

Backend (Spring Boot):

Modelo de Datos: Se utiliza JPA (Java Persistence API) para gestionar la persistencia de datos. La entidad Nota representa la estructura de los datos que se almacenan en la base de datos.

API REST: Se expone una API REST que permite realizar operaciones crud. crear, leer, actualizar y eliminar sobre las notas de los estudiantes.

Seguridad: El sistema incluye medidas de seguridad para proteger el acceso a la API y a la base de datos.

Frontend (React):

Interfaz de Usuario: Se desarrolla una interfaz intuitiva para la gestión de notas, permitiendo a los usuarios ingresar, visualizar y administrar las calificaciones de los estudiantes.

Consumo de API: Utiliza fetch o axios para comunicarse con la API REST del backend y actualizar la interfaz de usuario en tiempo real.

Nota.Java

```
9 public class Nota {
15
16     private String estudiante;
17     private int actividades; // Max 35 puntos
18     private int primerParcial; // Max 15 puntos
19     private int segundoParcial; // Max 15 puntos
20     private int examenFinal; // Max 35 puntos
21     private int puntajeTotal;
22
23
24     public Nota() {}
25
26
27     public Nota(String estudiante, int actividades, int primerParcial, int segundoParcial, int examenFinal) {
28         this.estudiante = estudiante;
29         this.actividades = actividades;
30         this.primerParcial = primerParcial;
31         this.segundoParcial = segundoParcial;
32         this.examenFinal = examenFinal;
33         this.puntajeTotal = actividades + primerParcial + segundoParcial + examenFinal;
34     }
35
36
37     public Long getId() {
38         return this.id;
39     }
40
41     public void setId(Long id) {
42         this.id = id;
43     }
44
45     public String getEstudiante() {
46         return estudiante;
47     }
48 }
```

id: Identificador único para cada nota, generado automáticamente por la base de datos.

estudiante: Nombre del estudiante al que pertenecen las notas.

actividades: Puntuación obtenida en actividades, con un máximo de 35 puntos.

primerParcial: Puntuación obtenida en el primer parcial, con un máximo de 15 puntos.

segundoParcial: Puntuación obtenida en el segundo parcial, con un máximo de 15 puntos.

examenFinal: Puntuación obtenida en el examen final, con un máximo de 35 puntos.

puntajeTotal: Suma de las puntuaciones obtenidas en actividades, parciales y examen final, que representa el rendimiento total del estudiante.

NotaController.java

```
pruebas > src > main > java > com > proyectonotas > pruebas > Controller > NotaController.java > Language Support for Java(TM) by Red Hat >
20 @RestController
21 @RequestMapping("/api/notas")
22 public class NotaController {
23
24     @Autowired
25     private NotaService notaService;
26
27     // Listar notas
28     @GetMapping
29     public List<Nota> listarNotas() {
30         return notaService.listarNotas();
31     }
32
33     // Registrar una nueva nota
34     @PostMapping
35     public ResponseEntity<Nota> registrarNota(@RequestBody Nota nota) {
36         Nota nuevaNota = notaService.guardarNota(nota);
37         return ResponseEntity.ok(nuevaNota);
38     }
39
40     // Obtener una nota por ID
41     @GetMapping("/{id}")
42     public ResponseEntity<Nota> obtenerNota(@PathVariable Long id) {
43         Nota nota = notaService.obtenerNotaPorId(id);
44         return ResponseEntity.ok(nota);
45     }
46
47     // Actualizar una nota existente
48     @PutMapping("/{id}")
49     public ResponseEntity<Nota> actualizarNota(@PathVariable Long id, @RequestBody Nota notaActualizada) {
50         notaActualizada.setId(id); // Establece el ID de la nota que se actualizará
51         Nota nota = notaService.guardarNota(notaActualizada);
52         return ResponseEntity.ok(nota);
53     }
54
55     // Eliminar una nota
56     @DeleteMapping("/{id}")
57     public ResponseEntity<Void> eliminarNota(@PathVariable Long id) {
```

La clase **NotaController** es un controlador REST en una aplicación Spring que gestiona las operaciones relacionadas con las notas de los estudiantes. Utiliza el servicio **NotaService** para interactuar con los datos de las notas.

listarNotas: Recupera y devuelve una lista de todas las notas registradas en la base de datos.

registrarNota: Permite crear una nueva nota a partir de los datos recibidos en el cuerpo de la solicitud (JSON). Retorna la nota recién creada.

obtenerNota: Busca y devuelve una nota específica mediante su ID, extraído de la URL de la solicitud.

actualizarNota: Actualiza una nota existente. Recibe los datos actualizados en el cuerpo de la solicitud, asigna el ID de la nota que se desea actualizar y devuelve la nota modificada.

eliminarNota: Elimina una nota de la base de datos utilizando su ID. No devuelve contenido en la respuesta, solo indica que la operación se ha completado.

