

Paso 1: Introducción y tecnologías seleccionadas

En este tutorial crearemos una aplicación backend con Node.js usando TypeScript como lenguaje principal. Para la base de datos emplearemos MySQL, y gestionaremos la comunicación con ella mediante el ORM Sequelize. El objetivo final será ejecutar un Hola Mundo desde el servidor para confirmar que la configuración funciona correctamente.

```
package.json x
Proyecto > Backend > package.json > ...
jugameza, 2 days ago | 1 author (jugameza)
1 {
2   "name": "heisenberg-backend",    "heisenberg": Unknown word.
3   "version": "1.0.0",
4   "description": "",
5   "main": "./dist/cmd/api/main.js",
6   "scripts": {
7     "start": "tsc; node ./dist/cmd/api/main.js",
8     "dev": "concurrently \"npx tsc --watch\" \"nodemon ./dist/cmd/api/main.js\" ",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "repository": {
12    "type": "git",
13    "url": "git+https://github.com/JuanDa72/heisenberg.git"
14  },
15  "author": "",
16  "license": "ISC",
17  "bugs": {
18    "url": "https://github.com/JuanDa72/heisenberg/issues"
19  },
20  "homepage": "https://github.com/JuanDa72/heisenberg#readme",
21  "devDependencies": {
22    "@types/cors": "^2.8.19",
23    "@types/express": "^5.0.3",
24    "@types/morgan": "^1.9.10",
25    "@types/sequelize": "^4.28.20",
26    "@types/node": "^24.9.1",
27    "typescript": "^5.9.3"
28  },
29  "dependencies": {
30    "cors": "^2.8.5",
31    "dotenv": "^16.3.1",
32    "express": "^5.1.0",
33    "morgan": "^1.10.1",
34    "mysql2": "^3.6.5",
35    "sequelize": "^6.25.5"
36  }
37 }
38
```

```
Backend : npm init — Konsole
File Edit View Bookmarks Plugins Settings Help
skywalker@R2D2:~/Documents/unal/ingesoft/heisenberg/Proyecto/Backend$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (heisenberg-backend) █
```

Paso 2: Inicialización del proyecto

Comenzamos creando una carpeta para el proyecto y configuramos TypeScript junto con Express. Esta configuración nos permitirá estructurar el código de manera modular y segura, aprovechando el tipado estático que ofrece TypeScript.

```
Backend : npm exec tsc -- — Konsole
File Edit View Bookmarks Plugins Settings Help
skywalker@R2D2:~/Documents/unal/ingesoft/heisenberg/Proyecto/Backend$ npx tsc --init
Need to install the following packages:
tsc@2.0.4
Ok to proceed? (y) █
```

Paso 3: Instalación de dependencias

Se instalan las librerías necesarias: `express`, `sequelize`, `mysql2` y los tipos de `@types/express` para garantizar compatibilidad con TypeScript.

```
Backend : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
skywalker@R2D2:~/Documents/unal/ingesoft/heisenberg/Proyecto/Backend$ npm i
added 126 packages, and audited 127 packages in 18s

19 packages are looking for funding
  run `npm fund` for details

2 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
skywalker@R2D2:~/Documents/unal/ingesoft/heisenberg/Proyecto/Backend$
```

