

```
unityofdisaster@pavilion: ~/Desktop/proyectos_web/stack_angular/backend
File Actions Edit View Help
unityofdisaster@pavilion: ~/D...tos_web/stack_angular/backend
.js
(base) unityofdisaster@pavilion:~/Desktop/proyectos_web/stack_angular/backend$ npm init -y
Wrote to /home/unityofdisaster/Desktop/proyectos_web/stack_angular/backend/package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

(base) unityofdisaster@pavilion:~/Desktop/proyectos_web/stack_angular/backend$ npm install
express cors

added 52 packages, and audited 53 packages in 7s

found 0 vulnerabilities
(base) unityofdisaster@pavilion:~/Desktop/proyectos_web/stack_angular/backend$
```

" ! " # \$

```
const express = require('express');
var cors = require('cors');

// creacion de servidor
const exprApp = express();

// uso de middlewares

// Habilitar todas las peticiones CORS
exprApp.use( cors() );

// se agregan rutas habilitadas para la API

exprApp.use('/servicio/api_notes_app/users', (req, res) => {res.send("<h1>Users</h1>")});
exprApp.use('/servicio/api_notes_app/notes', (req, res) => {res.send("<h1>Notes</h1>")});

// middleware utilizado para poder parsear formatos JSON
exprApp.use( express.json() );

// habilitar servidor para que escuche peticiones en el puerto especificado
exprApp.listen(8081, () => {
  console.log('Servidor corriendo');
})
```

\$ " % &

& " # \$!

" % ! " ' "

" "

"

" "

```
const { Router } = require('express');
const router = Router();

// se agregan las rutas y metodos correspondientes a cada operacion crud
// que se realizara con las notas

// en todas las operaciones se hace referencia a la raiz ya que la ruta
// completa se encuentra en el index y se completa al referencias este modulo

// ruta para obtener notas
router.get('/', () => {});

// ruta para agregar notas
router.post('/', () => {});

// ruta para modificar notas
router.put('/:id', () => {});

// ruta para eliminar una nota
router.delete('/:id', () => {});

module.exports = router;
```

\$ " ()

)

,

" !

& + ,

```
// ruta para obtener notas
router.get('/', getUsers);

// ruta para agregar notas
router.post('/', createUser);

// ruta para modificar notas
router.put('/:id', updateUser);

// ruta para eliminar una nota
router.delete('/:id', deleteUser);
```

\$ " - #

2

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

// se llama al metodo config para cargar el conte
nido del archivo .env en las variables
// de entorno
require('dotenv').config();

const dbConnection = async() => {

  try {

    // se toma cadena de conexion de las variables de
    entorno
    await mongoose.connect(process.env.
CONN_STR, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      useCreateIndex: true
    });
  } catch (error){
    throw new Error(
      'Error al intentar conectar a la base de datos');
  }

}

module.exports = {
  dbConnection
}
```

\$ "

1 &

"

"

```
const UserSchema = Schema({
  nombre: {
    type: String,
    required: true,
    unique: true
  }
});
```

\$ "

3)

"

```
const {Schema, model} = require('mongoose');

const NoteSchema = Schema({
  titulo: {
    type: String,
    required: true
  },
  informacion: {
    type: String,
    required: true
  },
  nombreCreador: {
    type: String,
    required: true
  },
  fecha: {
    type: Date,
    required: true
  },
});
```

\$ " 4) "

" !

" 5

```
const getNotes = async(req, res) => {
  // obtener todos los registros de la coleccion no
  tas
  const notas = await Nota.find({});
  res.json({
    notas
  });
}
```

\$ " 6)

5

```

const createNote = async(req, res) => {
  const { titulo, informacion, nombreCreador, fecha } = req.body;

  try{
    // validar que usuario ya existe, pero despues
    // asignar parametros que seran guardados en la base
    const nota = new Nota(titulo, informacion, nombreCreador, fecha);

    await nota.save();

    res.json({
      nota
    });
  } catch(error) {
    res.status(500).json({
      errorMsg: 'Error al intentar guardar registro'
    });
  }
}

```

```

const updateNote = async(req, res) => {
  const noteID = req.params.id;

  try {
    const { titulo, informacion,
nombreCreador, fecha } = req.body;

    console.log(noteID);

