UNIVERSIDAD AUTÓNOMA DE ZACATECAS "Francisco García Salinas"
Área de Ingeniería y Tecnologías
Unidad Académica de Ingeniería Eléctrica

PROGRAMA DE INGENIERÍA DE SOFTWARE
Arquitectura de Software

Tarea de Implementación II
Implementación de una Arquitectura

ALUMNOS:
Andrés Contreras Sánchez
Gerardo Rivas Delgado

DOCENTE:
Dra. Julieta G. Rodríguez Ruiz

SEMESTRE Y GRUPO:
6to SEMESTRE, GRUPO C

Enero – Junio 2024

Introducción

Esta tarea tiene como objetivo el adquirir conocimientos prácticos sobre diferentes patrones arquitecturales que hemos visto a través del curso. Para esta tarea se es necesario crear un “Hola Mundo” implementando una arquitectura de las mostradas en el documento, correspondientes a:

- Arquitectura de Microservicios
- Arquitectura Serverless
- Arquitectura Basada en Eventos
- Arquitectura Orientada a Servicios
- Arquitectura de Message Brokers

Se eligió una arquitectura para desarrollar nuestro sistema que para mostrar información se tuvo un “Hola Mundo” pero en forma de tienda demostrando los microservicios como elementos de la tienda (pago, consulta a clientes, etc.).
En este documento se agrega la definición de la arquitectura usada, la presentación de nuestro sistema con su arquitectura, funcionamiento, estructuración y diagramas de funcionamiento.

Definición de la arquitectura

Definición

Para este trabajo nosotros decidimos usar la arquitectura de Microservicios, para mayor flexibilidad de trabajo.
La arquitectura de microservicios es un enfoque para diseñar y desarrollar aplicaciones en el que la funcionalidad de la aplicación se divide en pequeños servicios independientes con distintas funcionalidades. Cada uno de estos servicios se centra en una funcionalidad específica y se puede administrar de manera autónoma para una mayor comodidad y/o flexibilidad de mantenimiento. Cada microservicio opera de manera independiente, con su propia base de datos y lógica

de negocio para cumplir dichas funcionalidades, sin embargo en este programa no usamos una base de datos como tal para cada servicio, y que se trataba de un “Hola Mundo” sencillo. Una ventaja de esta arquitectura es que si un servicio específico recibe muchas más solicitudes que otros, se puede escalar solo ese servicio para que todo el sistema se mantenga estable.

Explicación de la arquitectura

El diagrama general es este:

