

# Integradora

## BASES DE DATOS PARA CÓMPUTO EN LA NUBE

Maximiliano Carsi Castrejón

---

### INTEGRANTES:

Estrada Hernández Andrea Michelle

Garcia Araujo Ximena Natalia

Hernandez De la Cruz Axel

5A DSM

# ÍNDICE

○	Introducción .....	3	○
	Justificación .....	4	
○	Objetivo General .....	4	○
	Desarrollo .....	5	
○	Resultados.....	13	○
	Conclusión .....	17	
○			○

# INTRODUCCIÓN

En este proyecto se podrá observar cómo los estudiantes pusieron en marcha los conocimientos adquiridos a través del cuatrimestre, para poder realizar un proyecto integrador con las especificaciones que nos dio nuestro profesor.

Este mismo tratará de una banda de k-pop (blackpink), en el cual se podrá ver las siguientes opciones: Integrantes, Álbumes, Canciones y Marcas.

Cada opción tiene sus diferentes datos que se mostraran. Siendo básicos

Para los integrantes se mostrarán:

- Id
- Nombre
- Nacionalidad
- Cumpleaños
- Posición

Para los Álbumes se mostrarán:

- Id
- Nombre del álbum
- Fecha de lanzamiento
- Numero de canciones
- Ventas

Para las Canciones se mostrarán:

- Id
- Nombre de la canción
- Premios
- Album

Para las Marcas se mostrarán:

- Id
- Nombre de la marca
- Miembro
- Numero de comerciales
- Embajadora

Cómo tal no es una necesidad a resolver pero, solo es bajo entretenimiento. Se resolvió mediante la implementación de crud's básicos junto con las tecnologías vistas en clase, tal ejemplo es ReactJs, SpringBoot, MongoDB y aws.

# JUSTIFICACIÓN

Este proyecto se realizó ya que el grupo de k-pop tiene una problemática que es la siguiente:

La banda de k-pop mejor conocida como blackpink quiere tener una mejor rapidez en consultas de estas 4 opciones: Integrantes, Álbumes, Canciones y Marcas, ya que en su empresa se les dificulta esta parte porque las consultas son manualmente y a la hora de que les piden una consulta se tardan bastante

## OBJETIVO GENERAL

Nuestro objetivo es poder ayudar a la empresa de blackpink en cuanto a la rapidez de las consultas, ya que todo empleado que quiera realizar una consulta es entendible que no tiene mucho tiempo como para que se realicen las consultas manualmente

# PROPUESTA DE SOLUCIÓN

Para una mejor rapidez en las consultas se propuso la siguiente solución:

Realizar un proyecto donde las opciones que se mencionaron en la introducción se muestran sus diferentes datos en forma de tabla y así pueda ser más eficiente y rápido la consulta de los diferentes datos.

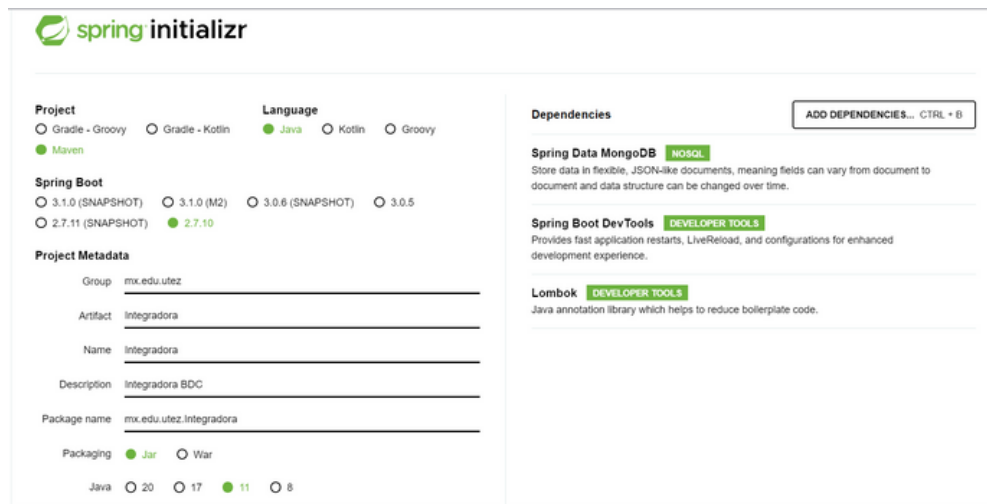
## TECNOLOGIAS Y HERRAMIENTAS UTILIZADAS