    // se verifica que el id ingresado sea valido
    const nota = await Nota.findById(noteID);

    if(!nota) {
      return res.status(400).json({
        errorMsg: 'no existe nota'
      });
    }

    // se agrega el parametro new: true para que la f
uncion retorne el contenido
    // del nuevo registro
    const notaActualizada = await Nota.
findByIdAndUpdate(noteID,
      { titulo, informacion, nombreCreador
, fecha },
      { new: true}, );

    console.log(notaActualizada)

    res.json({
      notaActualizada
    });

  } catch(error) {
    res.status(500).json({
      errorMsg:
'no ha sido posible actualizar la nota'
    });
  }
}

```

\$ " <)

"

"

"

```
const deleteNote = async(req, res) => {
  const noteID = req.params.id;

  try{
    // se verifica que exista el usuario que se pretende elimina
    const nota = await Nota.findById( noteID );
    if(!nota) {
      return res.status(404).json({
        errorMsg: 'No existe nota'
      });
    }

    await Nota.findOneAndDelete( noteID);

    res.json({
      msg: 'nota eliminada'
    })

  }catch(error) {
    res.status(500).json({
      errorMsg: 'No se pudo eliminar el registro'
    });
  }
}
```

\$ "

% ;)

"

5

= &

& + ,

"

&

"

\$

&

" "

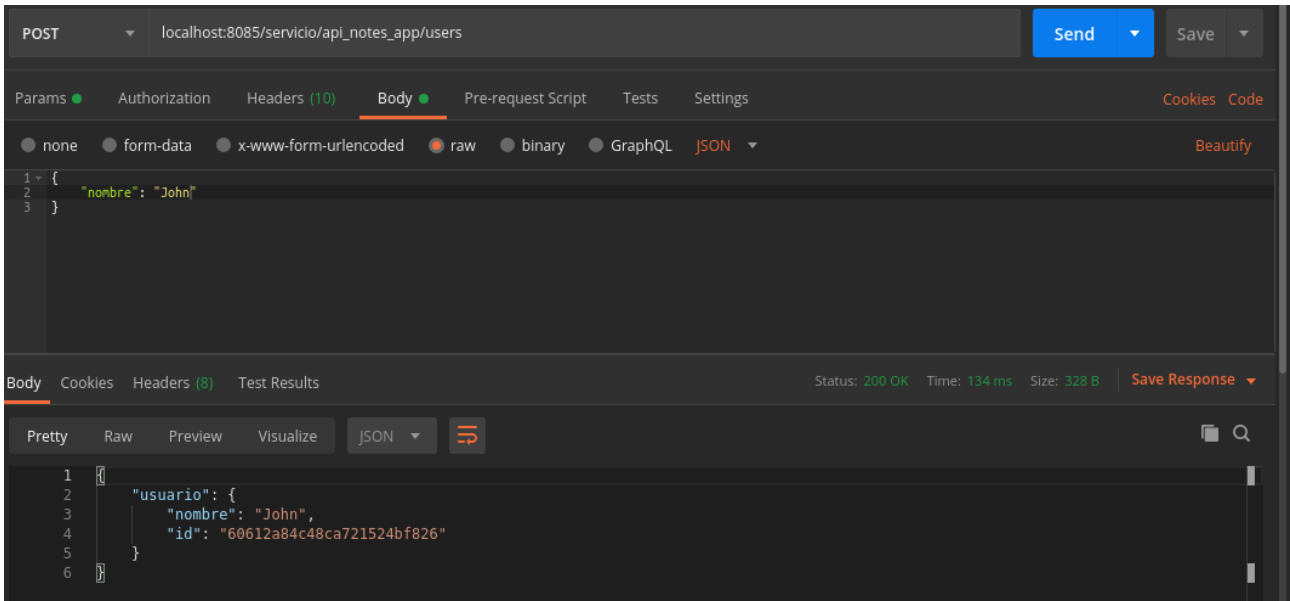
"

)

"

% %

/



\$ " % %

"

& " " ' @) ? # \$ % () "

obtener_usuarios

GET localhost:8085/servicio/api_notes_app/users

Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 147 ms Size: 433 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "usuarios": [
3     {
4       "nombre": "John",
5       "id": "60612a84c48ca721524bf826"
6     },
7     {
8       "nombre": "Jane",
9       "id": "60612abfc48ca721524bf827"
10    },
11    {
12      "nombre": "Peter",
13      "id": "60612ad8c48ca721524bf828"
14    }
15  ]
16 }
```

