TRABAJO DE GRADO
Opción Seminario-Diplomado.

Desarrollo de un Marketplace de Productos Artesanales mediante una Arquitectura de Microservicios en Python

Corporación Universitaria Remington.
Facultad de Ingeniería
Ingeniería en Sistemas

Andrés Esteban Vásquez Peña
Yesid Velásquez Giraldo
Tutor: Diego Fernando Marín Lozano

Opción de Trabajo de grado Seminario-Diplomado
2025

Dedicatoria

Dedicatoria de Andrés Vásquez

Dedico este trabajo a mi familia, por su apoyo incondicional, por la paciencia en los momentos difíciles y por confiar en mí incluso cuando las circunstancias no eran fáciles. A mis padres, que con su ejemplo y esfuerzo me enseñaron el valor del estudio y del trabajo honesto. También dedico este proyecto a quienes, con una palabra de ánimo o un buen consejo, me motivaron a continuar y a no rendirme.

Dedicatoria de Yesid Velásquez

Este trabajo se lo dedico a Dios, por darme la fortaleza y la salud necesarias para culminar esta etapa. A mi familia, que ha sido mi principal soporte emocional y mi inspiración para seguir adelante. A mis amigos y compañeros, por acompañarme en el camino, por las horas de estudio compartidas y por demostrarme que el esfuerzo en equipo siempre rinde frutos.

Agradecimientos

Agradecimientos de Andrés Vásquez

Agradezco el apoyo a mi familia, a mi compañero de trabajo y a la universidad Uniremington por su apoyo constante y en la creación de este proyecto.

Agradecimientos de Yesid Velásquez

Quiero agradecer a la institución y a los docentes que hicieron posible este proceso de aprendizaje, por compartir su experiencia y acompañar cada etapa del proyecto. A mi familia, por su confianza y respaldo en todo momento, y a mi compañero de trabajo de grado, por el compromiso, la responsabilidad y el esfuerzo compartido. Este resultado es fruto del trabajo en equipo y del apoyo de todas estas personas.

Tabla de Contenido

Resumen

Los artesanos están enfrentando diversas barreras técnicas y económica que no les permite acceder fácilmente al mercado llevando a recurrir intermediarios donde salen perjudicados por la falta de transparencia en los pagos. Proponemos un Marketplace especializado en la venta de productos para hacer más visible la cultura artesanal y un contacto más con los clientes, este Marketplace está basado en una arquitectura de microservicios desarrollado con las tecnologías modernas como Python, FastAPI y contenedores con Docker .

Para la construcción de este proyecto se aplicaron buenas prácticas de ingeniería de sistemas donde se incluye la documentación automática con Mkdoc y pruebas automáticas con Pytest, el sistema está compuesto por microservicios independientes tales como autenticación, productos, pedidos y pagos.

El resultado de este proyecto sugiere la facilidad de las transacciones directas, justas y transparentes entre artesanos y consumidores, reduciendo considerablemente la dependencia de intermediarios permitiendo la realización de pedidos personalizados buscando garantizar más la rentabilidad de los productores.

Palabras clave

Microservicios, FastAPI, Marketplace, Python, Pruebas Automatizadas.

Pregunta orientadora de la búsqueda

¿De qué manera puede un prototipo de Marketplace especializado en artesanías, basado en una arquitectura de microservicios y en buenas prácticas de ingeniería de software en Python, ayudar a reducir las barreras comerciales que enfrentan los artesanos locales y fortalecer una economía justa y transparente?

Sustentación teórica de la pregunta

Uno de los problemas más graves que enfrentan los artesanos es el exceso de intermediación en la comercialización de sus productos (Artesanías de Colombia, 2024; Baracaldo, n.d.), lo que dificulta la generación de ingresos justos para ellos. Manquillo Astaiza (2019) sostiene que los canales de comercialización actuales afectan la actividad de los artesanos porque la cadena que conecta el producto con el consumidor final se fragmenta, lo que implica una pérdida de control sobre el producto del cual ellos son artífices.