# DESARROLLO

## Spring Boot

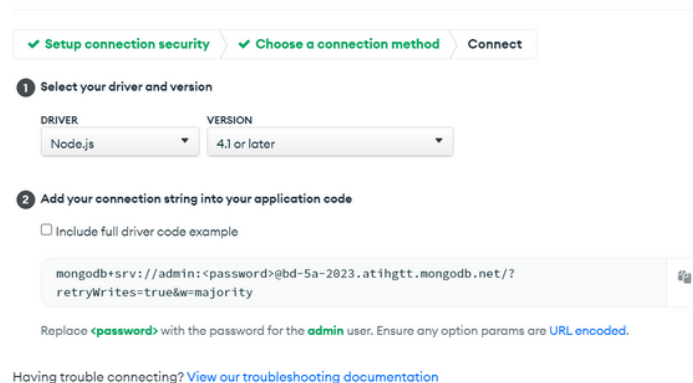
Se mostrará el proceso del backend usando spring boot. Es importante crear el proyecto con las dependencias necesarias para el uso de mongodb



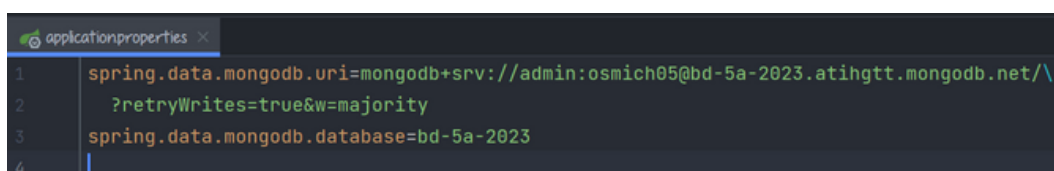
The image shows the Spring Initializr web form. Under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', '2.7.10' is selected. The 'Project Metadata' section has the following values: Group (mx.edu.utez), Artifact (Integradora), Name (Integradora), Description (Integradora BDC), and Package name (mx.edu.utez.Integradora). Under 'Packaging', 'Jar' is selected. Under 'Java', '11' is selected. The 'Dependencies' section on the right lists 'Spring Data MongoDB' (with a 'HSQL' tag), 'Spring Boot DevTools' (with a 'DEVELOPER TOOLS' tag), and 'Lombok' (with a 'DEVELOPER TOOLS' tag). Each dependency has a brief description.

Para la base de datos se usó la misma que se creó durante la clase en atlas. En este caso para poder usarla , seleccionamos la opción de 'Conect' y 'Connect your application'. El código ejemplo lo incryptamos en 'applicationproperties' junto con la contraseña y el nombre de ésta base

### Connect to BD-5A-2023



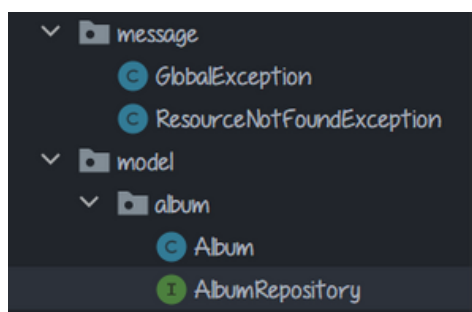
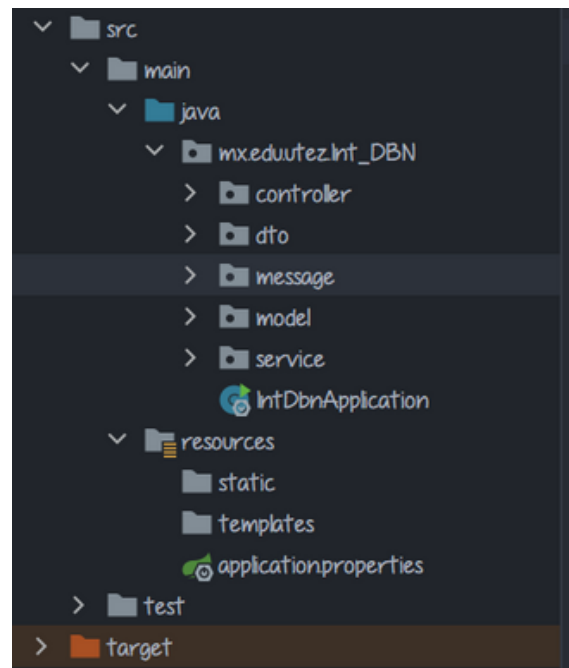
The image shows the MongoDB Atlas 'Connect to BD-5A-2023' form. It has three tabs: 'Setup connection security', 'Choose a connection method', and 'Connect'. The 'Choose a connection method' tab is active. Under '1 Select your driver and version', 'Node.js' is selected for the driver and '4.1 or later' for the version. Under '2 Add your connection string into your application code', there is a checkbox for 'Include full driver code example' which is unchecked. A text box contains the connection string: 'mongodb+srv://admin:<password>@bd-5a-2023.atihgtt.mongodb.net/?retryWrites=true&w=majority'. Below the text box, a note says: 'Replace <password> with the password for the admin user. Ensure any option params are URL encoded.' At the bottom, there is a link: 'Having trouble connecting? View our troubleshooting documentation'.



