NestJS (Backend) y Next.js (Frontend con MUI)

1. Introducción

Se presentan dos frameworks ampliamente utilizados en el desarrollo de aplicaciones web modernas: NestJS para el backend y Next.js con MUI para el frontend. Se presentan sus principales características, ejemplos básicos y la manera en que ambos pueden integrarse en un proyecto completo.

2. Backend con NestJS

NestJS es un framework backend de Node.js construido en TypeScript. Se basa en o en **arquitectura modular**.

- Una aplicación se divide en **módulos** (@Module), que agrupan **controllers, services, providers y repositories**.
- Esto permite tener proyectos **escalables, mantenibles y reutilizables**.
- Ejemplo: puedes tener un módulo AuthModule, otro UserModule, otro ProductModule, etc., y todos pueden conectarse entre sí de manera organizada.

2.1 Componentes principales de NestJS

Controller: define los endpoints que responden a las solicitudes HTTP.

```
@Controller('hello')
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }
}
```

Service: contiene la lógica de negocio.

```
@Injectable()
export class AppService {
  getHello(): string {
    return 'Hola desde el servicio de NestJS!';
  }
}
```

Entity: representa una tabla en la base de datos (usando TypeORM, por ejemplo).

```
@Entity()
export class User {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  nombre: string;

  @Column()
  email: string;
}
```

Repository: gestiona las operaciones sobre la entidad en la base de datos.

```
@EntityRepository(User)
export class UserRepository extends Repository<User> {

  async findAllOrdered(): Promise<User[]> {

    return this.find({ order: { nombre: "ASC" } });

  }}
```

Module: organiza controladores, servicios y repositorios en un bloque lógico.

```
@Module({
  imports: [TypeOrmModule.forFeature([UserRepository])],
  controllers: [AppController],
  providers: [AppService],
```

```
})
export class AppModule {}
```

Página oficial: https://nestjs.com/

---

3. Frontend con Next.js y MUI

Next.js es un framework basado en React que soporta SSR (Server-Side Rendering) y SSG (Static Site Generation). Al combinarlo con MUI (Material UI), se construyen interfaces modernas y responsivas con diseño profesional.

3.1 Componentes principales en Next.js

Page: cada archivo en pages/ es una ruta de la aplicación.

```
export default function Home() {
  return (
    <div>
      <h1>Hola desde Next.js 🚀</h1>
      <p>Esta es una página básica.</p>
    </div>
  );
}
```

Component: bloques reutilizables para la interfaz.

```
function Saludo({ nombre }: { nombre: string }) {
  return <p>Bienvenido, {nombre}!</p>;
}
```

Hooks (useState, useEffect): manejan estado y efectos.

```
import { useState, useEffect } from "react";

export default function EjemploHook() {
  const [mensaje, setMensaje] = useState("Cargando...");

  useEffect(() => {
    setTimeout(() => setMensaje("Listo 🚀"), 2000);
  }, []);
```

```
  return <h2>{mensaje}</h2>;
}
```

MUI Components: facilitan el diseño con Material Design.

```
import { Container, Typography, Button } from
"@mui/material";
import { useState } from "react";

export default function Contador() {
  const [count, setCount] = useState(0);

  return (
    <Container maxWidth="sm" sx={{ mt: 5, textAlign:
"center" }}>
      <Typography variant="h4" gutterBottom>
        Contador: {count}
      </Typography>
      <Button
        variant="contained"
        color="secondary"
        onClick={() => setCount(count + 1)}
      >
        Incrementar
      </Button>
    </Container>
  );
}
```

Página oficial Next.js: https://nextjs.org/

Página oficial MUI: https://mui.com/

4. Integración Backend y Frontend

NestJS y Next.js se integran de la siguiente manera:
1. El usuario accede a la interfaz creada con Next.js + MUI.
2. Next.js realiza peticiones HTTP al backend en NestJS (REST o GraphQL).
3. NestJS recibe la solicitud en su Controller, procesa la lógica en el Service, accede a la base de datos mediante el Repository, y devuelve la respuesta.
4. Next.js muestra la información en la interfaz utilizando componentes de MUI.

Ejemplo simple de flujo - Backend (NestJS):

```
@Get('users')
findAll(): User[] {
 return [{ id: 1, nombre: "Jorge", email: "jorge@test.com"
}];
}
```

Frontend (Next.js):

```
import { useEffect, useState } from "react";

export default function Usuarios() {
 const [users, setUsers] = useState([]);

 useEffect(() => {
  axios.get("http://localhost:3000/users")
   .then(res => res.json())
   .then(data => setUsers(data));
 }, []);

 return (
  <ul>
   {users.map((u) => (
    <li key={u.id}>{u.nombre} - {u.email}</li>
   ))}
  </ul>
 );
}
```

5. Conclusión

NestJS ofrece un backend modular y mantenible con soporte nativo para TypeScript. Next.js + MUI permite construir frontends modernos, rápidos y con diseño consistente. En conjunto, constituyen una solución robusta para aplicaciones web completas, combinando un backend escalable y un frontend atractivo.