Además, es necesario reconocer que el proceso de comercialización incluye varias etapas o subetapas desde el productor hasta el consumidor final, lo que genera un incremento en los precios que no se ve reflejado en los ingresos de los artesanos (Cortés Martínez, 2023). Estudios realizados en otros contextos latinoamericanos también evidencian problemáticas similares de intermediación y de acceso a mercados (Abarca & Hunneus, 2024; El Mercado Nacional de Artesanías, 2018).

Problema

El sector artesanal enfrenta múltiples dificultades para acceder a plataformas de comercio digital que reconozcan y respeten las particularidades culturales y técnicas de sus productos (Artesanías de Colombia, 2024). Como consecuencia, los artesanos tienen una visibilidad limitada, lo que los obliga a depender de más intermediarios, afectando sus ingresos y la transparencia del proceso de compra y venta (Cabrera Cevallos, 2025).

En este contexto, las plataformas digitales diseñadas con arquitecturas de microservicios representan una opción eficiente, escalable y mantenible para atender las necesidades específicas del sector artesanal (López & Liriano García, 2019). Además, la implementación de buenas prácticas en el desarrollo de software, que incluye metodologías ágiles y tecnologías modernas como Python y contenedores Docker, incrementa la calidad, seguridad y mantenibilidad del sistema (FastAPI, 2025.; Vásquez Ramírez, 2025.).

Todo esto permite que la creación de un marketplace especializado contribuya a reducir las barreras comerciales, promoviendo una economía local más justa y transparente (Artesanías de Colombia, 2024; Cabrera Cevallos, 2025).

Metodología

La búsqueda de información se orientó mediante las siguientes palabras clave: microservicios, FastAPI, marketplace, Python, Docker, Pytest y buenas prácticas de ingeniería de software.

Se emplearon principalmente buscadores y recursos especializados para priorizar documentos de primer nivel, entre ellos Google Scholar, Google Books y Google Search, que permitieron localizar literatura académica, documentación oficial y material de apoyo técnico.

Las estrategias de búsquedas implementadas se organizaron en torno a nuestra pregunta orientadora, se agrupo la información en tres bloques principales, 1. sustentación del problema, se identificaron las barreras comerciales y técnicas que enfrenta el sector artesanal, 2. fundamentos tecnológicos y arquitectónicos, se identificaron patrones de diseño del proyecto en termino a la estabilidad y mantenibilidad a futuro, 3 estándares de calidad de calidad, se revisaron métodos y herramientas de buenas prácticas para potencializar la calidad de este proyecto para que tenga una calidad profesional (Scrumban/XP, n.d.; Urtiaga, 2020; Okken, 2022).

Desarrollo

La implementación de un Marketplace representa una estrategia clave para disminuir las barreras comerciales, al ofrecer un canal de distribución alternativo y una forma de comercialización más directa (Artesanías de Colombia, 2024; Cabrera Cevallos, 2025). Un marketplace especializado en artesanías permite conectar de manera más cercana a los productores con los consumidores, reduciendo la dependencia de intermediarios y promoviendo precios más justos.