The image shows a screenshot of a code editor with a file named 'application.properties'. The file contains the following lines of code:  
1 spring.data.mongodb.uri=mongodb+srv://admin:osmich05@bd-5a-2023.atihgtt.mongodb.net/  
2 ?retryWrites=true&w=majority  
3 spring.data.mongodb.database=bd-5a-2023  
4

Creamos las carpetas necesarias para cada uno de los crud.

- Model: Paquete donde se crea la clase que representan la tabla de la base de datos.
- Service: Paquete en donde se crea la clase que tiene como fin hacer la implementación de los métodos que se definan para la aplicación.
- Dto: Paquete donde se crea la clase que se limita a ser un objeto de transferencia entre el cliente y el servidor, recordemos el principio de responsabilidad única, donde la idea es que la entidad como únicamente el modelo de la tabla de la base de datos.
- Controller: Paquete donde se crea la clase que actúa como controlador Rest, es decir exponer las Apis que se definan.



Las interfaces o mejor dicho los repositorios se utilizan para proporcionar una abstracción de acceso a datos para la capa de persistencia de una aplicación.

Los mensaje devuelven información de un usuario y la búsqueda del usuario falla porque el usuario no existe.

A continuación una muestra de las clases que se usaron para un sólo crud. Es importante poner el nombre de la colección así como los elementos. Nota para que el id sea automático se necesita hacer un método. Es importante traer los setters y getters, así como los constructores

```
@Document(collection = "members")
public class Members {
    3 usages
    @Id
    private int id;
    3 usages
    private String nombre;
    3 usages
    private String nacionalidad;
    3 usages
    private String cumpleaños;
    3 usages
    private String posicion;

    public Members() {}
    1 usage
```

```

public class MembersDto {
    3 usages
    private String nombre;
    3 usages
    private String nacionalidad;
    3 usages
    private String cumpleaños;
    3 usages
    private String posicion;

    public MembersDto() {
    }

    public MembersDto( String nombre, String nacionalidad, String cumpleaños, String posicion) {...}
}

```

En el caso del dto se usan los mismos componentes que en el model pero con la diferencia que el id no se llamará. La anotación @Repository se utiliza en la definición de la interfaz MembersRepository para marcarla como un componente de repositorio de datos y proporcionar una abstracción de acceso a datos para la entidad Members utilizando MongoDB como mecanismo de persistencia de datos.

```

@Repository
public interface MembersRepository extends MongoRepository<Members, Integer> {
    //public List<Members> findById(String id);
    /*public List<Members> findByNombre(String nombre);
    public List<Members> findByNacionalidad(String nacionalidad);
    public List<Members> findByPosicion(String posicion);*/
}