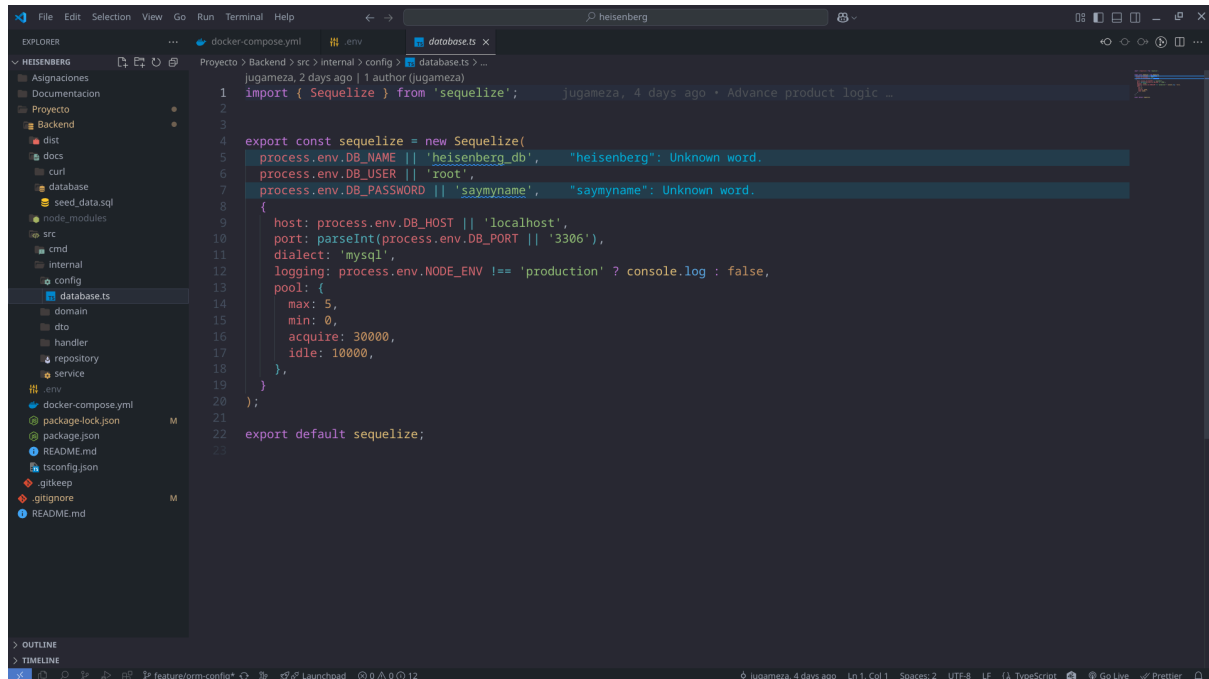
Paso 5: Estructura del proyecto

Se organizan los directorios principales del backend:

```
Backend : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
├── docs
│   ├── curl
│   │   └── test_api.sh
│   └── database
│       └── seed_data.sql
├── package.json
├── package-lock.json
├── README.md
├── src
│   ├── cmd
│   │   └── api
│   │       └── main.ts
│   └── internal
│       ├── config
│       │   └── database.ts
│       ├── domain
│       │   ├── chatbotMessage.model.ts
│       │   ├── chatbotSession.model.ts
│       │   ├── product.model.ts
│       │   └── user.model.ts
│       ├── dto
│       │   ├── product.dto.ts
│       │   └── response.dto.ts
│       ├── handler
│       │   ├── handler.ts
│       │   └── product.handler.ts
│       ├── repository
│       │   ├── product.repository.ts
│       │   └── repository.ts
│       └── service
│           ├── product.service.ts
│           └── service.ts
└── tsconfig.json
```

