INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Ingeniería en Sistemas Computacionales

SISTEMAS DISTRIBUIDOS
7CM1
PRÁCTICA 8: Micro servicios

Alumno:

- González González Ian Imanol

Dr. Chadwick Carreto Arellano

Noviembre-2025

Introducción

Los microservicios son una arquitectura de desarrollo de software que consiste en dividir una aplicación en componentes pequeños, independientes y autónomos. Cada microservicio se encarga de una funcionalidad específica del sistema y puede desarrollarse, desplegarse y escalarse de manera separada.

A diferencia de las arquitecturas monolíticas, donde todo el sistema depende de un único bloque de código, los microservicios promueven la modularidad y la independencia entre componentes. Esto permite una mayor flexibilidad en el mantenimiento, la integración de nuevas tecnologías y la distribución de tareas entre equipos de desarrollo.

Cada microservicio suele comunicarse con los demás mediante API REST o mensajería ligera, utilizando protocolos como HTTP o JSON. Además, puede estar desarrollado en diferentes lenguajes de programación o ejecutarse en distintos entornos, siempre que respeten las interfaces de comunicación definidas.

Entre sus principales ventajas se encuentran la escalabilidad, la resiliencia ante fallos, la facilidad de actualización y la rapidez en el despliegue continuo. Sin embargo, esta arquitectura también implica desafíos, como la gestión de la comunicación entre servicios, la sincronización de datos y la implementación de mecanismos de seguridad y monitoreo distribuidos.

Problemática

Al desarrollar aplicaciones basadas en microservicios, surgen diversos retos relacionados con la comunicación, la integración y la gestión distribuida de los componentes del sistema:

- **Comunicación entre servicios:** Cada microservicio opera de forma independiente, lo que requiere establecer mecanismos eficientes y seguros para el intercambio de información entre ellos, normalmente mediante API REST o colas de mensajes.
- **Sincronización y consistencia de datos:** Dado que cada servicio puede tener su propia base de datos, mantener la coherencia y actualización de la información entre los distintos módulos representa un desafío importante.
- **Despliegue y mantenimiento:** Coordinar la implementación, actualización y monitoreo de múltiples servicios puede aumentar la complejidad operativa del sistema.
- **Gestión de fallos:** Si un servicio falla, es necesario garantizar que los demás continúen funcionando sin afectar la disponibilidad total de la aplicación.

La problemática central consiste en lograr que la arquitectura de microservicios funcione de manera coherente, escalable y confiable, asegurando una correcta comunicación entre los módulos y manteniendo la integridad de los datos, sin comprometer el rendimiento ni la estabilidad del sistema.

Solución implementada

Para abordar los retos identificados, se diseñó e implementó una arquitectura basada en microservicios, donde cada módulo cumple una función específica y se comunica con los demás mediante peticiones HTTP.

En esta práctica se desarrollaron dos microservicios principales:

- **Servicio de productos:** encargado de administrar la información de los productos disponibles.
- **Servicio de reseñas:** responsable de gestionar las opiniones y calificaciones asociadas a cada producto.

Ambos servicios fueron construidos utilizando Node.js con el framework Express, aplicando el middleware CORS para permitir la comunicación entre orígenes distintos. Cada servicio opera de manera independiente en un puerto diferente, lo que facilita su mantenimiento, escalabilidad y despliegue.