```

Finalmente la clase controller maneja las solicitudes HTTP entrantes y proporcionar una respuesta apropiada a través de la lógica de procesamiento definida en sus métodos. Es de suma importancia usar el @CrossOrigin que habilita el acceso a recursos desde orígenes distintos al del servidor que proporciona los recursos.

```

@RestController
@RequestMapping("/members")
@CrossOrigin
public class MembersControllers {
    5 usages
    @Autowired
    MembersService membersService;

    @GetMapping("/")
    public ResponseEntity<List<Members>> getAllMembers() { return ResponseEntity.ok(membersService.getAllMembers()); }

    @GetMapping("/{id}")
    public ResponseEntity<Members> getMembersById(@PathVariable("id") int id) throws ResourceNotFoundException {
        return ResponseEntity.ok(membersService.getMembersById(id));
    }

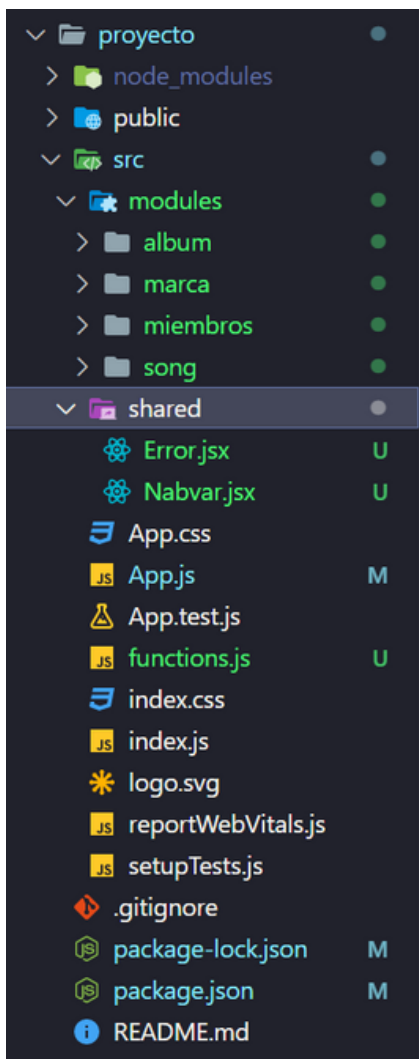
    @PostMapping("/add")
    public ResponseEntity<Members> save(@RequestBody MembersDto membersDto){
        return ResponseEntity.ok(membersService.save(membersDto));
    }

    @PutMapping("/{id}/update/{id}")
    public ResponseEntity<Members> update(@PathVariable("id") int id, @RequestBody MembersDto membersDto) throws ResourceNotFoundException {
        return ResponseEntity.ok(membersService.update(id, membersDto));
    }

    @DeleteMapping("/{id}/delete/{id}")
    public ResponseEntity<Members> delete(@PathVariable("id") int id) throws ResourceNotFoundException {
        return ResponseEntity.ok(membersService.delete(id));
    }
}

```





## React Js

En el caso del front se usó ReactJS, la mayoría de las tareas se realizan utilizando la herramienta de línea de comandos "npm" (Node Package Manager) o "yarn". Importamos dependencias así como diferentes componentes, un gran ejemplo es la implementación de bootstrap y router-dom.

La carpeta modules contiene las diferentes pantallas según los CRUDs establecidos, dentro de shared está el navbar que tendrán en común.

Dentro del app se hizo el control de las rutas así como protegerlas y en caso de no encontrarse arrojar un error

```
function App() {
  return (
    <BrowserRouter>
      <Navbar/>
      <Routes>
        <Route path="/" element={<Members/>} />
        <Route path="/members" element={<Members/>} />
        <Route path="/songs" element={<Songs/>} />
        <Route path="/album" element={<Album/>} />
        <Route path="/marcas" element={<Marca/>} />
        <Route path="/*" element={<Error/>} />
      </Routes>
    </BrowserRouter>
  );
}
```

El siguiente código sirve para realizar una solicitud GET a un servidor local utilizando la biblioteca axios de JavaScript y obtener una lista de miembros. El resultado de la solicitud se almacena en la variable respuesta y se establece en el estado member mediante la función setMember. Además, utiliza React Hooks para definir y actualizar el estado de las variables id, nombre, nacionalidad, cumpleaños, posición y title.

La función useEffect se usa para que la solicitud GET se realice cuando el componente se monte por primera vez. Esto se logra pasando un array vacío como segundo argumento de la función useEffect.