App.txt

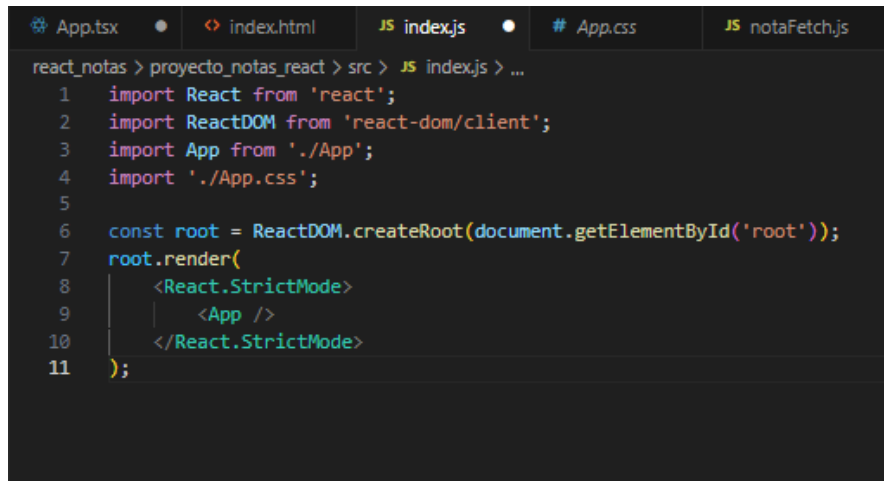
```
3 interface Nota {
4   primerParcial: number;
5   segundoParcial: number;
6   examenFinal: number;
7   total: number;
8 }
9
10
11
12 function App() {
13   const [notas, setNotas] = useState<Nota[]>([]);
14   const [estudiante, setEstudiante] = useState('');
15   const [actividades, setActividades] = useState(0);
16   const [primerParcial, setPrimerParcial] = useState(0);
17   const [segundoParcial, setSegundoParcial] = useState(0);
18   const [examenFinal, setExamenFinal] = useState(0);
19   const [total, setTotal] = useState(0);
20
21
22   useEffect(() => {
23     const fetchNotas = async () => {
24       try {
25         const response = await fetch('http://localhost:8080/api/notas');
26         const data = await response.json();
27         setNotas(data);
28       } catch (error) {
29         console.error('Error al obtener las notas:', error);
30       }
31     };
32
33     fetchNotas();
34   }, []);
35
36   useEffect(() => {
37     const calcularTotal = () => {
38       setTotal(actividades + primerParcial + segundoParcial + examenFinal);
39     };
40     calcularTotal();
41   }, [actividades, primerParcial, segundoParcial, examenFinal]);
42
43   const agregarNota = () => {
```

Al cargar el componente, hago una solicitud a mi API en `http://localhost:8080/api/notas` para obtener las notas almacenadas en la base de datos y las asigno a mi estado `notas`.

También implemento un cálculo automático del puntaje total cada vez que cambio alguno de los puntajes individuales. Para agregar una nueva nota, tengo una función que verifica que el nombre del estudiante no esté vacío y, si todo está correcto, añade la nota a la lista. Después de agregar la nota, limpio los campos de entrada para que el usuario pueda ingresar los datos de un nuevo estudiante fácilmente.

Finalmente, el componente renderiza un formulario que permite ingresar los datos del estudiante y sus notas, junto con un botón para agregar la nota. También muestro una tabla con todas las notas registradas, donde se pueden ver el nombre del estudiante y sus puntajes en cada evaluación, así como el puntaje total.

Index.js



```
react_notas > proyecto_notas_react > src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import App from './App';
4  import './App.css';
5
6  const root = ReactDOM.createRoot(document.getElementById('root'));
7  root.render(
8    <React.StrictMode>
9      <App />
10    </React.StrictMode>
11  );
```

configuro el punto de entrada de mi aplicación React. Primero, importo las dependencias necesarias, incluyendo React y ReactDOM, así como el componente principal App que contiene la lógica de mi aplicación. También importo el archivo de estilos App.css para aplicar estilos a los componentes.