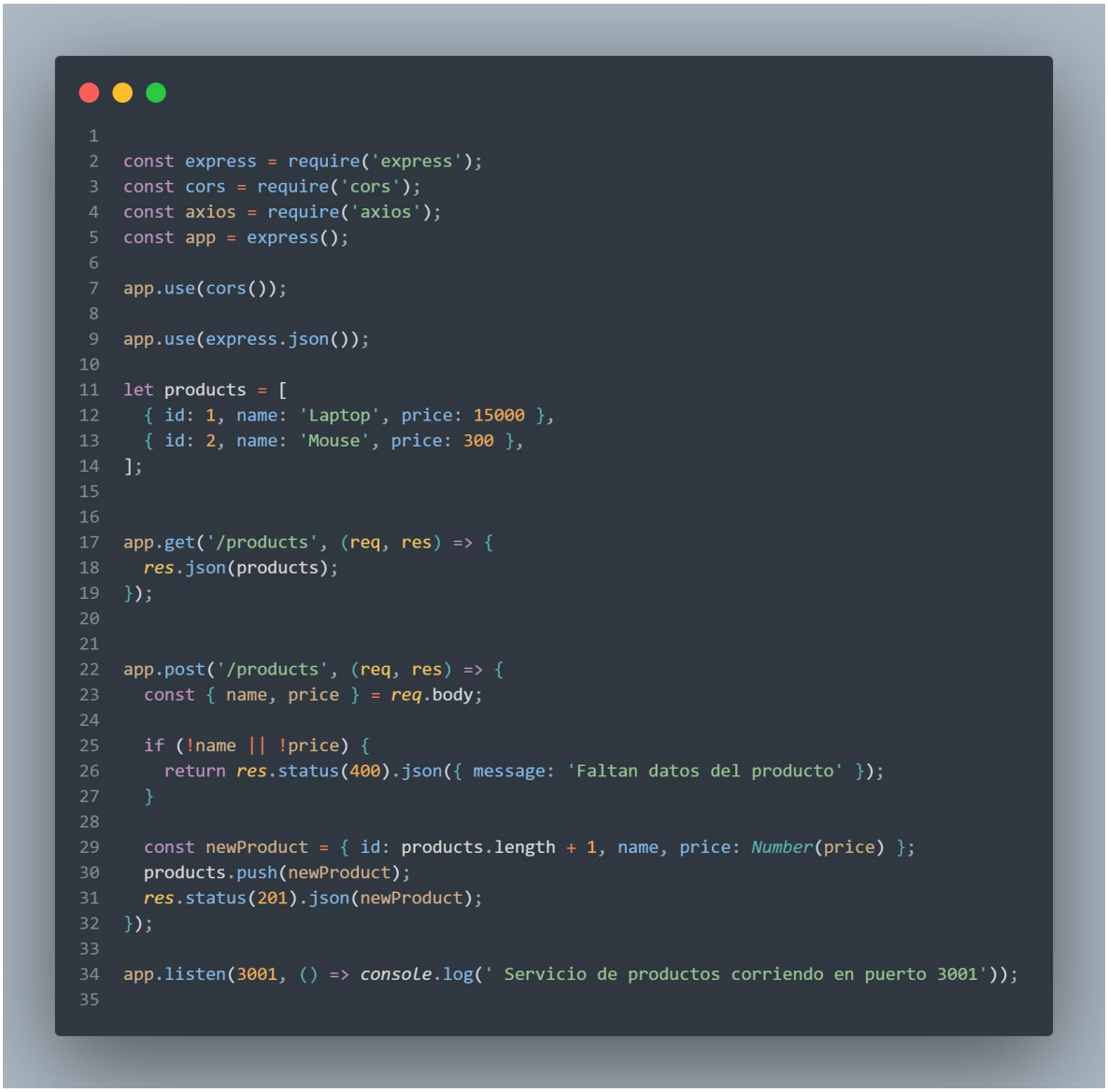
Adicionalmente, se implementó una interfaz web desarrollada con HTML, CSS y JavaScript, que actúa como cliente y consume las APIs de ambos microservicios. Desde esta interfaz, el usuario puede visualizar los productos, agregar nuevos y registrar reseñas.

La comunicación entre los módulos se realiza mediante solicitudes REST, garantizando una separación clara entre las responsabilidades de cada servicio. Con esta implementación se logró demostrar cómo los microservicios pueden colaborar entre sí para construir sistemas distribuidos más flexibles, mantenibles y escalables.

Server reviews

```
1
2  const express = require('express');
3  const cors = require('cors');
4  const axios = require('axios');
5  const app = express();
6
7
8  app.use(cors());
9
10 app.use(express.json());
11
12 let reviews = [
13   { id: 1, productId: 1,  comment: 'Excelente laptop', rating: 5 },
14   { id: 2, productId: 2, comment: 'Buen mouse por el precio', rating: 4 },
15 ];
16
17
18 app.get('/reviews', (req, res) => {
19   res.json(reviews);
20 });
21
22
23 app.get('/reviews/with-products', async (req, res) => {
24   try {
25     const { data: products } = await axios.get('http://localhost:3001/products');
26     const combined = reviews.map(r => ({
27       ...r,
28       product: products.find(p => p.id === r.productId),
29     }));
30     res.json(combined);
31   } catch (error) {
32     res.status(500).json({ message: 'Error obteniendo productos', error: error.message });
33   }
34 });
35
36 app.get('/reviews/:productId', (req, res) => {
37   const productId = parseInt(req.params.productId);
38   const productReviews = reviews.filter(r => r.productId === productId);
39   res.json(productReviews);
40 });
41
42 app.post('/reviews', (req, res) => {
43   let { productId, comment, rating } = req.body;
44     productId = parseInt(productId);
45     rating = parseInt(rating);
46   if (!productId || !comment || !rating) {
47     return res.status(400).json({ message: 'Faltan datos en la reseña' });
48   }
49
50   const newReview = {
51     id: reviews.length + 1,
52     productId,
53     comment,
54     rating,
55   };
56
57   reviews.push(newReview);
58   res.status(201).json(newReview);
59 });
60
61 app.listen(3002, () => console.log('Servicio de reseñas corriendo en puerto 3002'));
62
```