Paso 6: Configuración de la base de datos

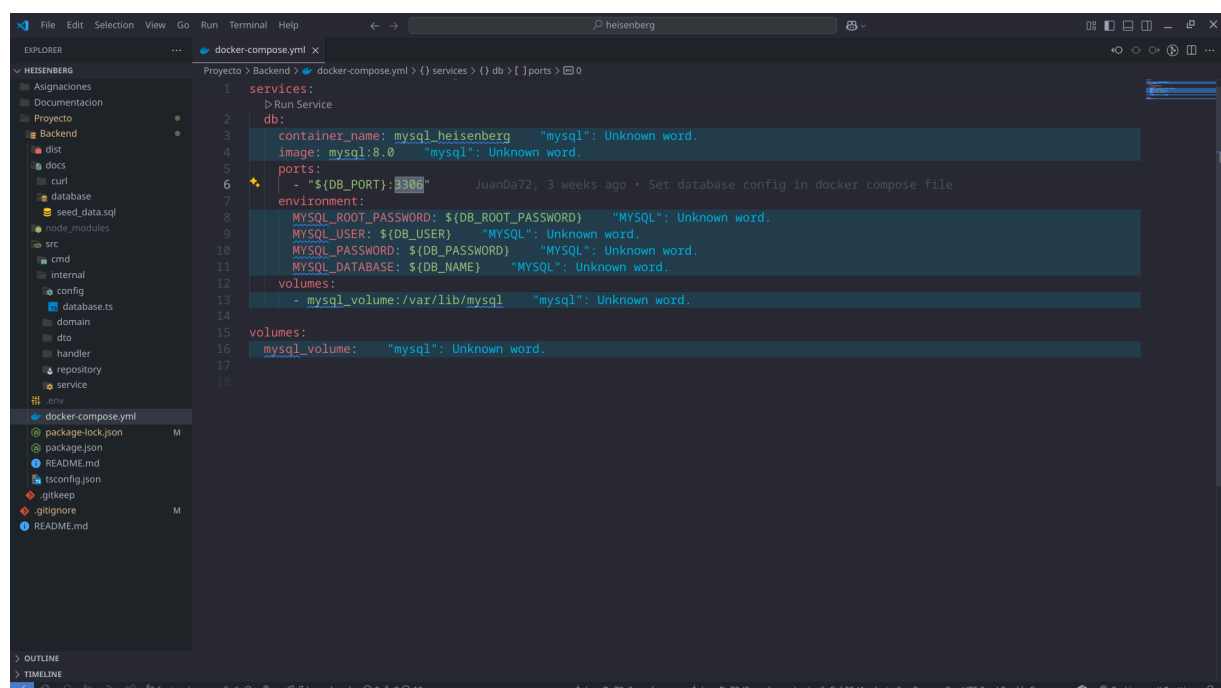
Dentro del directorio `config/`, se define la conexión a MySQL utilizando Sequelize. Aquí se especifican las credenciales, el host y el dialecto correspondiente. Una vez configurado, el ORM puede conectarse y sincronizar las tablas con la base de datos.



```
1 import { Sequelize } from 'sequelize';
2
3
4 export const sequelize = new Sequelize(
5   process.env.DB_NAME || 'heisenberg_db',
6   process.env.DB_USER || 'root',
7   process.env.DB_PASSWORD || 'saymyname',
8   {
9     host: process.env.DB_HOST || 'localhost',
10    port: parseInt(process.env.DB_PORT || '3306'),
11    dialect: 'mysql',
12    logging: process.env.NODE_ENV !== 'production' ? console.log : false,
13    pool: {
14      max: 5,
15      min: 0,
16      acquire: 30000,
17      idle: 10000,
18    },
19  },
20 );
21
22 export default sequelize;
```

Paso 7: Archivo `.yml` de configuración

Se añade un archivo `docker-compose.yml` (según el caso del proyecto) donde se definen las variables de conexión a la base de datos: nombre, usuario, contraseña, host y puerto.



```
1 services:
2   db:
3     container_name: mysql_heisenberg
4     image: mysql:8.0
5     ports:
6       - "${DB_PORT}:3306"
7     environment:
8       MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
9       MYSQL_USER: ${DB_USER}
10      MYSQL_PASSWORD: ${DB_PASSWORD}
11      MYSQL_DATABASE: ${DB_NAME}
12    volumes:
13      - mysql_volume:/var/lib/mysql
14
15 volumes:
16   mysql_volume:
```