Luego, creo el nodo raíz de mi aplicación utilizando ReactDOM.createRoot, apuntando al elemento del DOM con el ID root, que es donde se montará mi aplicación.

Finalmente, renderizo el componente App dentro del modo estricto de React. El React.StrictMode es una herramienta de desarrollo que ayuda a identificar problemas potenciales en la aplicación. Este archivo es crucial porque establece la base sobre la cual se construye y se ejecuta toda mi aplicación React.

Creación de base de datos

Query 1 x Administrator

```
1 • CREATE DATABASE sistema_notas;
2 • CREATE USER 'usuario_notas'@'localhost' IDENTIFIED BY 'contrasena123';
3 • GRANT ALL PRIVILEGES ON sistema_notas.* TO 'usuario_notas'@'localhost';
4 • FLUSH PRIVILEGES;
5 • SELECT user FROM mysql.user;
6 • CREATE TABLE `NOTA` (
7   `ID` BIGINT,
8   `ESTUDIANTE` VARCHAR(60),
9   `CURSO` VARCHAR(40),
10  `ACTIVIDADES` INT,
11  `PARCIAL1` INT,
12  `PARCIAL2` INT,
13  `EXAMEN_FINAL` INT,
14  `TOTAL` INT,
15  PRIMARY KEY (`ID`)
16 );
17 • SELECT * FROM NOTA;
```

Result Grid

| ID | ESTUDIANTE | CURSO | ACTIVIDADES | PARCIAL1 | PARCIAL2 | EXAMEN_FINAL | TOTAL |
|----|------------|-------|-------------|----------|----------|--------------|-------|
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

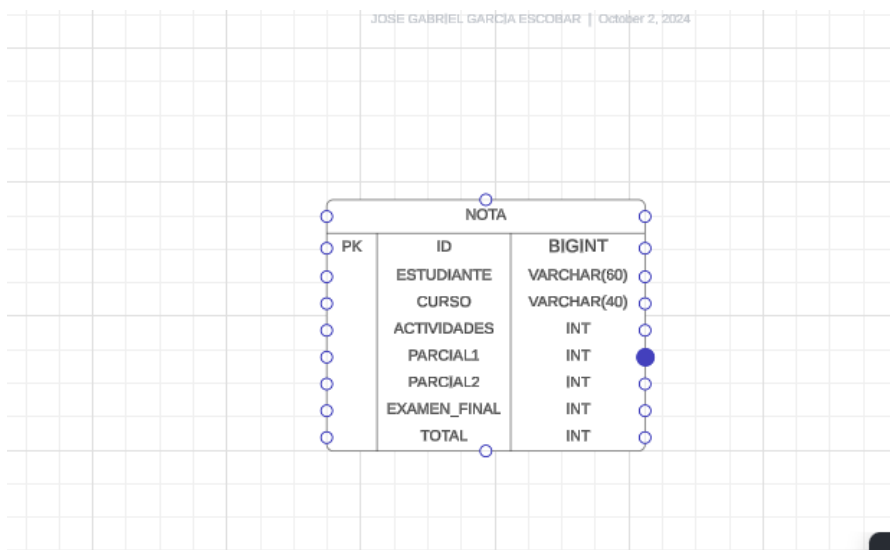
BASEDEDATOSVISUALPROY...

```
1 • CREATE DATABASE sistema_notas;
2 • use sistema_notas;
3
4 • CREATE USER 'usuario_notas'@'localhost' IDENTIFIED BY 'contrasena123';
5 • GRANT ALL PRIVILEGES ON sistema_notas.* TO 'usuario_notas'@'localhost';
6 • FLUSH PRIVILEGES;
7 • SELECT user FROM mysql.user;
8 • CREATE TABLE `NOTA` (
9   `ID` BIGINT auto_increment,
10  `ESTUDIANTE` VARCHAR(60),
```