Server productos



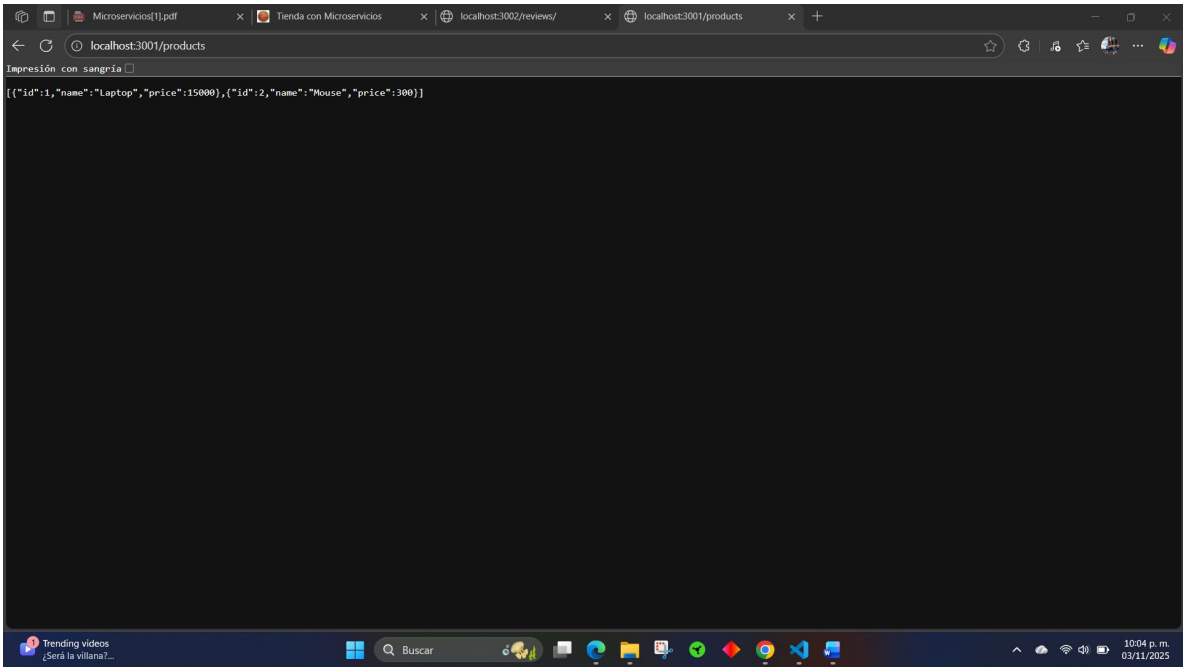
```
1
2  const express = require('express');
3  const cors = require('cors');
4  const axios = require('axios');
5  const app = express();
6
7  app.use(cors());
8
9  app.use(express.json());
10
11 let products = [
12   { id: 1, name: 'Laptop', price: 15000 },
13   { id: 2, name: 'Mouse', price: 300 },
14 ];
15
16
17 app.get('/products', (req, res) => {
18   res.json(products);
19 });
20
21
22 app.post('/products', (req, res) => {
23   const { name, price } = req.body;
24
25   if (!name || !price) {
26     return res.status(400).json({ message: 'Faltan datos del producto' });
27   }
28
29   const newProduct = { id: products.length + 1, name, price: Number(price) };
30   products.push(newProduct);
31   res.status(201).json(newProduct);
32 });
33
34 app.listen(3001, () => console.log(' Servicio de productos corriendo en puerto 3001'));
35
```

