

## **Actividad 4**

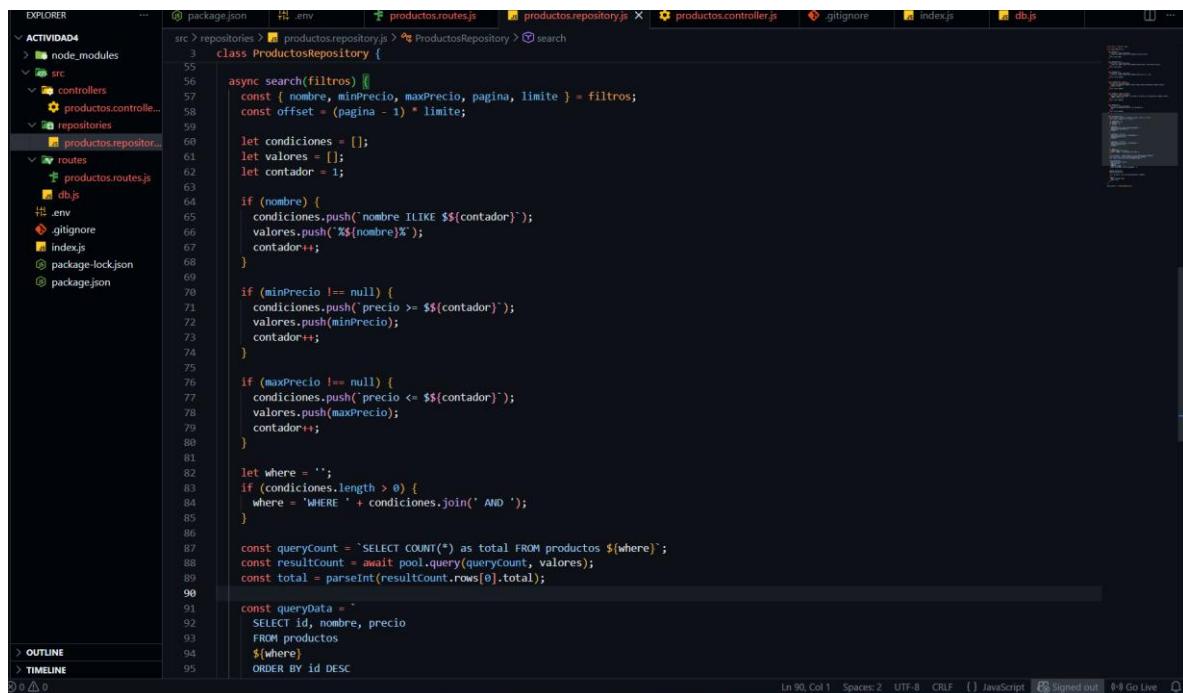
**Abish Abril Santana García**  
**Desarrollo full stack**  
**Universidad Tecmilenio**

## Cómo construiste la query dinámica

En mi archivo productos.repository.js, dentro de la función search:

- Primero, creé un array que se llama condiciones. Ahí voy guardando cada filtro que el usuario pide en la url.
  - Segundo, creé otro array que se llama valores. Aquí guardo los datos reales que el usuario escribió.
  - Tercero, usé un contador que empieza en 1 y va aumentando cada vez que agrego un filtro.
  - Cuarto, al final reviso si hay algo en condiciones. Si tiene, las junto con AND para formar el WHERE. Si no hay filtros, no pongo WHERE.
1. Para la paginación, recibo page y limit, calculo OFFSET y los uso en la consulta.

Así mi consulta funciona con cualquier combinación de filtros que el usuario quiera usar.