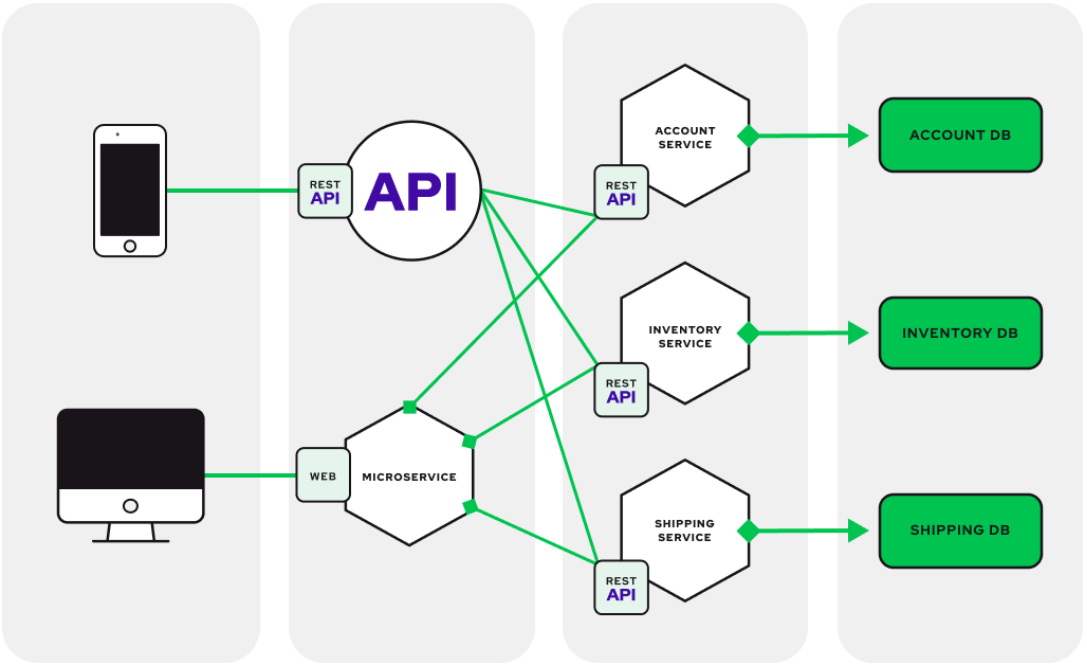

Figura 1. Estructura de una arquitectura de microservicios.

Podemos ver que en el diagrama se tienen diferentes elementos y así mismo capas o niveles del trabajo. De izquierda a derecha se tiene la primera capa de los usuarios que usan el sistema mostrándose como computadoras y dispositivos móviles. Luego se puede ver las que se llaman “API Gateway” en el segundo nivel, estas serían las conexiones de nuestros microservicios para presentarlos al usuario, puede ser un servicio web de página o una interfaz de usuario para mostrar la lista de funcionalidades a las que el usuario puede acceder, esto se comunica mediante una API a los microservicios para envíar los datos del usuario a la lógica de cada uno de los microservicios para operar bajo lo que se deba hacer, por último si la lógica lo requiere se hace una consulta a las bases de datos de cada microservicio para recuperar los datos consultados.

Pros y Contras:

Pros de la Arquitectura de Microservicios:

1. Facilita el desarrollo paralelo por diferentes equipos.
2. Permite hacer que partes específicas de la aplicación manejen varios usuarios según las necesidades.
3. Permite usar diferentes tecnologías y lenguajes de programación.
4. Facilita la actualización y el mantenimiento de partes específicas de la aplicación al estar divididos en diferentes servicios.

Contras de la Arquitectura de Microservicios:

1. Requiere una gestión más compleja para el despliegue.
2. Introduce la necesidad de gestionar la comunicación y posibles fallos en la red.
3. En otros casos, puede ser más difícil de mantener, ya que cada servicio tiene su propia base de datos.

A continuación presentaremos nuestro proyecto de manera más amplia en la siguiente sección.

Presentación

Nuestro sistema de tienda de colchas y frazadas online, nombrado como Colchinilla.com es un sistema que está diseñado para ofrecer a nuestros clientes una experiencia de compra fluida y eficiente. Este sistema ha sido desarrollado utilizando una arquitectura de microservicios, lo que nos permite gestionar y escalar cada componente de forma independiente para asegurar un rendimiento óptimo y una alta disponibilidad.

La implementación del sistema se llevó a cabo con tecnologías de vanguardia. Empleamos Docker para contener y desplegar nuestros servicios de manera consistente y segura, garantizando que cada parte de la aplicación opere en un entorno controlado. Utilizamos Node.js como nuestro entorno de ejecución para desarrollar una aplicación rápida y escalable, y Express como el framework que facilita la creación de una API robusta y eficiente para manejar las solicitudes de nuestros usuarios.

Con esta plataforma, buscamos proporcionar una experiencia de compra en línea que sea tan cómoda y acogedora como nuestras colchas. La combinación de tecnologías y un enfoque centrado en el usuario nos permite ofrecer un servicio de alta calidad que satisface las necesidades y expectativas de nuestros clientes.

Sobre el sistema

El sistema de tienda de Colchinilla.com es un sistema construído con la anteriormente explicada arquitectura de microservicios, en dicho sistema creamos

un sistema de una tienda para mostrar unos pequeños registros de unas colchas que se tenían anteriormente, el sistema se basa en tener cuatro principales servicios: pagos, órdenes, clientes e inventario. En este sistema la API Gateway es la tienda principal dónde se puede acceder a dichos servicios.
Para detallar mejor como es el sistema, lo definiremos por partes.

Estructura del Sistema

En este sistema se tienen los cuatro servicios que se han mencionado anteriormente.