\$ " % (

" " "

" % - ' "

(; ;

actualizar_usuario

PUT localhost:8085/servicio/api_notes_app/users/60612a84c48ca721524bf826

Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "nombre": "Juan"
3 }
```

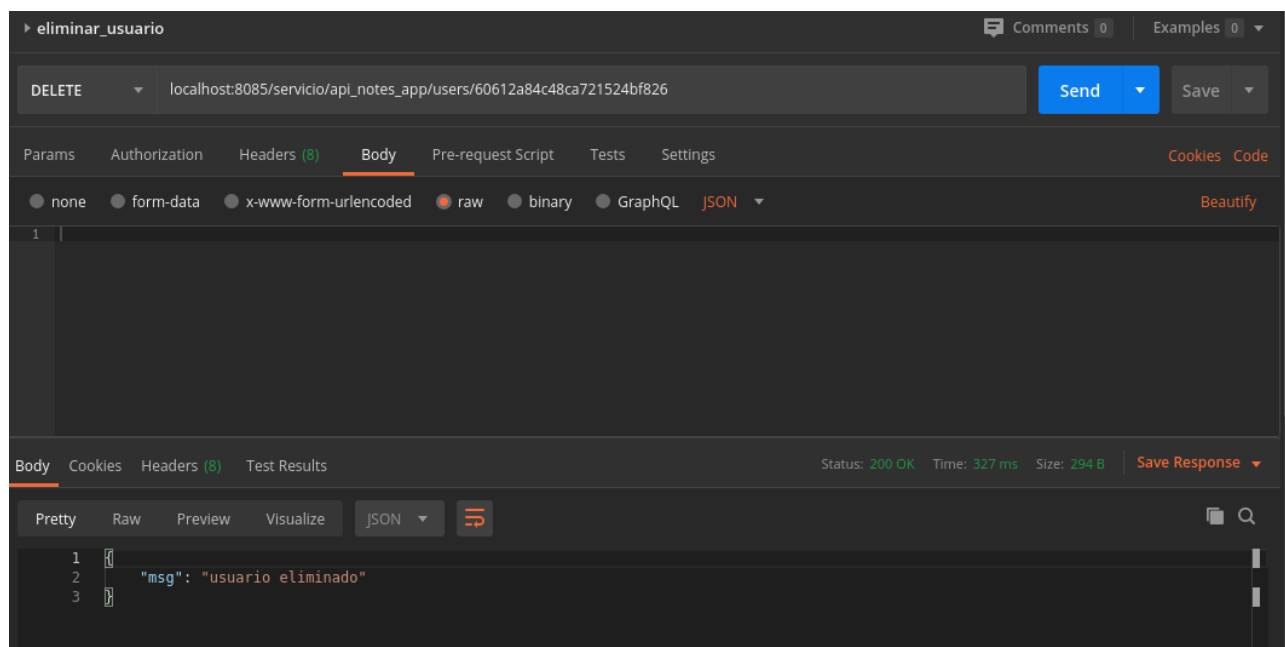
Body Cookies Headers (8) Test Results Status: 200 OK Time: 233 ms Size: 339 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "usuarioActualizado": {
3     "nombre": "Juan",
4     "id": "60612a84c48ca721524bf826"
5   }
6 }
```

\$ " % -

" ! " " \$)
* & "



\$ " % 1 "


0

"

"

"

"

Notes

Create Note

Create User

One

Title

titulo

Contenido

info

Fecha

mm/dd/yyyy

Submit

\$ "

% 4

7

"

0

8


"

!

