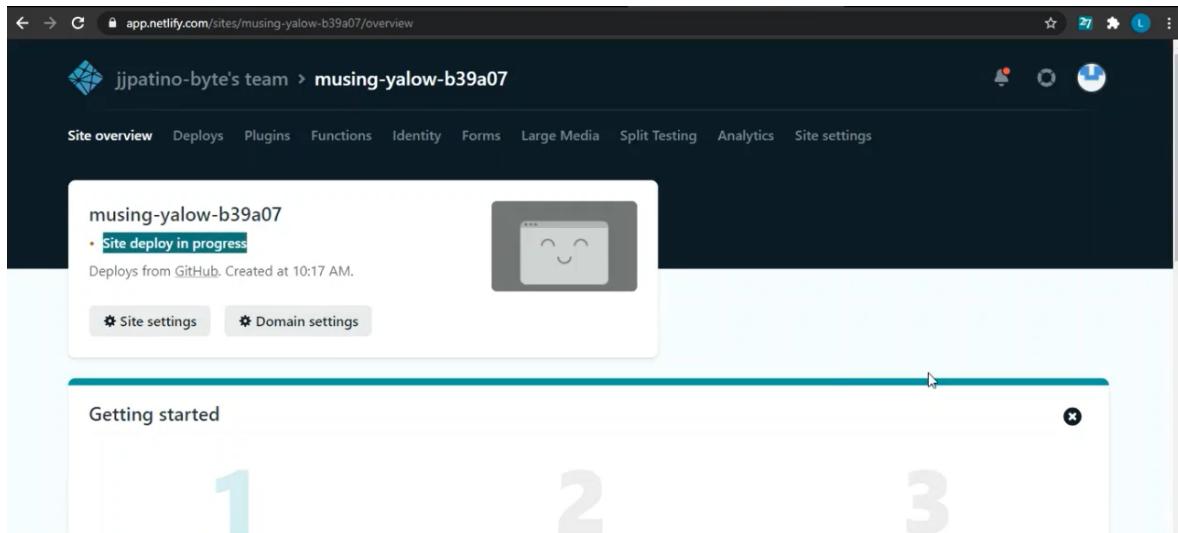




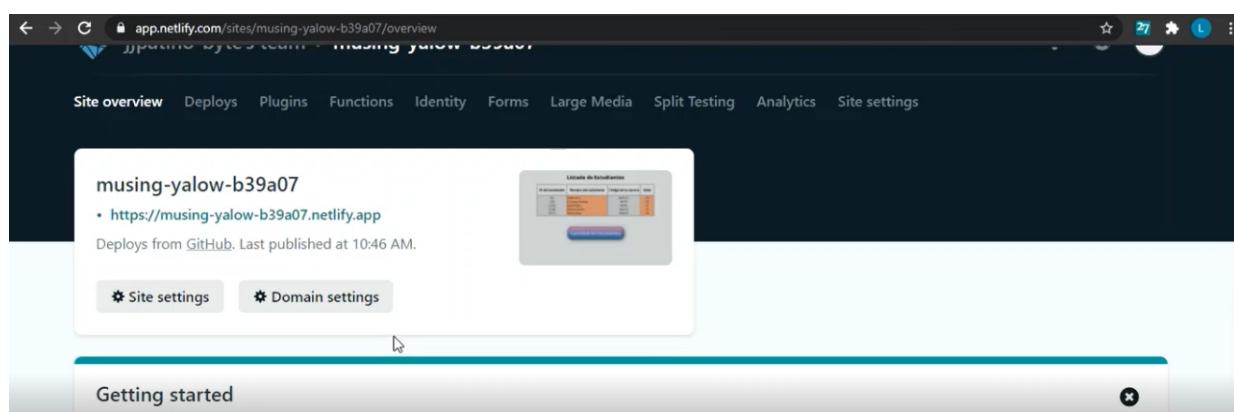
[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Cuando se hayan finalizado los tres pasos, se debe mostrar una previsualización de la página en progreso.



26. Para concluir con el despliegue, se debe ejecutar la siguiente línea de comando: `npm run build`. Luego, subir los cambios a GitHub mediante los comandos a mencionados.
27. Si todo ha sido realizado correctamente, la página de previsualización debe haber cargado el link de su página remota.





[NT4] NOTA TÉCNICA N°. 4

DESARROLLO WEB LOCAL Y EN LA NUBE CON REACT, NODE.JS, MYSQL

Al entrar al link de la página creada, se debe mostrar su frontend con los datos registrados en la base de datos.

The screenshot shows a web browser window with the URL 'musing-yallow-b39a07.netlify.app'. The page title is 'Listado de Estudiantes'. Below the title is a table with four columns: 'ID del estudiante', 'Nombre del estudiante', 'Código de la carrera', and 'Edad'. The data in the table is as follows:

ID del estudiante	Nombre del estudiante	Código de la carrera	Edad
183	Stalin Vera	INGELCT	24
1235	Carolina Ormaza	MCTG	18
12322	José Patiño	MCTG	21
12348	Andrés Patiño	INGCVL	19
18273	Ammy Bast	EMPGET	34

Below the table is a blue button labeled 'Cantidad de Estudiantes'.

Al presionar el botón, se comprueba la funcionalidad de la página:

The screenshot shows a web browser window with the URL 'musing-yallow-b39a07.netlify.app'. A modal dialog box is displayed in the center of the screen. The dialog contains the text 'musing-yallow-b39a07.netlify.app dice' and 'Existen 5 estudiantes en la lista'. Below the dialog is a blue button labeled 'Aceptar'. Below the dialog is the same table as in the previous screenshot, showing the same student data. At the bottom of the page is another blue button labeled 'Cantidad de Estudiantes'.