- **Pagos:** En este microservicio se puede realizar un pago desde la tienda para hacer una compra de una colcha o frazada, este pago se hace después de que el cliente haga una orden.
- **Órdenes:** La orden lo que permite es comprar sobre pedido una colcha por un cliente, esta demuestra el total y permite proceder a realizar el pago.
- **Clientes:** Este servicio funciona para poder almacenar los nombres de clientes y sus cuentas para poder verlas si un usuario administrador quisiera para administrarlos.
- **Inventario:** El inventario contiene todo el catálogo de productos que se venden en la tienda para poder ser vistos por un administrador de inventarios y así tener en cuenta que productos si están disponibles y cuáles no.

También hay que hacer mención de la **tienda** en línea, que es dónde se conectan todos estos servicios para ser operados por el usuario.
Cabe aclarar que en este sistema no se usó ninguna API; la razón principal de no usar ninguna API fue que los servicios simplemente se comunican por red para este trabajo, si fuéramos a crear una página en línea se podría tomar en cuenta el uso de APIs. Teniendo en cuenta todos estos elementos podemos proceder a explicarlos visualmente.

Diagramas del Sistema

El diagrama principal de la Arquitectura del sistema es este:

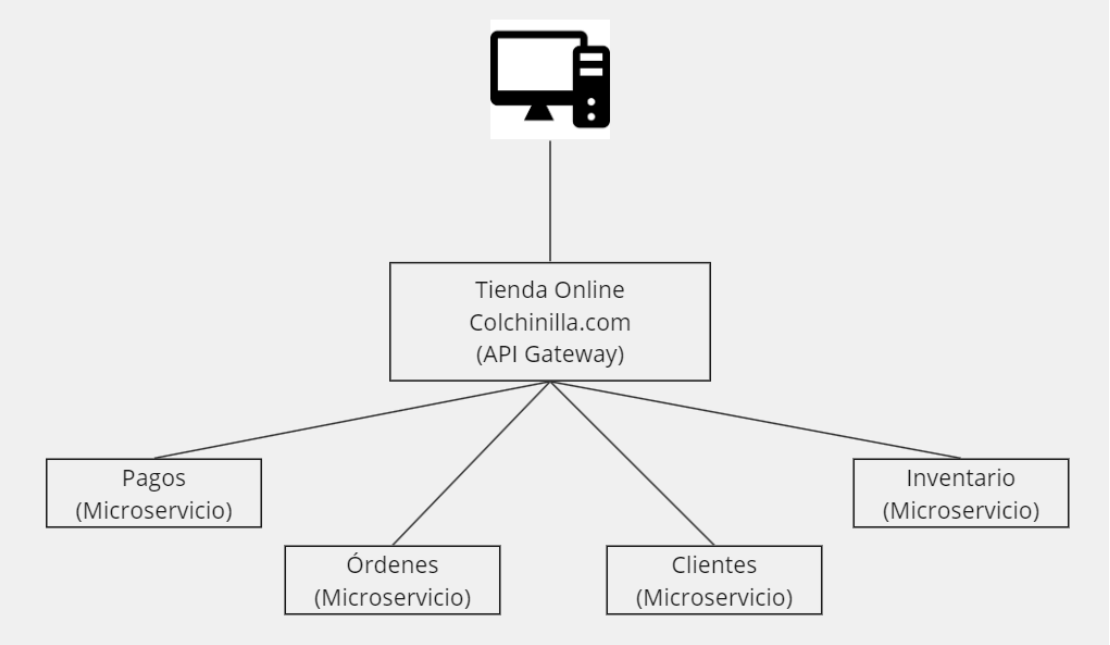

Figura 2. *Diagrama que muestra cómo está estructurado el sistema del equipo.*

Como se puede observar, en el diagrama de la figura 2 se muestra cómo es que el sistema está implementado siguiendo la estructura de arquitectura de microservicios, teniendo desde la computadora la comunicación principal para el usuario para conectarlo a nuestro sistema. Procedemos con el uso de una API Gateway dónde se puede ver que se conectan todos los servicios, aunque en este caso no es una puerta de enlace de APIs debido a que no utilizamos APIs. Volviendo al diagrama, se tienen los microservicios que explicamos en la parte de la estructura del sistema, dónde se es posible acceder a cada uno mediante la página principal de la tienda online.
Es de esta manera que está construído nuestro sistema y sus componentes que se tienen para funcionar para poder contener los microservicios se utilizó Docker conteniendo los microservicios y la tienda al mismo tiempo.