Paso 8: Definición del modelo inicial

Se crea un modelo de ejemplo con Sequelize para probar la comunicación con la base de datos. Este modelo representa una tabla simple, que servirá más adelante para verificar que la conexión y sincronización funcionen correctamente.

```
File Edit Selection View Go Run Terminal Help
productmodel.js
1 import { DataTypes } from 'sequelize';
2 import sequelize from '../config/database';
3
4 const Product = sequelize.define('Product', {
5   id: {
6     type: DataTypes.INTEGER,
7     allowNull: false,
8     primaryKey: true,
9   },
10   name: {
11     type: DataTypes.STRING,
12     allowNull: false,
13     unique: true,
14   },
15   type: {
16     type: DataTypes.STRING,
17     allowNull: false,
18   },
19   use_cases: {
20     type: DataTypes.TEXT,
21     allowNull: false,
22   },
23   warnings: {
24     type: DataTypes.TEXT,
25     allowNull: false,
26   },
27   contraindications: {
28     type: DataTypes.TEXT,
29     allowNull: false,
30   },
31   expiration_date: {
32     type: DataTypes.DATEONLY,
33     allowNull: false,
34   },
35   price: {
36     type: DataTypes.DECIMAL(10, 2),
37     allowNull: false,
38     validate: {
39       min: 0,
40     },
41   },
42   stock: {
43     type: DataTypes.INTEGER,
44     allowNull: false,
45     validate: {
46       min: 0,
47     },
48   },
49   created_at: {
50     type: DataTypes.DATE,
51     allowNull: false,
52     defaultValue: DataTypes.NOW,
53   },
54 }, {
55   tableName: 'product',
56   timestamps: false,
57 });
58
59 Product.sync();
60
61 export default Product;
```

```
File Edit Selection View Go Run Terminal Help
productrepository.js
1 import Product from '../domain/product.model';
2 import ProductDTO from '../dto/product.dto';
3 import { Op, UniqueConstraintError, ValidationError } from 'sequelize';
4
5 jugameza, 2 days ago | 1 author (jugameza)
6 export interface ProductRepositoryInterface {
7   findById(id: number): Promise<ProductDTO | null>;
8   findAll(limit?: number, offset?: number): Promise<ProductDTO[]>;
9   create(productData: Omit<ProductDTO, 'id' | 'created_at'>): Promise<ProductDTO>;
10  update(id: number, productData: Partial<Omit<ProductDTO, 'id' | 'created_at'>>): Promise<ProductDTO | null>;
11  delete(id: number): Promise<boolean>;
12  existsById(id: number): Promise<boolean>;
13 }
14
15 jugameza, 2 days ago | 1 author (jugameza)
16 export class ProductRepository {
17   async findById(id: number): Promise<ProductDTO | null> {
18     try {
19       const product = await Product.findPk(id);
20       return product?.get({ plain: true }) as ProductDTO;
21     } catch (error) {
22       console.error('Error findById:', error);
23       throw error;
24     }
25   }
26
27   async findAll(limit?: number, offset?: number): Promise<ProductDTO[]> {
28     // ...
29   }
30
31   async findByName(name: string): Promise<ProductDTO | null> {
32     // ...
33   }
34
35   async create(productData: Omit<ProductDTO, 'id' | 'created_at'>): Promise<ProductDTO> {
36     // ...
37   }
38
39   async update(id: number, productData: Partial<Omit<ProductDTO, 'id' | 'created_at'>>): Promise<ProductDTO | null> {
40     // ...
41   }
42
43   async delete(id: number): Promise<boolean> {
44     // ...
45   }
46
47   async count(): Promise<number> {
48     // ...
49   }
50
51   async existsById(id: number): Promise<boolean> {
52     // ...
53   }
54 }
```

```

File Edit Selection View Go Run Terminal Help
heisenberg

product.service.ts
1 jugameza, 2 days ago | 1 author (jugameza)
2 import { ProductRepositoryInterface } from '../repository/product.repository';
3 import ProductDTO, { CreateProductDTO, UpdateProductDTO } from '../dto/product.dto';
4
5 jugameza, 2 days ago | 1 author (jugameza)
6 export interface ProductServiceInterface {
7   getProduct(id: number): Promise<ProductDTO>;
8   getAllProducts(limit?: number, offset?: number): Promise<ProductDTO[]>;
9   createProduct(productData: CreateProductDTO): Promise<ProductDTO>;
10  updateProduct(id: number, productData: UpdateProductDTO): Promise<ProductDTO>;
11  deleteProduct(id: number): Promise<boolean>;
12 }
13
14 jugameza, 2 days ago | 1 author (jugameza)
15 export class ProductService implements ProductServiceInterface {
16   private productRepository: ProductRepositoryInterface;
17   constructor(productRepository: ProductRepositoryInterface) {
18     this.productRepository = productRepository;
19   }
20   async getProduct(id: number): Promise<ProductDTO> {
21     try {
22       const product = await this.productRepository.findById(id);
23       if (!product) {
24         throw new Error('Product with ID ' + id + ' not found');
25       }
26       return product;
27     } catch (error) {
28       console.error('Error getProduct:', error);
29       throw error;
30     }
31   }
32   async getAllProducts(limit?: number, offset?: number): Promise<ProductDTO[]> {
33     return this.productRepository.getAllProducts(limit, offset);
34   }
35   async createProduct(productData: CreateProductDTO): Promise<ProductDTO> {
36     return this.productRepository.createProduct(productData);
37   }
38   async updateProduct(id: number, productData: UpdateProductDTO): Promise<ProductDTO> {
39     return this.productRepository.updateProduct(id, productData);
40   }
41   async deleteProduct(id: number): Promise<boolean> {
42     return this.productRepository.deleteProduct(id);
43   }
44 }