En resumen, este código obtiene los datos de una lista de miembros de un servidor local y los almacena en el estado del componente React.

```
const apiUrl = 'http://localhost:8080/members/';

const [members, setMembers] = useState([]);
const [id, setId] = useState('');
const [nombre, setNombre] = useState('');
const [nacionalidad, setNacionalidad] = useState('');
const [cumpleaños, setCumpleaños] = useState('');
const [posición, setPosición] = useState('');

const [title, setTitle] = useState('');

useEffect(() => {
  getMembers();
}, []);

const getMembers = async () => {
  const respuesta = await axios.get(apiUrl);
  setMembers(respuesta.data)
}
```

```

<tbody>
  {members.map((member, i) => (
    <tr key={member.id} className="table-active">
      <td>{(i + 1)}</td>
      <td>{member.nombre}</td>
      <td>{member.nacionalidad}</td>
      <td>{member.cumpleaños}</td>
      <td>{member.posicion}</td>
      <td>...</td>
      <td>...</td>
    </tr>
  )
  )}
</tbody>

```

Usamos la función `map()` para iterar sobre un array de `members` y renderizar cada miembro en una tabla HTML.

Cada fila de la tabla se asigna una clave única utilizando el valor del `id` del miembro (`key={member.id}`) para mejorar el rendimiento y evitar problemas de renderizado innecesario.

Pruebas:

#	Nombre	Nacionalidad	Cumpleaños	Posición		
1	Kim Jennie	Coreana	1996-01-16	Rapera Principal	Editar	Eliminar
2	Kim Jisoo	Coreana	1995-01-03	Visual Sub-Vocalista	Editar	Eliminar
3	Roseanne Park	Neozelandesa	1997-01-12	Vocalista Principal	Editar	Eliminar
4	Lalisa Manoban	Tailandesa	1997-03-27	Bailarina Principal	Editar	Eliminar
5	Lalisa Manoban	Tailandesa	1997-03-27	Bailarina Principal	Editar	Eliminar

# AWS

Se creo una instancia con los permisos necesarios para la ejecución del proyecto

### Resumen de instancia de i-01ef52f793134fbec (IntB1Mich) [Información](#)

Se ha actualizado hace less than a minute

[Conectar](#) [Estado de la instancia](#) [Acciones](#)

ID de la instancia i-01ef52f793134fbec (IntB1Mich)	Dirección IPv4 pública 54.164.31.87   <a href="#">dirección abierta</a>	Direcciones IPv4 privadas 172.31.85.128
Dirección IPv6 -	Estado de la instancia <span>En ejecución</span>	DNS de IPv4 pública ec2-54-164-31-87.compute-1.amazonaws.com   <a href="#">dirección abierta</a>
Tipo de nombre de anfitrión Nombre de IP: ip-172-31-85-128.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-172-31-85-128.ec2.internal	
Responder al nombre DNS de recurso privado IPv4 (A)	Tipo de instancia t2.micro	Direcciones IP elásticas -
Dirección IP asignada automáticamente 54.164.31.87 [IP pública]	ID de VPC vpc-0a003f76bdc7fa876	Hallazgo de AWS Compute Optimizer <a href="#">Suscribirse a AWS Compute Optimizer para recibir recomendaciones.</a> <a href="#">  Más información</a>
Rol de IAM -	ID de subred subnet-03a6d3fe419a20241	Nombre del grupo de Auto Scaling -
IMDSv2 Optional		

Se debe de entrar al cmd en donde usaremos lo instancia junto con la llave encriptada, así mismo la ipv pública

```
ubuntu@ip-172-31-85-128:~/jars$ sudo lsof -i 8080
lsof: unknown protocol name (8080) in: -i 8080
lsof 4.93.2
latest revision: https://github.com/lsof-org/lsof
latest FAQ: https://github.com/lsof-org/lsof/blob/master/00FAQ
latest (non-formatted) man page: https://github.com/lsof-org/lsof/blob/master/Lsof.8
usage: [-?abhKlnNoOPRtUvVX] [+|-c c] [+|-d s] [+D D] [+|-E] [+|-e s] [+|-f[gG]]
[-F [f]] [-g [s]] [-i [i]] [+|-L [l]] [+m [m]] [+|-M] [-o [o]] [-p s]
[+|-r [t]] [-s [p:s]] [-S [t]] [-T [t]] [-u s] [+|-w] [-x [f]] [--] [names]
Use the ``-h'' option to get more help information.
ubuntu@ip-172-31-85-128:~/jars$ sudo lsof -i:8080
sudo: lsof: command not found
ubuntu@ip-172-31-85-128:~/jars$ sudo Isuf -i:8080
sudo: Isuf: command not found
ubuntu@ip-172-31-85-128:~/jars$ sudo Isuf -i :8080
sudo: Isuf: command not found
ubuntu@ip-172-31-85-128:~/jars$ sudo isof -i :8080
sudo: isof: command not found
ubuntu@ip-172-31-85-128:~/jars$ sudo Lsof -i :8080
sudo: Lsof: command not found
ubuntu@ip-172-31-85-128:~/jars$ sudo lsof -i :8080
COMMAND  PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java     13564 ubuntu  16u  IPv6  34246      0t0  TCP *:http-alt (LISTEN)
java     13564 ubuntu  21u  IPv6  34262      0t0  TCP ip-172-31-85-128.ec2.internal:http-alt->fixed-187-188-133-8.to
talplay.net:62720 (ESTABLISHED)
ubuntu@ip-172-31-85-128:~/jars$ sudo kill -13564
Terminated
ubuntu@ip-172-31-85-128:~/jars$ sudo lsof -i :8080
```