"

"

"

Notes

Create Note

Create User

Create new User

asdasdsad

Submit

Jane

Peter

\$ "

% 6

/

"

/" 8 " #
& + ,
)
!"
" " % 6
" ' , "
) * "
"

```
getUsers() {  
  return this.http.get(`${apiURL}/users`).pipe(  
    map((response: any) => {  
      console.log(response);  
  
      // se extrae arreglo de usuarios de la respuesta  
      return response.usuarios;  
    })),  
  )  
}  
  
createUser(user: UserInterface) {  
  return this.http.post(`${apiURL}/users`, user  
    , { observe: 'response' }).pipe(  
    map((response)=> {  
      if(response.status === 200) {  
        return true;  
      } else {  
        return false;  
      }  
    }  
  ))  
}
```

\$ " % 9 &

```

deleteNoteDB(id: string) {
  return this.http.delete(`${apiURL}/notes/${id}`
    , {observe: 'response'}).pipe(
    map((response: any) => {
      if(response.status == 200) {
        return true;
      } else {
        return false;
      }
    })
  );
}

updateNote(nota: NotaInterface) {
  return this.http.put(`${apiURL}/notes/${nota.id}
    `, nota, {observe: 'response'}).pipe(
    map((response: any) => {
      if(response.status == 200) {
        return true;
      } else {
        return false;
      }
    })
  );
}

```

```

this.noteService.getNotes().subscribe((notas) => {
  this.noteArray = notas;
});

```

?

"

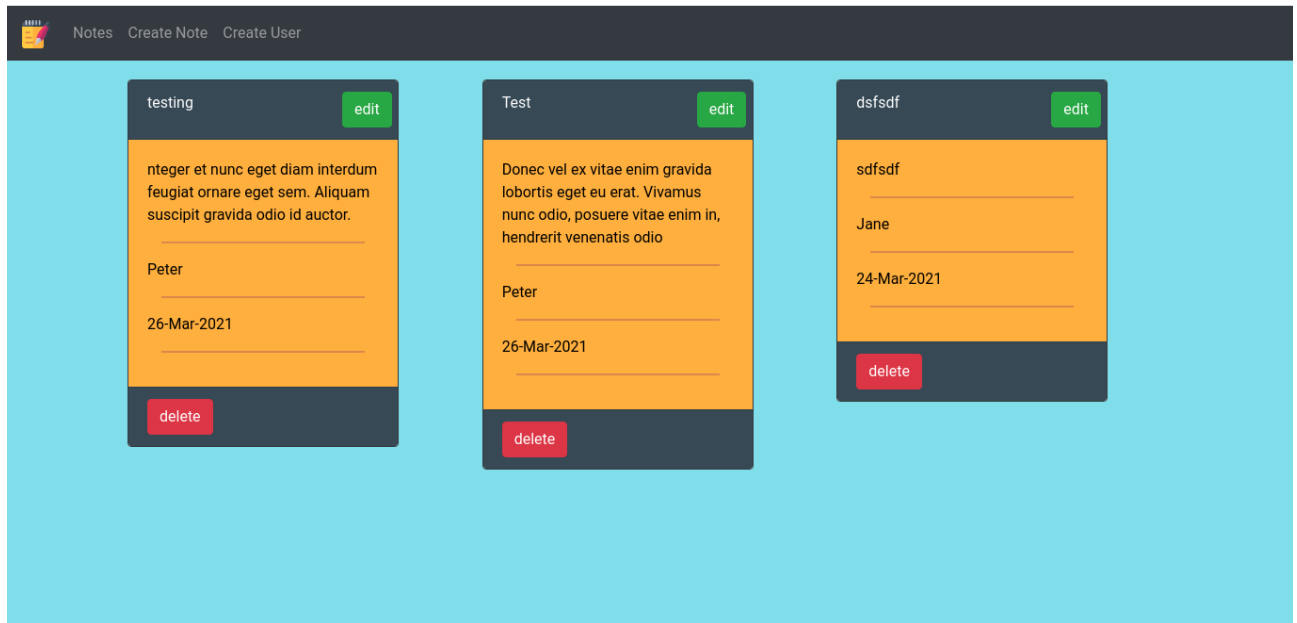
7

C \$ "

((D #

"

" "



\$ "

(% ?