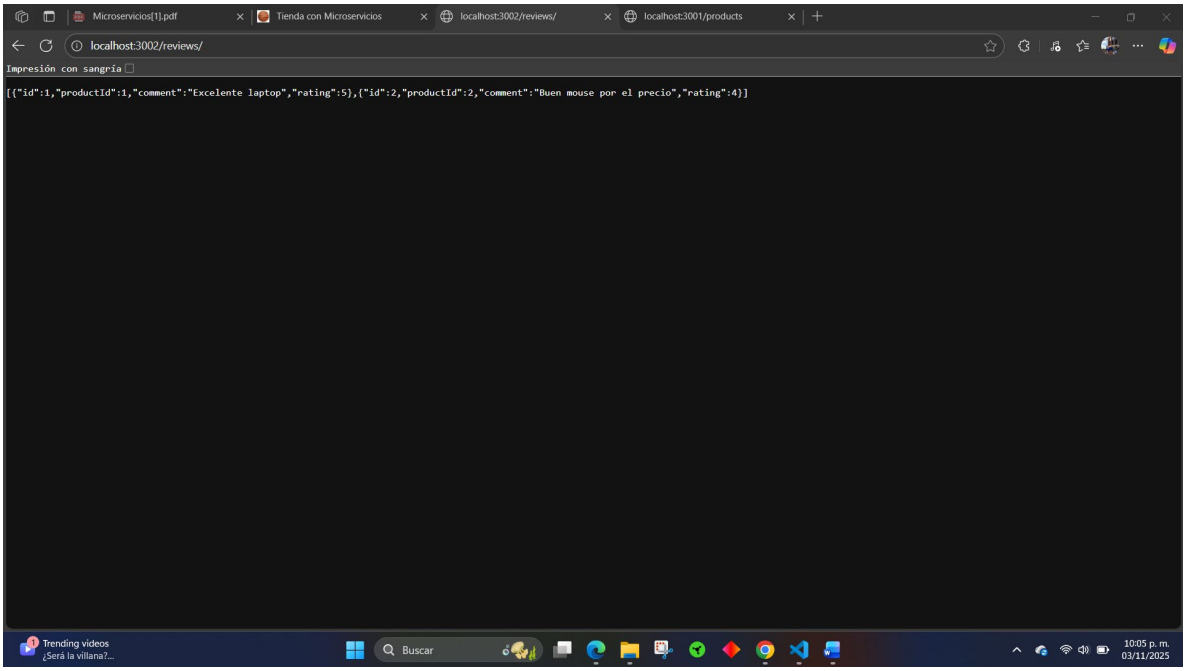
Script js

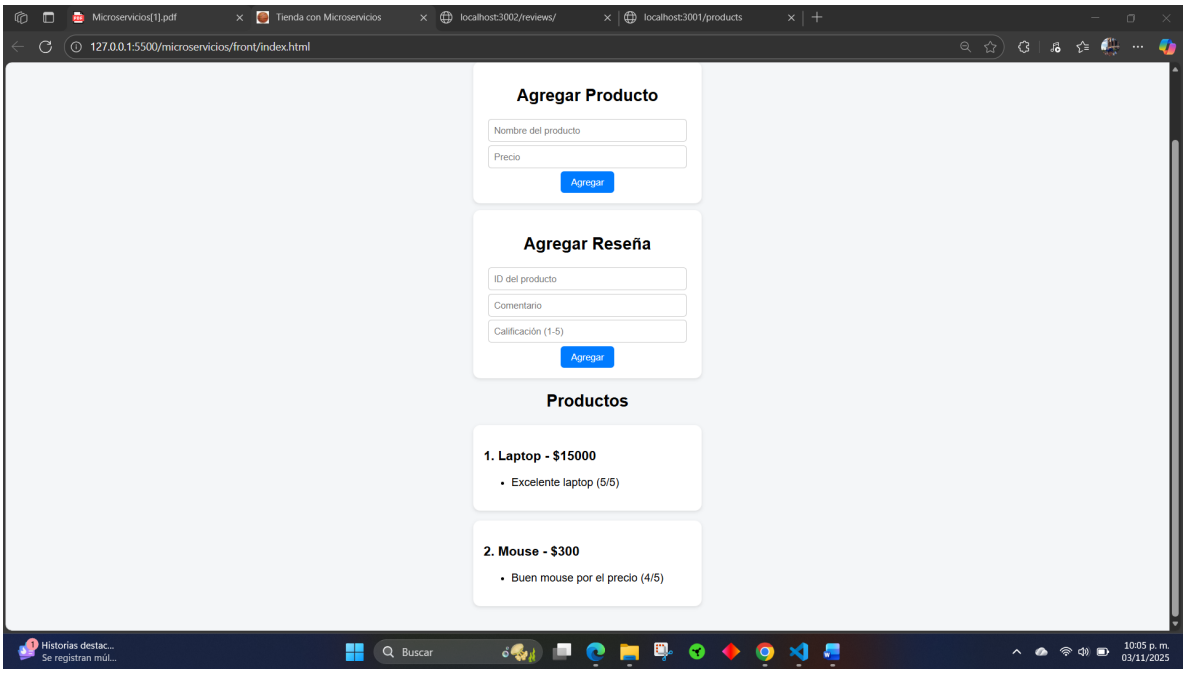
```
const PRODUCT_URL = 'http://localhost:3001/products';
const REVIEW_URL = 'http://localhost:3002/reviews';

async function loadProducts() {
  const res = await fetch(PRODUCT_URL);
  const products = await res.json();

  const container = document.getElementById('products');
  container.innerHTML = '';

  for (const p of products) {
    const resReviews = await fetch(`${REVIEW_URL}/${p.id}`);
    const reviews = await resReviews.json();

    const div = document.createElement('div');
    div.className = 'product';
    div.innerHTML = `
      <h3>${p.id}. ${p.name} - $${p.price}</h3>
      <ul>
        ${reviews.length ? reviews.map(r => `<li>${r.comment} (${r.rating}/5)</li>`).join('') : '<li>Sin reseñas</li>'}
      </ul>
    `;
    container.appendChild(div);
  }
}

async function addProduct() {
  const name = document.getElementById('name').value;
  const price = document.getElementById('price').value;

  const res = await fetch(PRODUCT_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ name, price })
  });

  if (res.ok) {
    alert('Producto agregado!');
    loadProducts();
  } else {
    alert('Error al agregar producto');
  }
}

async function addReview() {
  const productId = document.getElementById('productId').value;
  const comment = document.getElementById('comment').value;
  const rating = document.getElementById('rating').value;

  const res = await fetch(REVIEW_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ productId, comment, rating })
  });

  if (res.ok) {
    alert('Reseña agregada!');
    loadProducts();
  } else {
    alert('Error al agregar reseña');
  }
}

loadProducts();