```

```

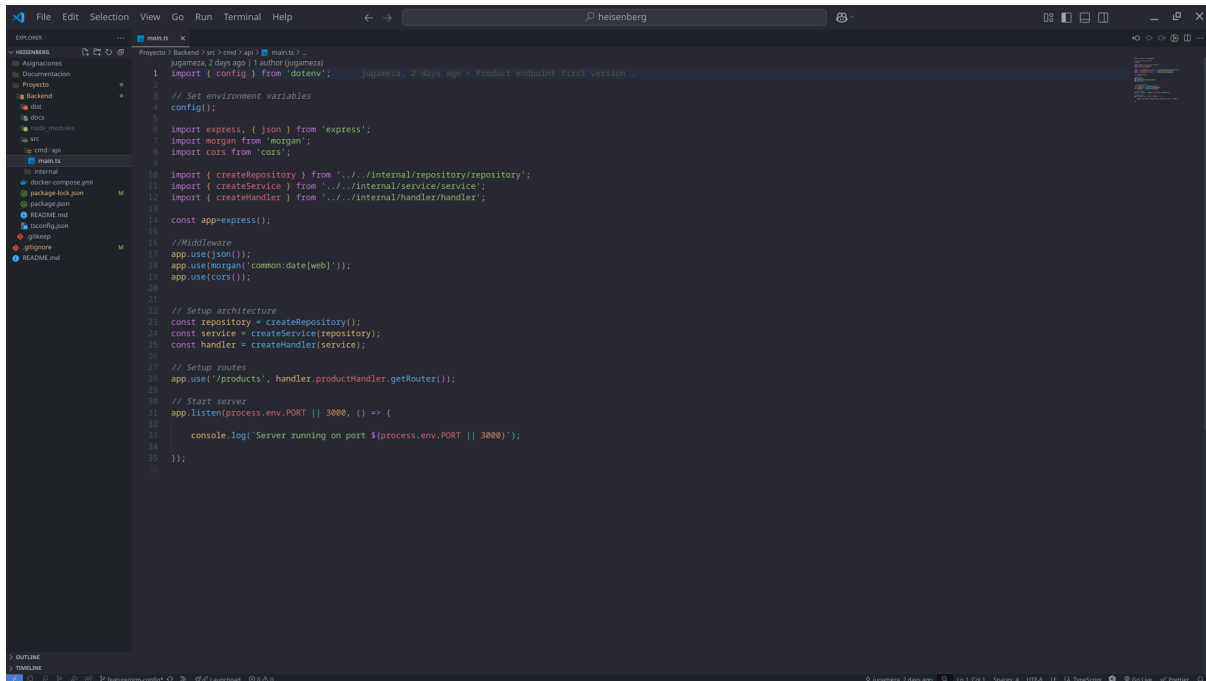
File Edit Selection View Go Run Terminal Help
heisenberg

product.handlers.ts
1 jugameza, 2 days ago | 1 author (jugameza)
2 import { Router, Request, Response } from 'express';
3 import { ProductServiceInterface } from '../service/product.service';
4 import ProductDTO, { CreateProductDTO, UpdateProductDTO } from '../dto/product.dto';
5 import ResponseDTO from '../dto/response.dto';
6
7 jugameza, 2 days ago | 1 author (jugameza)
8 export interface ProductHandlerInterface {
9   setupRoutes(): void;
10  getRouter(): Router;
11 }
12
13 jugameza, 2 days ago | 1 author (jugameza)
14 export class ProductHandler implements ProductHandlerInterface {
15   private productService: ProductServiceInterface;
16   private router: Router;
17   constructor(productService: ProductServiceInterface) {
18     this.productService = productService;
19     this.router = Router();
20   }
21   public setupRoutes(): void {
22     this.router.get('/', this.getAllProducts.bind(this));
23     this.router.get('/:id', this.getProductById.bind(this));
24     this.router.post('/', this.createProduct.bind(this));
25     this.router.put('/:id', this.updateProduct.bind(this));
26     this.router.delete('/:id', this.deleteProduct.bind(this));
27   }
28   // GET /products - Get all products with pagination
29   private async getAllProducts(req: Request, res: Response): Promise<void> {
30     const products = await this.productService.getAllProducts(
31       parseInt(req.query.limit || '10'),
32       parseInt(req.query.offset || '0')
33     );
34     res.json(products);
35   }
36   // GET /products/:id - Get product by ID
37   private async getProductById(req: Request, res: Response): Promise<void> {
38     const product = await this.productService.getProduct(
39       parseInt(req.params.id)
40     );
41     if (!product) {
42       res.status(404).json({ message: 'Product not found' });
43     } else {
44       res.json(product);
45     }
46   }
47   // POST /products - Create a new product
48   private async createProduct(req: Request, res: Response): Promise<void> {
49     const productData = req.body;
50     const product = await this.productService.createProduct(
51       productData
52     );
53     res.status(201).json(product);
54   }
55   // PUT /products/:id - Update an existing product
56   private async updateProduct(req: Request, res: Response): Promise<void> {
57     const { id } = req.params;
58     const productData = req.body;
59     const product = await this.productService.updateProduct(
60       parseInt(id),
61       productData
62     );
63     res.json(product);
64   }
65   // DELETE /products/:id - Delete a product
66   private async deleteProduct(req: Request, res: Response): Promise<void> {
67     const { id } = req.params;
68     const result = await this.productService.deleteProduct(
69       parseInt(id)
70     );
71     if (result) {
72       res.status(204).send();
73     } else {
74       res.status(404).json({ message: 'Product not found' });
75     }
76   }
77   // Get router (encapsulation)
78   public getRouter(): Router {
79     return this.router;
80   }
81 }