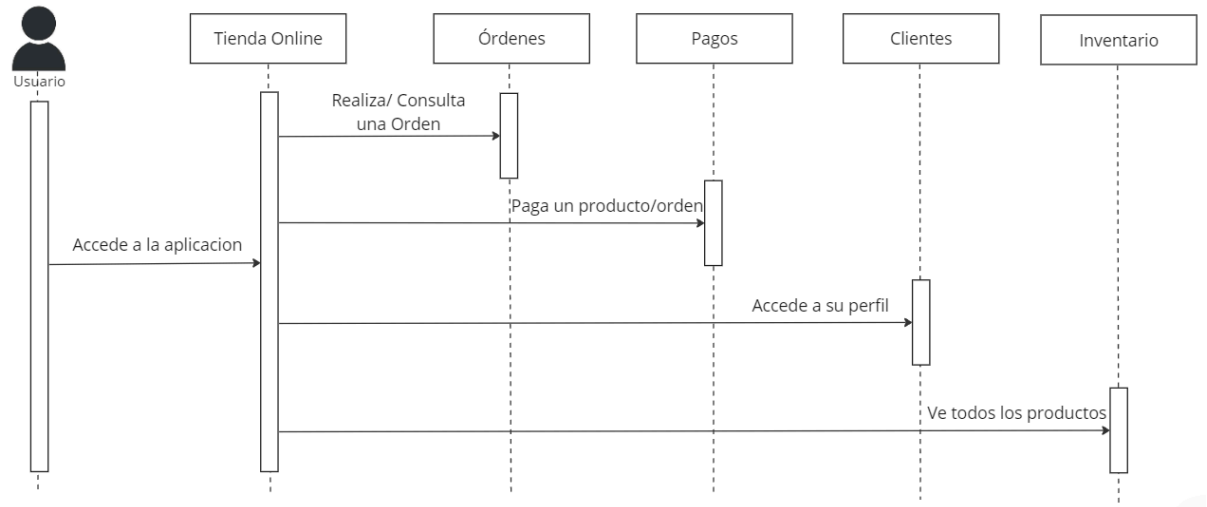

Figura 3. *Diagrama que muestra el intercambio de mensajes y secuencias del sistema.*

En este diagrama se explica a grandes rasgos el cómo se intercambia la información a través de los diferentes componentes del sistema mediante el uso de la aplicación. Cómo se puede ver primeramente el usuario accede a la aplicación de la tienda online de Colchinilla.com, es a partir de aquí que se puede conectar al usuario con los microservicios que le proporcionamos, entonces es posible decir que a través de la tienda online puede *realizar o consultar una orden*, puede *pagar un simple producto o una orden*, también se puede *consultar la información de la cuenta mediante la opción de "Mi Perfil"* y por último, *se pueden ver todos los productos del servicio de inventario*.