```

Resultados

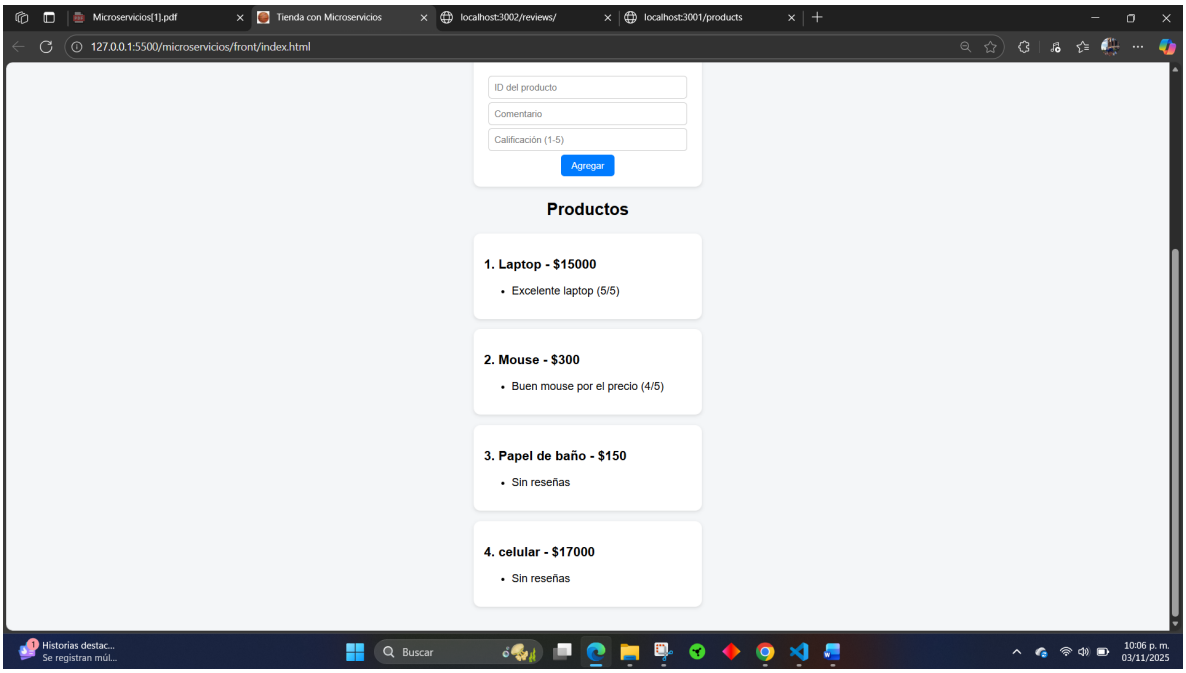
Al iniciar los servidores

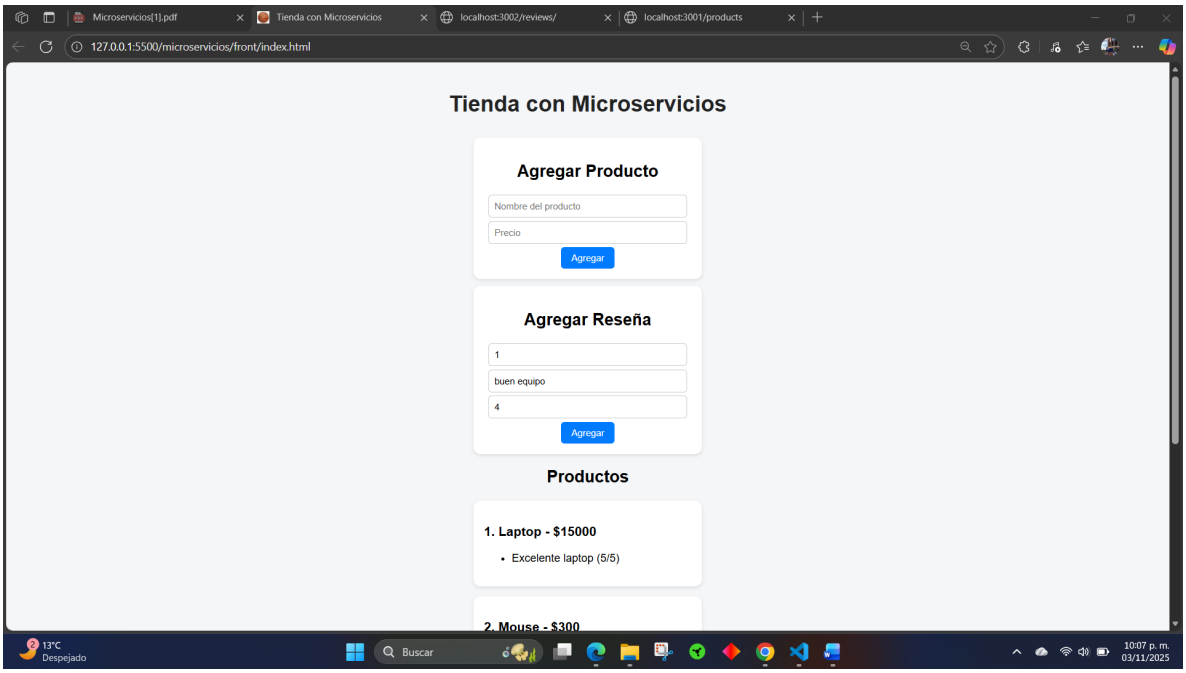


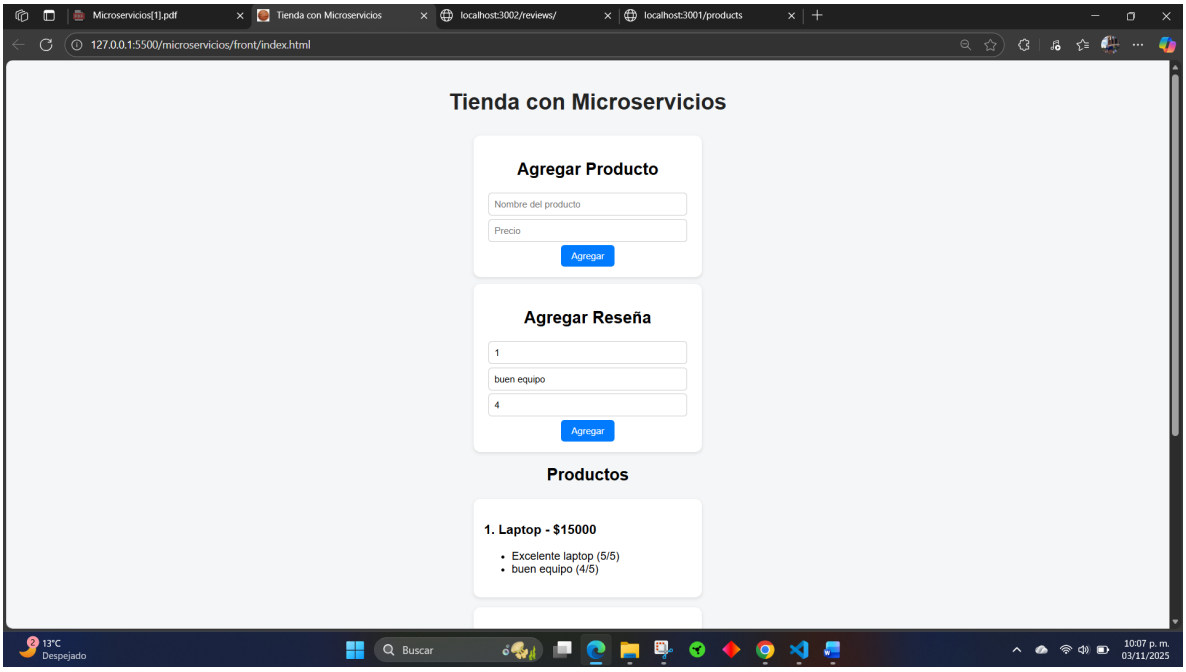




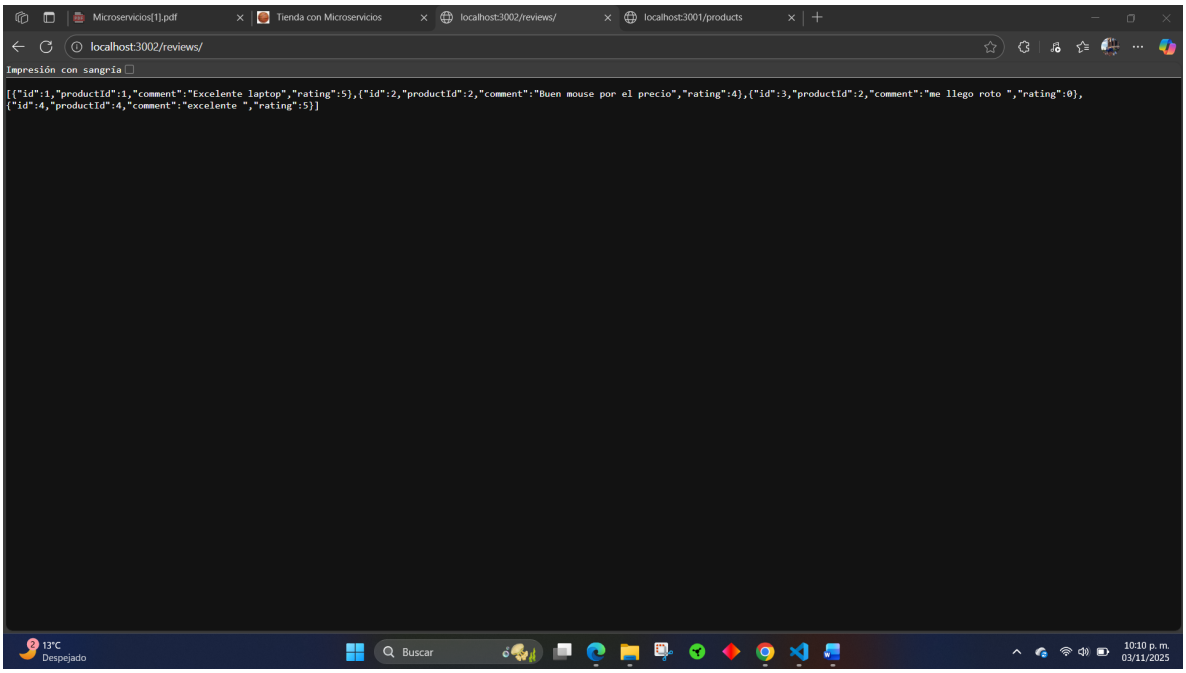
Insertamos productos y reviews

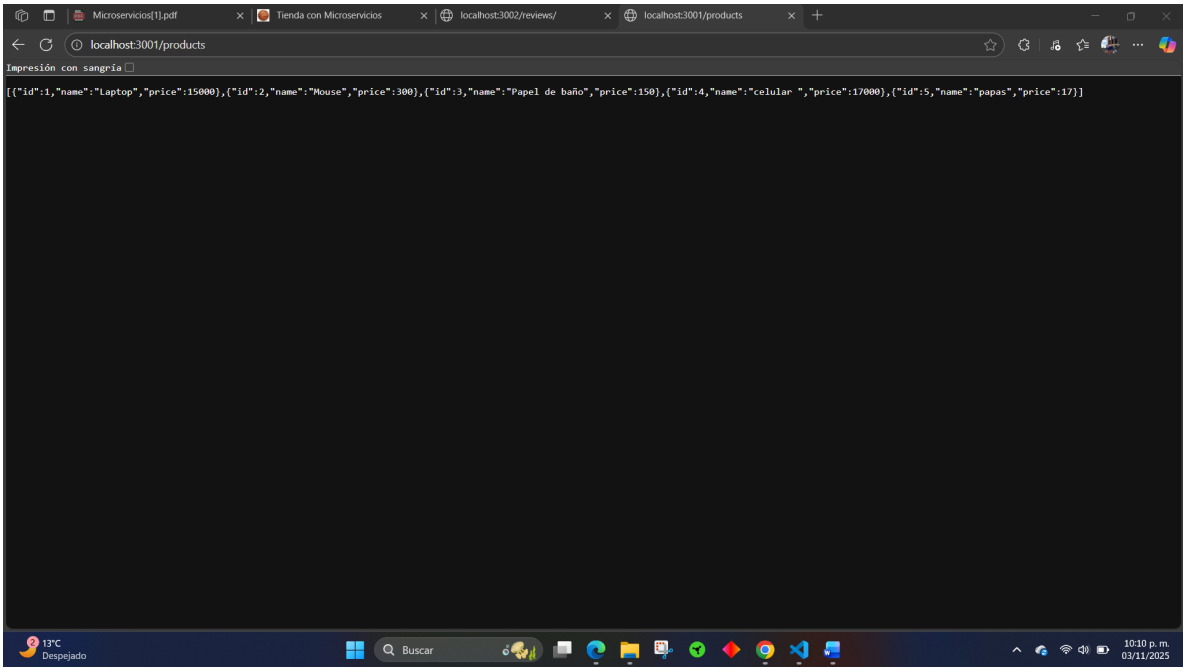


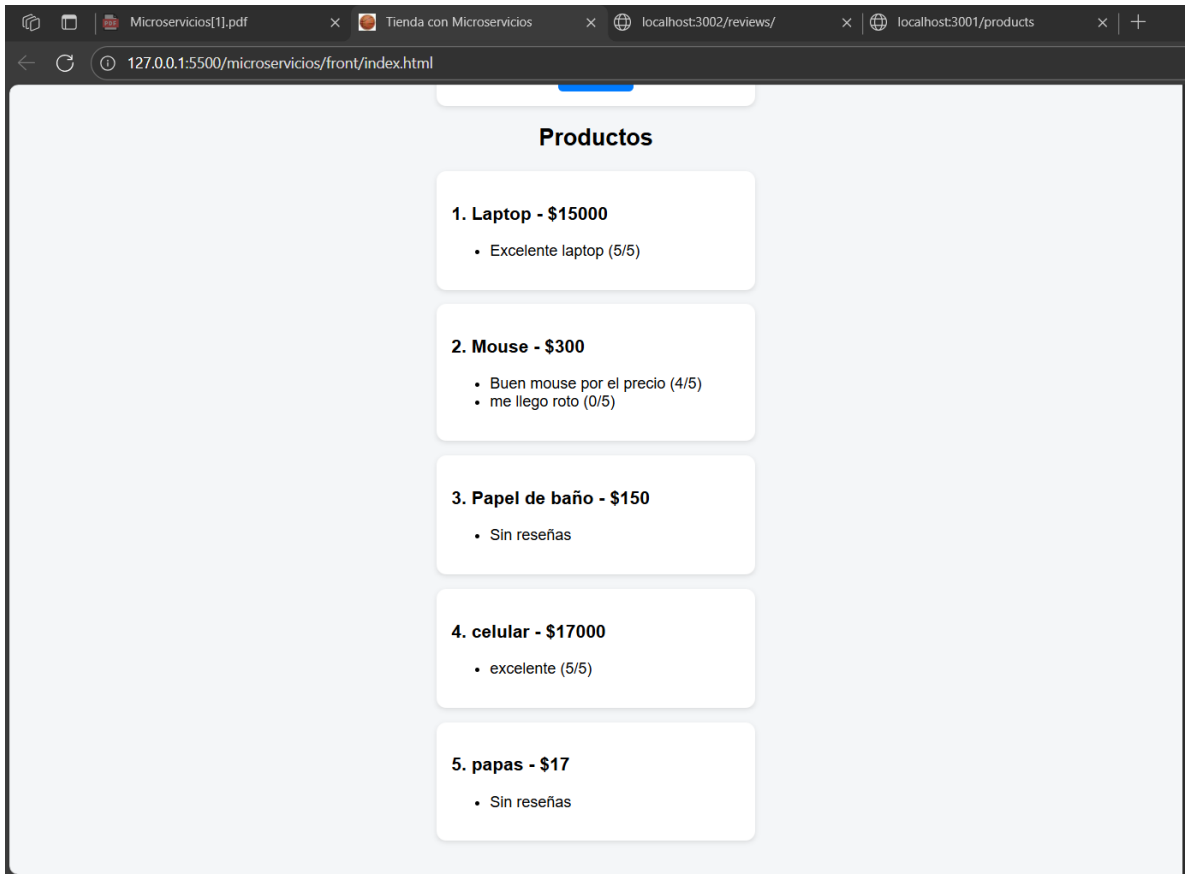




Seguimos agregando


Microservicios[1].pdf
Tienda con Microservicios
localhost:3002/reviews/
localhost:3001/products
localhost:3002/reviews/
Impresión con sangría