Se visualiza si el puerto local está usado y se lanza para que finalmente sea usado dentro de postman. Cabe recalcar que el archivo a subir es el .jar. Hecho en IntelliJ y en conexión con MongoDB. Tiene que ser la versión normal. Entrar a los directorios necesarios para encontrar cada archivo

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-85-128:~$ sudo lsof -i :8080
ubuntu@ip-172-31-85-128:~$ tail -f output.log
tail: cannot open 'output.log' for reading: No such file or directory
tail: no files remaining
ubuntu@ip-172-31-85-128:~$ ls
jars
ubuntu@ip-172-31-85-128:~$ cd jars/
ubuntu@ip-172-31-85-128:~/jars$ tail -f output.log
*****
Description:
Web server failed to start. Port 8080 was already in use.
Action:
Identify and stop the process that's listening on port 8080 or configure this application to listen on another port.
^C
ubuntu@ip-172-31-85-128:~/jars$ sudo lsof -i :8080
ubuntu@ip-172-31-85-128:~/jars$ java -jar Int_DBN-0.0.1-SNAPSHOT.jar
```

```
ubuntu@ip-172-31-85-128:~$ java -jar Int_DBN-0.0.1-SNAPSHOT.jar

:: Spring Boot :: (v2.7.10)

2023-04-13 18:32:59.027 INFO 2442 --- [main] mx.edu.utez.Int_DBN.IntDbnApplication : Starting IntDbnA
pplication v0.0.1-SNAPSHOT using Java 11.0.18 on ip-172-31-85-128 with PID 2442 (/home/ubuntu/Int_DBN-0.0.1-SNAPSHOT
.jar started by ubuntu in /home/ubuntu)
2023-04-13 18:32:59.031 INFO 2442 --- [main] mx.edu.utez.Int_DBN.IntDbnApplication : No active profil
e set, falling back to 1 default profile: "default"
2023-04-13 18:33:00.751 INFO 2442 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Sp