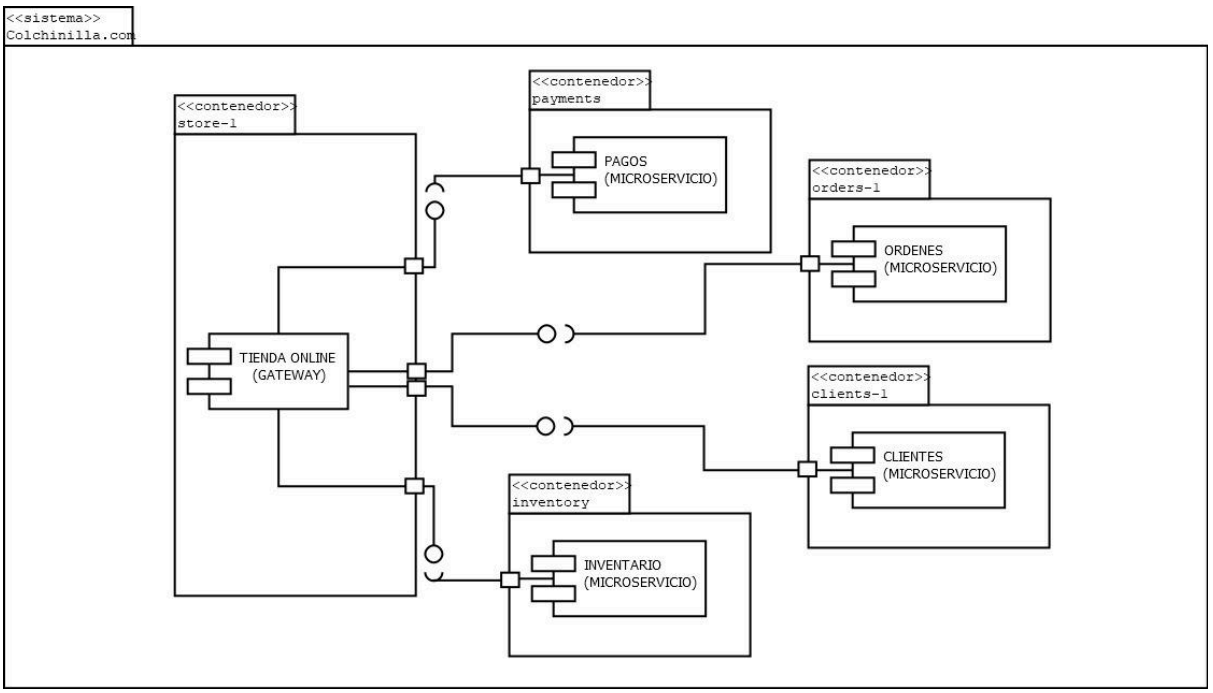

Figura 4. *Diagrama de componentes del sistema.*

Andrés Contreras Sánchez
Gerardo Rivas Delgado

En este diagrama se muestran las partes del sistema general, en sí todo el sistema está contenido en el sistema de “Colchinilla.com” y de ahí podemos ver los componentes de los contenedores todos los servicios son proporcionados por la interfaz del contenedor de “store-1” para luego ser mandados por los puertos explicados en la sección de “Funcionamiento del sistema” cada microservicio requiere de la página de tienda para ser accedida a ellos.
También se puede ver que se tiene en cada uno de los contenedores un microservicio, como se tiene explicado en la parte siguiente de “Tecnologías Usadas”.

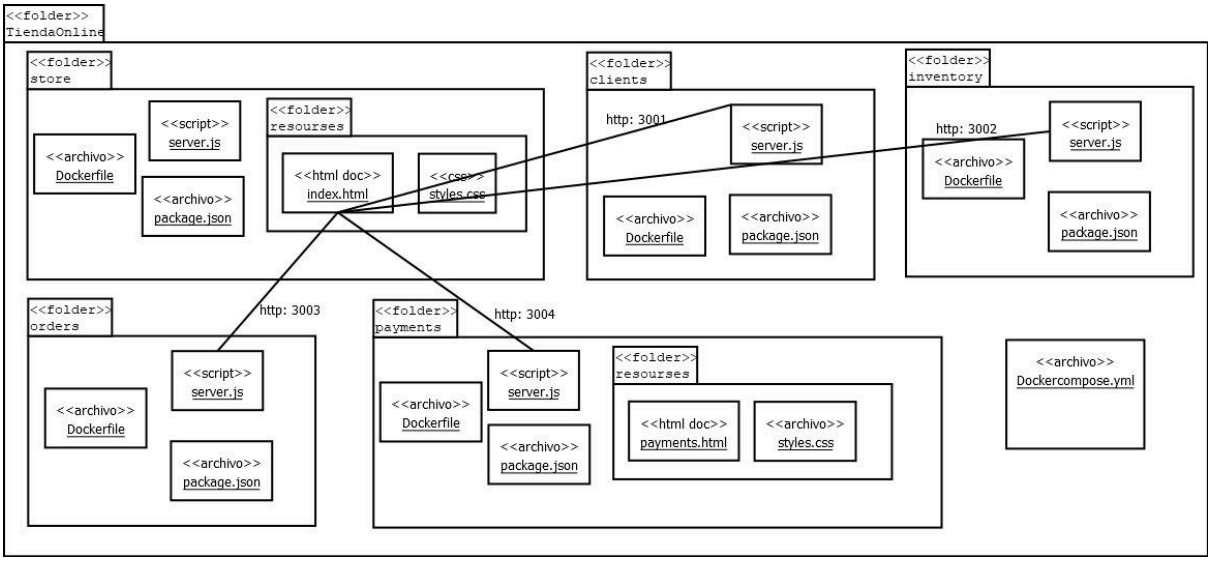

Figura 5. *Diagrama de deployment.*

En este último diagrama se especifican los artefactos físicos producidos por el proceso de desarrollo del sistema, teniendo en cuenta todos los componentes que hemos explicado anteriormente.