"

B

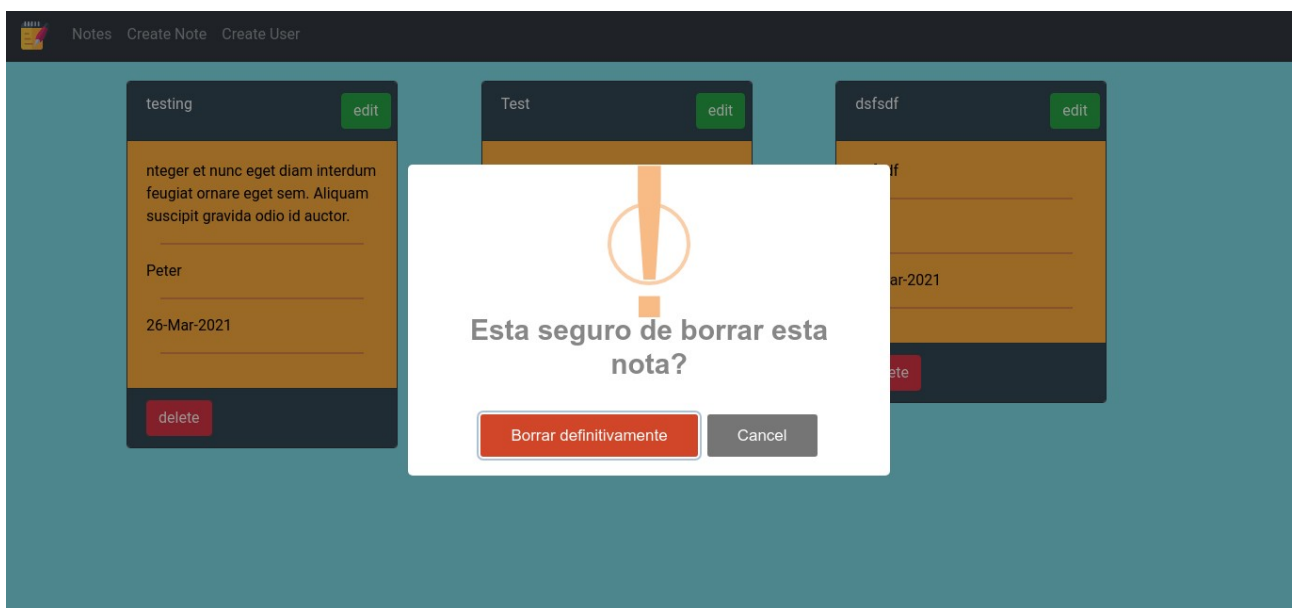
"

```

editarNota(index: number) {
  this.noteService.storeNote(this.noteArray[index]);
  this.router.navigate(['note']);
}

eliminarNota(index: number) {
  Swal.fire({
    title: 'Esta seguro de borrar esta nota?',
    showDenyButton: true,
    icon: 'warning',
    showCancelButton: true,
    denyButtonText: `Borrar definitivamente`,
  }).then((result) => {
    // se hace la operacion con el boton deny solo por estetica
    if(result.isDenied) {
      // llamado a la base de datos para eliminar registro
      const id = this.noteArray[index].id;
      this.noteService.deleteNoteDB(id).subscribe((res) => {
        if(res){
          Swal.fire('se ha eliminado la nota!', '', 'success');
        } else {
          Swal.fire('algo salio mal', '', 'error');
        }
      });
      // se elimina nota de la lista de elementos interna y tambien
      // en el html
      this.noteArray.splice(index,1);
    }
  });
}

```



Notes Create Note Create User

Title

Contenido

Fecha

mm/dd/yyyy

Guardar

Notes Create Note Create User

Peter

Title

testing

Contenido

nteger et nunc eget diam interdum feugi

Fecha

03/26/2021

Editar

```

this.noteToEdit = this.noteService.retrieveNote
();
// en caso de que se haya solicitado la edicion d
e una nota se ligam los valores
// de la nota al formulario, en caso contrario se
dejan vacios
if(this.noteToEdit) {
  this.noteForm = fb.group({
    creador: [this.noteToEdit.nombreCreador,
Validators.required],
    titulo: [this.noteToEdit.titulo, Validators.
required],
    informacion: [this.noteToEdit.informacion,
Validators.required],
    fecha: [ this.dateFormat(this.noteToEdit.
fecha), Validators.required],
  });
} else {
  this.noteForm = fb.group({
    creador: ['', Validators.required],
    titulo: ['', Validators.required],
    informacion: ['', Validators.required],
    fecha: ['', Validators.required],
  });
}

// se obtiene lista de usuarios para mostrarlos e
n el contenedor correspondiente
this.userService.getUsers().subscribe(users => {
  this.listaNombres = users;
});