Result Grid

| ID | estudiante | ACTIVIDADES | EXAMEN_FINAL | primer_parcial | segundo_parcial | TOTAL | puntaje_total |
|----|-------------|-------------|--------------|----------------|-----------------|-------|---------------|
| 3 | Juan Perez | 30 | 25 | 10 | 10 | 0 | 0 |
| 4 | Juan Perez | 15 | 15 | 15 | 15 | 0 | 0 |
| 5 | Juan Perez | 15 | 15 | 15 | 15 | 0 | 0 |
| 6 | Pedro Perez | 30 | 15 | 15 | 15 | 0 | 0 |
| 7 | Pedro Perez | 30 | 15 | 15 | 15 | 0 | 0 |
| 8 | Pedro Perez | 30 | 15 | 15 | 15 | 0 | 0 |

Lucidchart para realizar la estructura de mysql



REACT: http://localhost:5173

```
C:\WINDOWS\system32\cmd. X + v

VITE v5.4.8 ready in 393 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

```
PS C:\Users\Willys GaBo> cd .\Desktop\
PS C:\Users\Willys GaBo\Desktop> cd .\prueba2\
PS C:\Users\Willys GaBo\Desktop\prueba2> cd .\react_notas\
PS C:\Users\Willys GaBo\Desktop\prueba2\react_notas> cd .\proyecto_notas_react\
PS C:\Users\Willys GaBo\Desktop\prueba2\react_notas\proyecto_notas_react> npm run dev

> proyecto_notas_react@0.0.0 dev
> vite

VITE v5.4.8 ready in 1768 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

← → ↻ ⓘ localhost:5173

Will Ocio Programas Cristianos Trabajo (20+) Facebook (20+) Video | Faceb... Como ejecutar el Se... WhatsApp YouTube

Aplicación de Notas

Nombre del Estudiante

Total: 0

Agregar Nota

Lista de Notas

| Estudiante | Actividades | Primer Parcial | Segundo Parcial | Examen Final | Total |
|------------|-------------|----------------|-----------------|--------------|-------|
| Pablo | 10 | 10 | 20 | 35 | 75 |
| Calculo | 10 | 10 | 20 | 20 | 60 |

<http://localhost:8080/api/notas>

esta todo en lo que me sale en la base de datos, las pruebas que he realizado porque no me salía "PuntajeTotal"

```
localhost:8080/api/notas

Impresión con formato estilístico ☐

[{"id":1,"estudiante":"JOSE GABRIEL","actividades":30,"primerParcial":0,"segundoParcial":0,"examenFinal":25,"puntajeTotal":0}, {"id":2,"estudiante":"Juan Perez","actividades":30,"primerParcial":10,"segundoParcial":10,"examenFinal":25,"puntajeTotal":0}, {"id":3,"estudiante":"Juan Perez","actividades":30,"primerParcial":10,"segundoParcial":10,"examenFinal":25,"puntajeTotal":0}, {"id":4,"estudiante":"Juan Perez","actividades":15,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":5,"estudiante":"Juan Perez","actividades":15,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":6,"estudiante":"Pedro Perez","actividades":30,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":7,"estudiante":"Pedro Perez","actividades":30,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":8,"estudiante":"Pedro Perez","actividades":30,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":9,"estudiante":"Pedro Perez","actividades":30,"primerParcial":15,"segundoParcial":15,"examenFinal":15,"puntajeTotal":0}, {"id":10,"estudiante":null,"actividades":0,"primerParcial":0,"segundoParcial":0,"examenFinal":0,"puntajeTotal":0}, {"id":11,"estudiante":"Pedro Garcia","actividades":30,"primerParcial":15,"segundoParcial":15,"examenFinal":20,"puntajeTotal":0}]
```

```
localhost:8080/api/notas

Impresión con formato estilístico ☒

[
  {
    "id": 1,
    "estudiante": "JOSE GABRIEL",
    "actividades": 30,
    "primerParcial": 0,
    "segundoParcial": 0,
    "examenFinal": 25,
    "puntajeTotal": 0
  },
  {
    "id": 2,
    "estudiante": "Juan Perez",
    "actividades": 30,
    "primerParcial": 10,
    "segundoParcial": 10,
    "examenFinal": 25,
    "puntajeTotal": 0
  },
  {
    "id": 3,
    "estudiante": "Juan Perez",
    "actividades": 30,
    "primerParcial": 10,
    "segundoParcial": 10,
    "examenFinal": 25,
    "puntajeTotal": 0
  },
  {
    "id": 4,
    "estudiante": "Juan Perez",
    "actividades": 15,
    "primerParcial": 15,
    "segundoParcial": 15,
    "examenFinal": 15,
    "puntajeTotal": 0
  }
]
```

De postman a base de datos