Tecnologías Usadas

Se utilizaron las tecnologías de Docker para contener, Express para conexiones y Node.js para definir la lógica.

- **Docker:** Docker principalmente es el sistema que tenemos para contener los microservicios de la página web que desarrollamos, en estos contenedores se alojan los servicios de pagos (*payments*), órdenes (*orders-1*), clientes (*clients-1*) e inventario (*inventory*) la lógica de cada uno de los microservicios referentes. Y así asignar también el contenedor de la tienda para ser mostrado (*store-1*). [Véase Figura 6]
- **Express:** Express es utilizado para poder manejar archivos estáticos como el HTML, CSS e imágenes, además con él se pueden configurar las rutas de los diferentes microservicios, así como procesar solicitudes y enviar respuestas HTTP.
- **Node.js:** Para definir la lógica de cada microservicio se utilizó Node.js para poder programar los microservicios y lo que se tiene que hacer en cada

microservicio, así mismo como para poder programar las vistas principales de esta página.

-

También podemos decir que dicho sistema está alojado en el puerto 3000 de Localhost y sus contenedores en sus puertos consiguientes.

Figura 6. *Contenedores de microservicios en Docker.*

Funcionamiento del Sistema

Al entrar al sistema se puede ver la pantalla principal con los productos más vendidos y una opción para comprarlos, se puede también observar en la parte superior derecha los botones de “Mis órdenes” para acceder al microservicio de las órdenes. La parte de “Mi perfil” mostrando la información del usuario y por último la parte de “Ver todos los productos” para mostrar el catálogo de productos vendidos en general.

La tienda en línea Colchinilla.com está compuesta por varios microservicios, cada uno implementado en Node.js y framework Express. El proyecto utiliza Docker y Docker Compose para administrar y desplegar los diferentes contenedores.

Contenedores de Colchinilla.com:
Clientes (puerto 3001): Gestiona la información de los usuarios.
Inventario (puerto 3002): Maneja los productos en inventario.
Órdenes (puerto 3003): Administra las órdenes realizadas.
Pagos (puerto 3004): Procesa los pagos realizados.
Tienda (puerto 3000): Interfaz principal que integra todos los microservicios.
Docker Compose se utiliza para construir y ejecutar todos los servicios de la tienda.
Cada servicio tiene su propio Dockerfile para configurar el entorno de ejecución.

Funcionamiento General:
Docker Compose construye y levanta todos los servicios de la tienda colchinilla.com
Los usuarios interactúan con la interfaz principal (tienda) que coordina con los servicios de clientes, inventario, órdenes y pagos para proporcionar la funcionalidad completa de la tienda.

Cada microservicio maneja su propia lógica de negocio y datos.
Esta estructura permite que cada componente funcione de manera independiente, lo cual facilita el constante mantenimiento, la disponibilidad y el poder tener una escalabilidad de la tienda en línea como y donde sea necesario.