```

ⓘ

Se debe seleccionar un usuario

Title

ⓘ

El titulo es obligatorio

Contenido

ⓘ

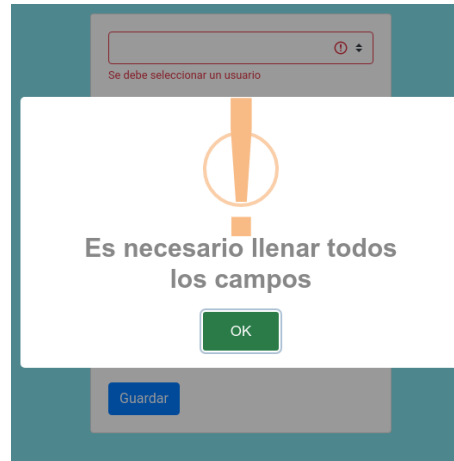
El contenido es obligatorio

Fecha

📅 ⓘ

La fecha es obligatoria

Guardar



\$ " (4) "

"

"

)

7 " .

"

"

"

"

7 "

"

```
guardarNota() {
  if(!this.noteForm.valid) {
    for (const control of Object.values(this.noteForm.controls)) {
      control.markAsTouched();
    }
    Swal.fire('Es necesario llenar todos los campos', '',
'warning');
  } else {

    // se guarda contenido de formulario en un objeto que sera enviado
    a la base de datos

    let nuevaNota: NotaInterface;
    nuevaNota = {
      titulo: this.noteForm.get('titulo').value,
      fecha: this.noteForm.get('fecha').value,
      informacion: this.noteForm.get('informacion').value,
      nombreCreador: this.noteForm.get('creador').value,
    }

    // si se solicito editar una nota se llama a la peticion para actu
    alizar
    if(this.noteToEdit) {
      nuevaNota.id = this.noteToEdit.id;
      this.peticionActualizar(nuevaNota);
    } else {
      // se guarda el contenido de la nueva nota

      this.peticionGuardar(nuevaNota);
    }
  }
}
```

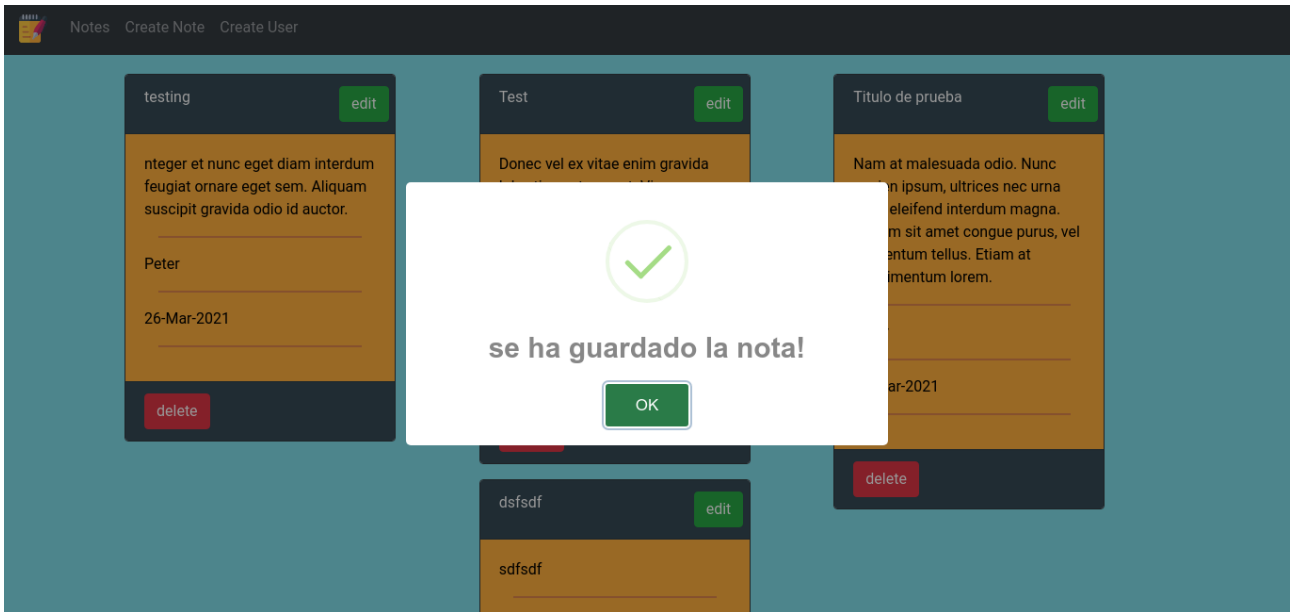
\$ " (6 0

& "

!