```

Paso 9: Configuración del servidor

Se configura Express para levantar un servidor básico en el puerto definido. Además, se incluye una ruta principal (/products) que responderá con un mensaje de texto plano. Esto permitirá comprobar que el servidor está en funcionamiento.

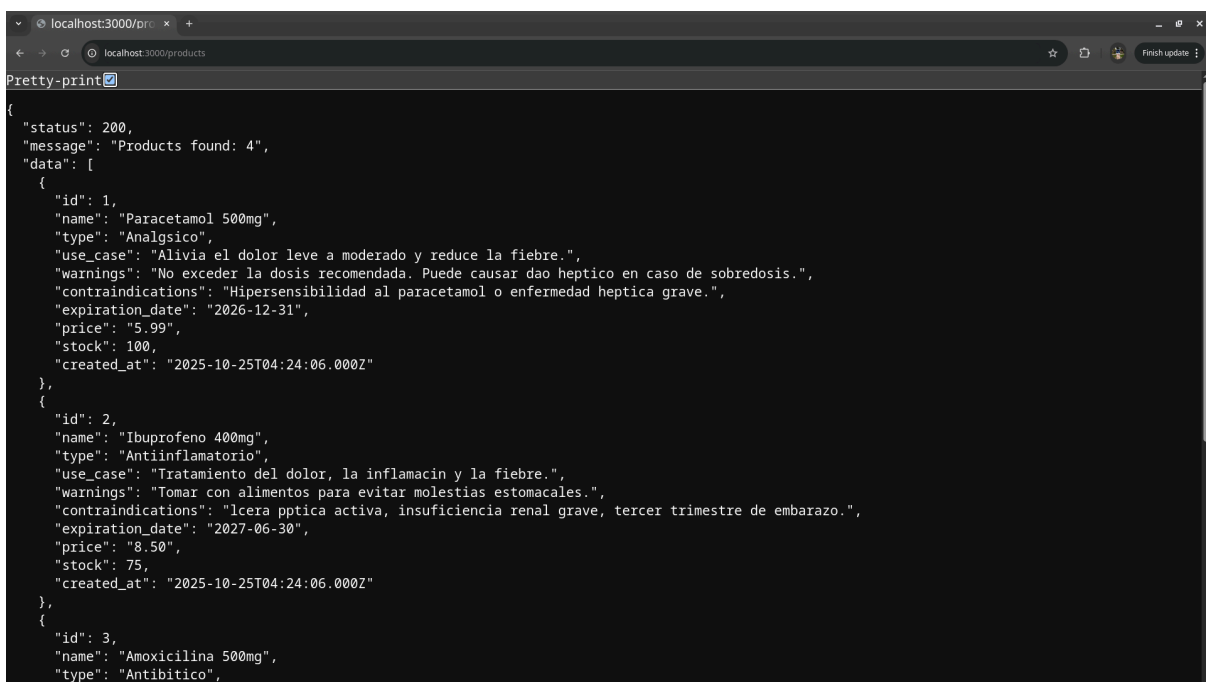


```

1  import { config } from 'dotenv';
2  import { config } from 'dotenv';
3  // Set environment variables
4  config();
5
6  import express, { json } from 'express';
7  import mongoose from 'mongoose';
8  import cors from 'cors';
9
10 import { createRepository } from '../internal/repository/repository';
11 import { createService } from '../internal/service/service';
12 import { createHandler } from '../internal/handler/handler';
13
14 const app = express();
15
16 //Middleware
17 app.use(json());
18 app.use(morgan('common:date[web]'));
19 app.use(cors());
20
21
22 // Setup architecture
23 const repository = createRepository();
24 const service = createService(repository);
25 const handler = createHandler(service);
26
27 // Setup routes
28 app.use('/products', handler.productHandler.getRouter());
29
30 // Start server
31 app.listen(process.env.PORT || 3000, () => {
32   console.log('Server running on port ${process.env.PORT || 3000}');
33 });
  