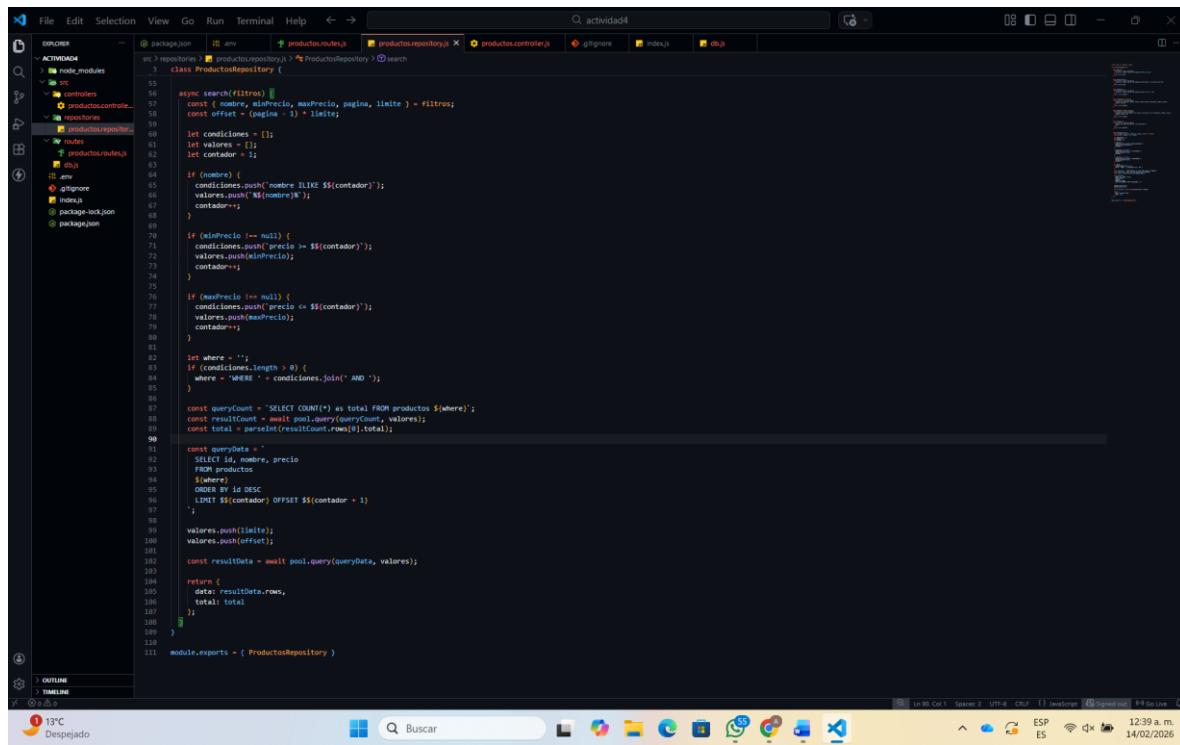
```
class ProductosRepository {  
  async search(filtros) {  
    const { nombre, minPrecio, maxPrecio, pagina, limite } = filtros;  
    const offset = (pagina - 1) * limite;  
  
    let condiciones = [];  
    let valores = [];  
    let contador = 1;  
  
    if (nombre) {  
      condiciones.push(`nombre ILIKE ${contador}`);  
      valores.push(`%${nombre}%`);  
      contador++;  
    }  
  
    if (minPrecio !== null) {  
      condiciones.push(`precio >= ${contador}`);  
      valores.push(minPrecio);  
      contador++;  
    }  
  
    if (maxPrecio !== null) {  
      condiciones.push(`precio <= ${contador}`);  
      valores.push(maxPrecio);  
      contador++;  
    }  
  
    let where = '';  
    if (condiciones.length > 0) {  
      where = 'WHERE ' + condiciones.join(' AND ');  
    }  
  
    const queryCount = `SELECT COUNT(*) as total FROM productos ${where}`;  
    const resultCount = await pool.query(queryCount, valores);  
    const total = parseInt(resultCount.rows[0].total);  
  
    const queryData = `  
      SELECT id, nombre, precio  
      FROM productos  
      ${where}  
      ORDER BY id DESC`  
  }  
}
```

## Cómo evitaste SQL Injection

En mi archivo productos.repository.js, dentro de la función search:

- Primero, nunca concatene los datos directamente en la consulta.
- Segundo, utilicé los parámetros \$1 para recibir valores de forma segura. Por ejemplo, en lugar de pegar el texto, escribo "WHERE nombre ILIKE \$1".
- Tercero, todos los datos que el usuario escribe los guardé en el array valores. Así la consulta queda limpia y los datos van aparte.
- Cuarto, la librería pg que usamos para conectar con la base de datos se encarga automáticamente de escapar cualquier carácter especial.
- Quinto, en mi controlador también validé que page y limit sean números antes de usarlos.

De esta forma, aunque alguien intente meter código malicioso en la búsqueda, la base de datos lo interpreta solo como texto normal y no como parte de la consulta.



```
File Edit Selection View Go Run Terminal Help ← → 🔍 actividad4
ACTIVIDAD
src > repositories > productos.repository.js ↗ ProduktoRepository : search
class ProduktoRepository {
 55   static async search(filtros) {
 56     const { nombre, minPrecio, maxPrecio, pagina, limite } = #filtros;
 57     const offset = (pagina - 1) * limite;
 58
 59     let condiciones = [];
 60     let valores = [];
 61     let contador = 1;
 62
 63     if (nombre) {
 64       condiciones.push(`nombre ILIKE ${contador}`);
 65       valores.push(`%${nombre}%`);
 66       contador++;
 67     }
 68
 69     if (minPrecio != null) {
 70       condiciones.push(`precio >= ${contador}`);
 71       valores.push(minPrecio);
 72       contador++;
 73     }
 74
 75     if (maxPrecio != null) {
 76       condiciones.push(`precio <= ${contador}`);
 77       valores.push(maxPrecio);
 78       contador++;
 79     }
 80
 81     let where = '';
 82     if (condiciones.length > 0) {
 83       where = `WHERE ${condiciones.join(" AND ")}`;
 84     }
 85
 86     const queryCount = 'SELECT COUNT(*) as total FROM productos ${where}';
 87     const resultCount = await pool.query(queryCount, valores);
 88     const total = parseInt(resultCount.rows[0].total);
 89
 90     const queryData =
 91       `SELECT id, nombre, precio
 92        FROM productos
 93        ${where}
 94        ORDER BY id DESC
 95        LIMIT ${contador} OFFSET ${contador + 1}`;
 96
 97     valores.push(id);
 98     valores.push(offset);
 99
100    const resultData = await pool.query(queryData, valores);
101
102    return {
103      data: resultData.rows,
104      total
105    };
106  }
107}
108
109 module.exports = { ProduktoRepository }
```

Link de postman <https://documenter.getpostman.com/view/51907016/2sBXcBnheh>