[{"id":1,"productId":1,"comment":"Excelente laptop","rating":5},{"id":2,"productId":2,"comment":"Buen mouse por el precio","rating":4},{"id":3,"productId":2,"comment":"me llego roto ","rating":0},
{"id":4,"productId":4,"comment":"excelente ","rating":5}]
13°C
Despejado
Buscar
10:10 p. m.
03/11/2025


Microservicios[1].pdf
Tienda con Microservicios
localhost:3002/reviews/
localhost:3001/products
localhost:3001/products
Impresión con sangría
[{"id":1,"name":"Laptop","price":15000},{"id":2,"name":"Mouse","price":300},{"id":3,"name":"Papel de baño","price":150},{"id":4,"name":"celular ","price":17000},{"id":5,"name":"papas","price":17}]
13°C
Despejado
Buscar
10:10 p. m.
03/11/2025


Microservicios[1].pdf
Tienda con Microservicios
localhost:3002/reviews/
localhost:3001/products
127.0.0.1:5500/microservicios/front/index.html
Productos
1. Laptop - $15000
Excelente laptop (5/5)
2. Mouse - $300
Buen mouse por el precio (4/5)
me llego roto (0/5)
3. Papel de baño - $150
Sin reseñas
4. celular - $17000
excelente (5/5)
5. papas - $17
Sin reseñas

Conclusión

La arquitectura de microservicios permite dividir una aplicación en servicios pequeños, independientes y especializados, lo que facilita su desarrollo, mantenimiento y escalabilidad. Cada microservicio puede ser implementado, actualizado y desplegado de forma autónoma, comunicándose con los demás mediante APIs. Esta separación de responsabilidades mejora la resiliencia del sistema, permite integrar nuevas tecnologías y facilita la colaboración entre equipos. Sin embargo, también requiere una buena gestión de la comunicación, la consistencia de datos y el monitoreo de los servicios distribuidos. En general, los microservicios ofrecen una forma flexible y moderna de construir aplicaciones complejas y adaptables a las necesidades del negocio.