ring Data MongoDB repositories in DEFAULT mode.
2023-04-13 18:33:00.871 INFO 2442 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring
Data repository scanning in 110 ms. Found 4 MongoDB repository interfaces.
2023-04-13 18:33:02.007 INFO 2442 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initializ
ed with port(s): 8080 (http)
2023-04-13 18:33:02.043 INFO 2442 --- [main] o.apache.catalina.core.StandardService : Starting service
[Tomcat]
2023-04-13 18:33:02.043 INFO 2442 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet
engine: [Apache Tomcat/9.0.73]
2023-04-13 18:33:02.267 INFO 2442 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spr
ing embedded WebApplicationContext
2023-04-13 18:33:02.267 INFO 2442 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicat
ionContext: initialization completed in 3080 ms
```

# RESULTADOS

Postman interface showing a GET request to `http://54.164.31.87:8080/marcas/`. The response is a 200 OK status with a JSON body containing three brand entries.

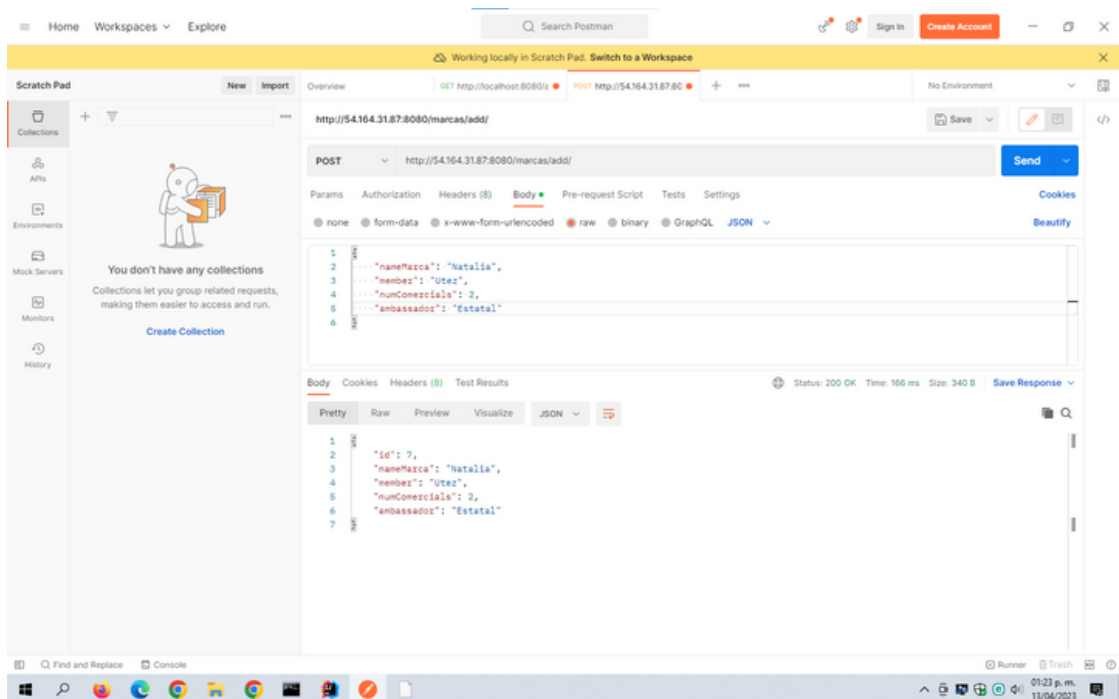
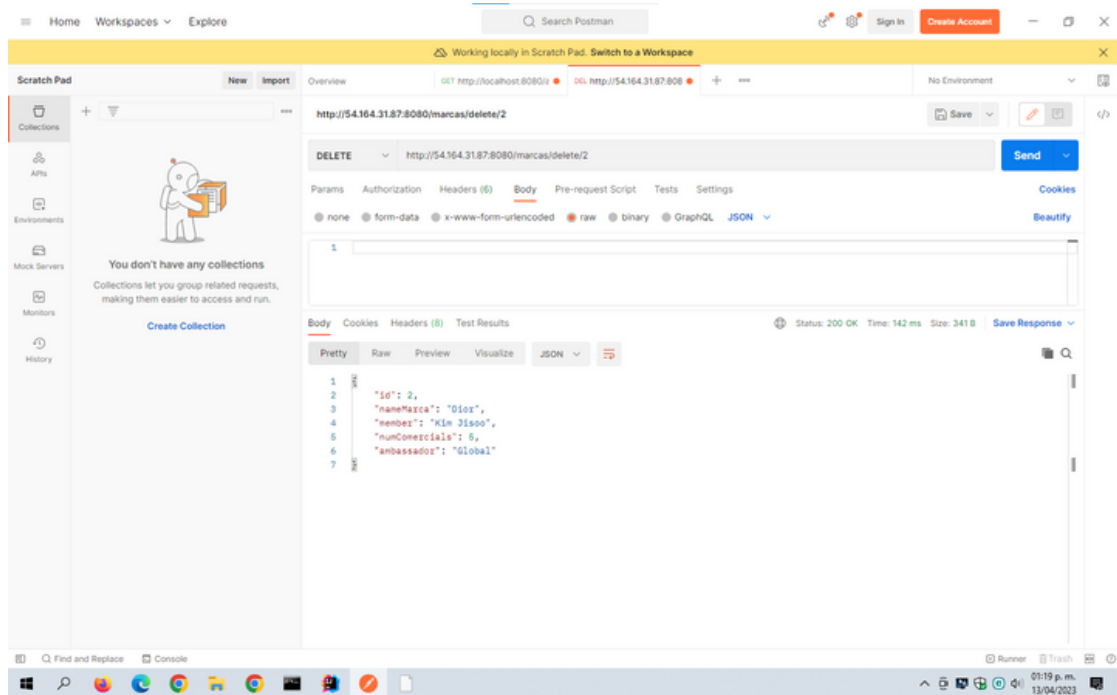
KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "id": 1,
3   "nameMarca": "Channel",
4   "membres": "Kia Jennie",
5   "numComerciales": 19,
6   "ambassadors": "Global"
7 },
8 {
9   "id": 2,
10  "nameMarca": "Bios",
11  "membres": "Kia Jisoo",
12  "numComerciales": 5,
13  "ambassadors": "Global"
14 },
15 {
16  "id": 3,
17  "nameMarca": "Cartier",
18  "membres": "Kia Jisoo",
19  "numComerciales": 2,
20  "ambassadors": "Corea"
21 }
```