"

C \$ " (9 D



\$ " (9 >

"

*

)

"

"

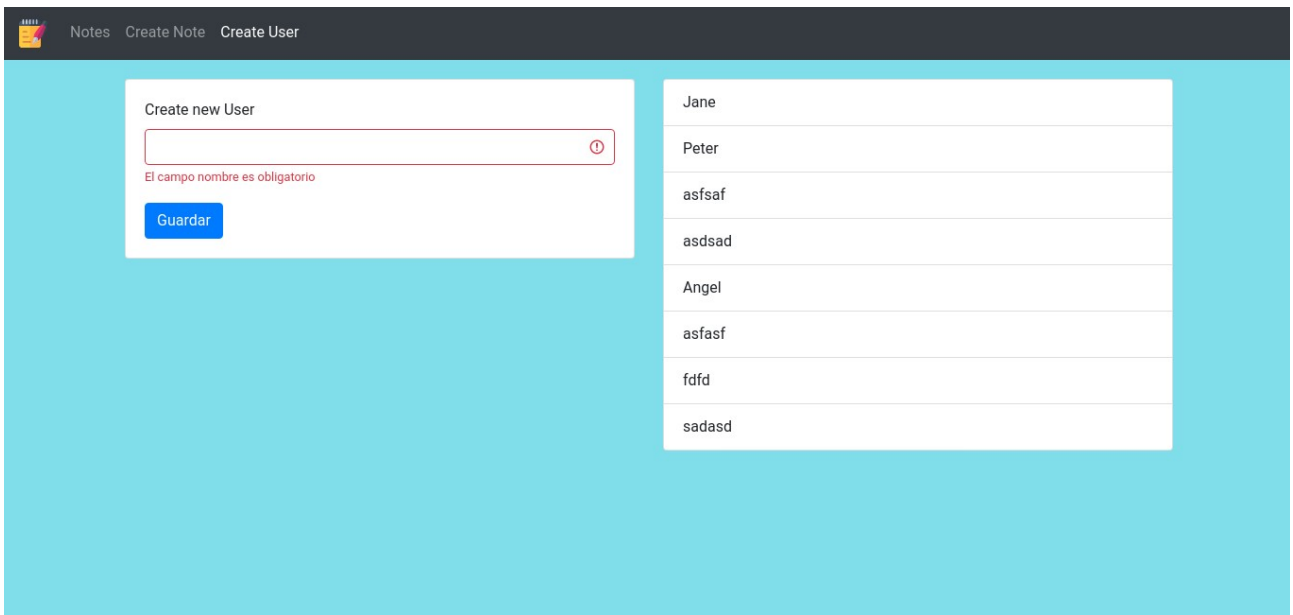
"

"

"

"

7



\$ " (< F

& "

"

"

(<

"