Capturas de Pantalla del Sistema

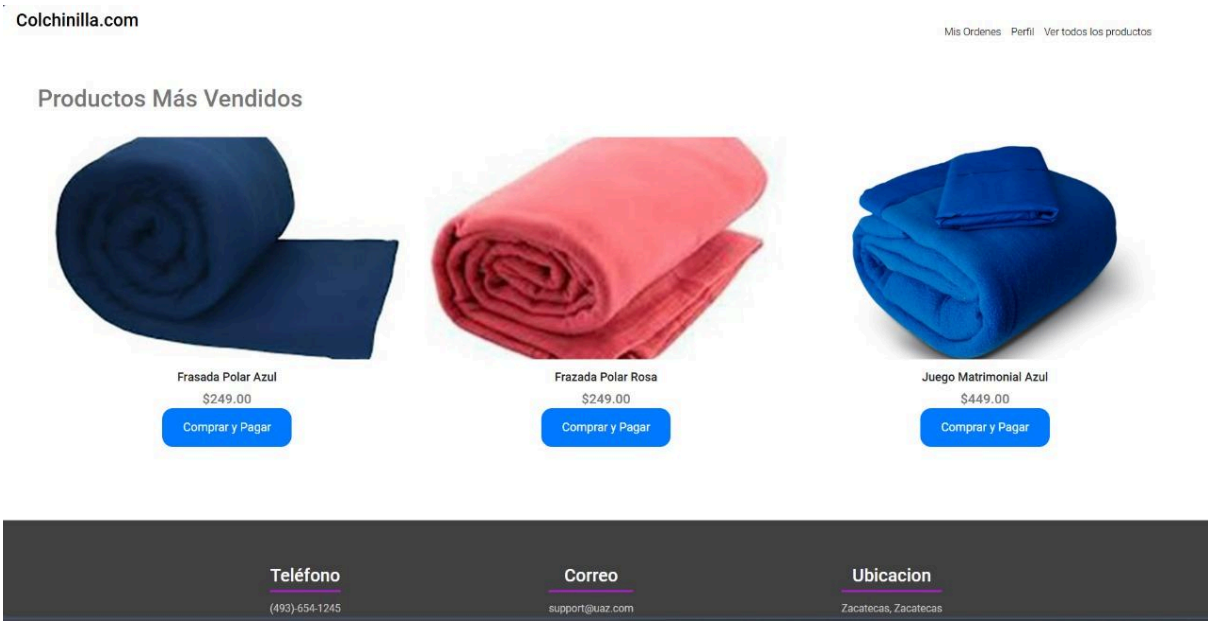

Figura 7. *Página principal de la tienda de Colchinilla.com*

Figura 8. *Página de la sección del listado de pedidos accedida desde “Mis Órdenes”.*

Figura 9. *Página de la sección del listado de clientes accedida desde “Mi Perfil”.*

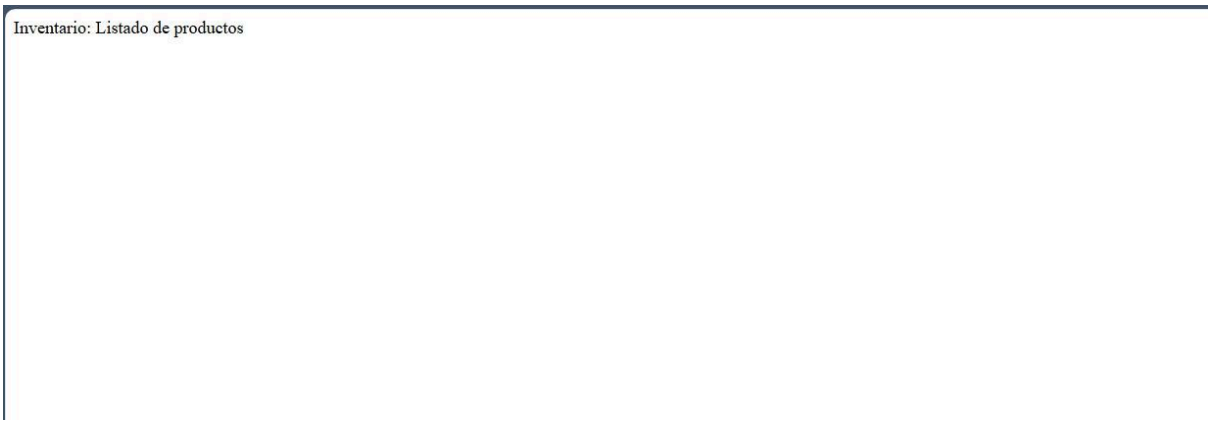

Figura 10. *Página de la sección del inventario accedida desde “Ver todos los productos”.*

Figura ?. *Pantalla de servicios no disponibles.*

Figura ?. *Pantalla de los pagos realizados desde el microservicio de “payments”.*