Postman interface showing a GET request to `http://54.164.31.87:8080/marcas/3`. The response is a 200 OK status with a JSON body containing a single brand entry.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "id": 3,
3   "nameMarca": "Cartier",
4   "membres": "Kia Jisoo",
5   "numComerciales": 2,
6   "ambassadors": "Corea"
7 }
```



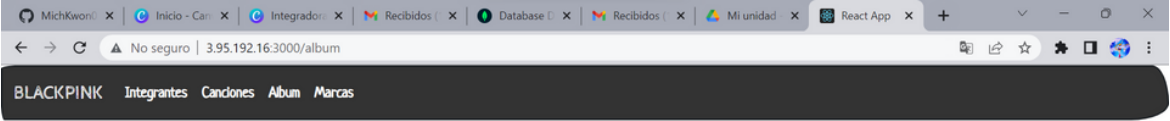


## Integrantes

#	Nombre	Nacionalidad	Cumpleaños	Posicion		
1	Kim Jennie	Coreana	1996-01-16	Rapera Principal	Editar	Eliminar
2	Kim Jisoo	Coreana	1995-01-03	Visual Sub-Vocalista	Editar	Eliminar
3	Roseanne Park	Neozelandesa	1997-01-12	Vocalista Principal	Editar	Eliminar
4	Lalisa Manoban	Tailandesa	1997-03-27	Bailarina Principal	Editar	Eliminar
5	Lalisa Manoban	Tailandesa	1997-03-27	Bailarina Principal	Editar	Eliminar

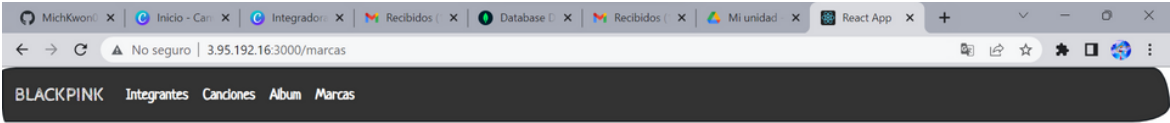
## Canciones

#	Nombre de la canción	Fecha de lanzamiento	Premios	Album
1	Lalisa Manoban	2016-08-08	2	Square One
2	Whistle	2016-08-08	5	Square One
3	Playing with fire	2016-11-06	2	Square Two
4	As If It's Your Last	2017-06-22	2	Sencillo



## Album

#	Nombre del album	Fecha de lanzamiento	Numero de canciones	Ventas
1	BornPink	2022-09-16	8	2141281
2	The album	2020-10-02	8	689066



## Marca

#	Nombre de la marca	Miembro	Numero de comerciales	Embajadora
1	Channel	Kim Jennie	10	Global
2	Cartier	Kim Jisoo	2	Corea
3	Cartier	Lalisa Manoban	8	Mundial
4	Calvin Klein	Kim Jennie	6	Mundial
5	Sulwhasoo	Rose	3	Corea
6	Natalia	Utez	2	Estatal





## CONCLUSIÓN

En conclusión podemos decir que si ayudamos a la empresa del grupo de k-pop blackpink en la parte de la organización y rapidez a la hora de las consultas, ya que no se tendrán que hacer manualmente y para todo el personal de la empresa será más rápido y no perderán tanto tiempo para realizar una consulta.