```

Paso 10: Ejecución del “Hola Mundo”

Finalmente, se ejecuta el servidor y se accede a la ruta principal desde el navegador o mediante herramientas como Postman. Si todo está correctamente configurado, se mostrará el mensaje “Hola Mundo”, confirmando que el backend con Node.js + TypeScript + Sequelize + MySQL está funcionando de forma adecuada.



```

{
  "status": 200,
  "message": "Products found: 4",
  "data": [
    {
      "id": 1,
      "name": "Paracetamol 500mg",
      "type": "Analgsico",
      "use_case": "Alivia el dolor leve a moderado y reduce la fiebre.",
      "warnings": "No exceder la dosis recomendada. Puede causar dao hepatico en caso de sobredosis.",
      "contraindications": "Hipersensibilidad al paracetamol o enfermedad hepatica grave.",
      "expiration_date": "2026-12-31",
      "price": "5.99",
      "stock": 100,
      "created_at": "2025-10-25T04:24:06.000Z"
    },
    {
      "id": 2,
      "name": "Ibuprofeno 400mg",
      "type": "Antiinflamatorio",
      "use_case": "Tratamiento del dolor, la inflamacin y la fiebre.",
      "warnings": "Tomar con alimentos para evitar molestias estomacales.",
      "contraindications": "lceras opticas activas, insuficiencia renal grave, tercer trimestre de embarazo.",
      "expiration_date": "2027-06-30",
      "price": "8.50",
      "stock": 75,
      "created_at": "2025-10-25T04:24:06.000Z"
    },
    {
      "id": 3,
      "name": "Amoxicilina 500mg",
      "type": "Antibitico",
      "use_case": "Tratamiento de infecciones bacterianas.",
      "warnings": "Tomar con alimentos para mejorar la absorcion.",
      "contraindications": "Alergia a la amoxicilina o a otros antibioticos beta-lactamicos.",
      "expiration_date": "2026-06-30",
      "price": "12.00",
      "stock": 50,
      "created_at": "2025-10-25T04:24:06.000Z"
    },
    {
      "id": 4,
      "name": "Aspirina 100mg",
      "type": "Analgsico",
      "use_case": "Alivia el dolor y reduce la fiebre.",
      "warnings": "Evitar el consumo excesivo.",
      "contraindications": "Alergia a la aspirina o a otros AINEs.",
      "expiration_date": "2026-12-31",
      "price": "4.50",
      "stock": 150,
      "created_at": "2025-10-25T04:24:06.000Z"
    }
  ]
}
  
```