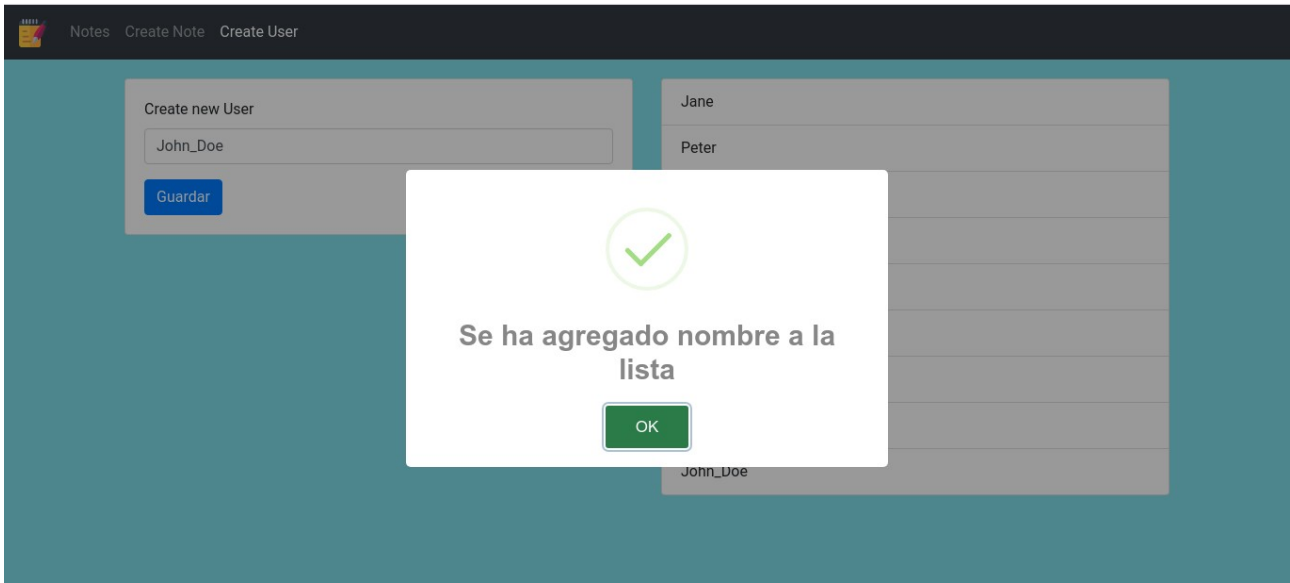
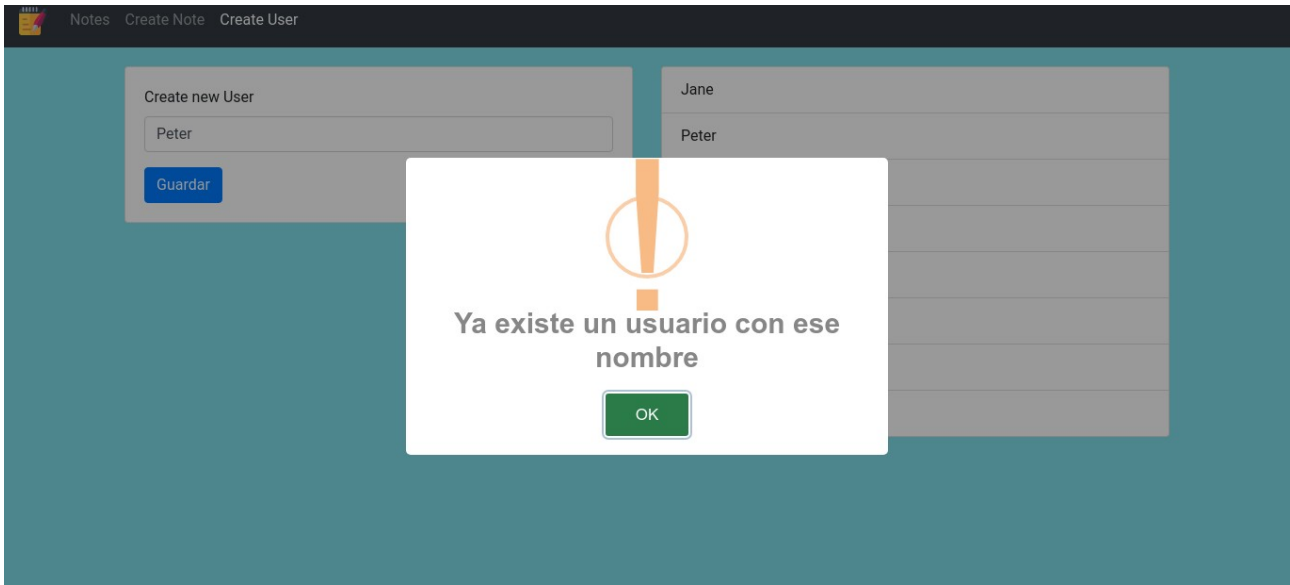
'

"

*

#

"



0
= 2 2 " " B B - < 6 < - " 2 G 2

. +

"
= 2 2 " 2 (2 B B B

= 2 2 " 2 (2 H

& # " =
" "
) / "
" " " "

& '

I I J \$ " I K L = 2 2 " L I X W 7 2 J L J " L