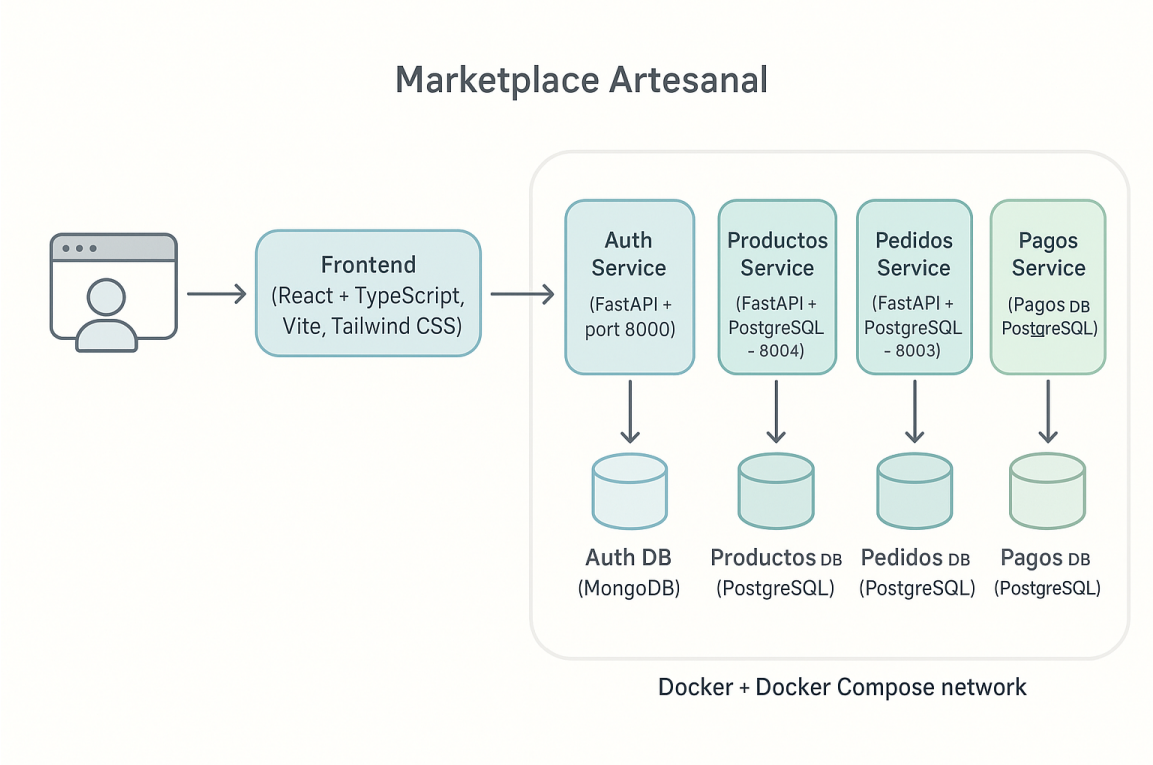
Este tipo de Marketplace busca crear un espacio donde artesanos puedan tener interacción más cercana con sus clientes, ofreciendo sus productos, explicado su valor culturar y ofreciendo precios más transparentes (Cabrera Cevallos, 2025).

Arquitectura de Microservicios

La arquitectura de microservicio se basa en la creación de servicios pequeños que trabajan de forma independiente, cada uno tiene una responsabilidad bien definida y se comunica a través de interfaces bien establecidas. Este enfoque nos permite facilitar el mantenimiento, escalabilidad y la posibilidad de realizar despliegues de manera independiente (López & Liriano García, 2019).

En este proyecto se definirá varios microservicios como la autenticación de usuarios, la gestión del catálogo de productos, el manejo de pedidos y el procesamiento de pago, esta separación nos permitirá mejorar la organización del código y la facilidad de incorporar nuevas funcionalidades o mejoras en el futuro.

Figura 1: arquitectura del proyecto



Buenas prácticas

La integración con el lenguaje Python y el framework FastAPI nos permite tener un alto rendimiento orientado a entornos de producción, esto nos facilita el desarrollo de

las funcionalidades del Marketplace minimizando la duplicación de código. FastAPI se basa más que todo en anotaciones entandares de Python ayudando a soportar los editores de código y detectando errores tempranos.

La documentación que incorpora FastAPI se genera de manera automática mediante OpenAPI, nos ofrece una interfaz web interactiva que nos ayudará a validad el funcionamiento de los servicios, por su parte, Docker nos permitirá desplegar de manera automática nuestro Marketplace completo. (Urtiaga, 2020). Finalmente, Pytest es una herramienta fundamental para la ejecución de pruebas automatizadas, contribuyendo a la estabilidad y calidad del software (Okken, 2022; Vásquez Ramírez, 2025).

Tabla 1: Arquitectura general del sistema

Módulo / Microservicio	Descripción funcional	Tecnología utilizada
Autenticación	Manejo de registro y login de usuarios	FastAPI, JWT, PostgreSQL
Catálogo	Administración y visualización de productos artesanales	FastAPI, SQLite
Pedidos	Gestión del flujo de compra y seguimiento de pedidos	FastAPI, Celery
Pagos	Procesamiento seguro de transacciones	FastAPI, Stripe API
Gateway / API Gateway	Enrutamiento de peticiones entre microservicios	Nginx Proxy, Docker

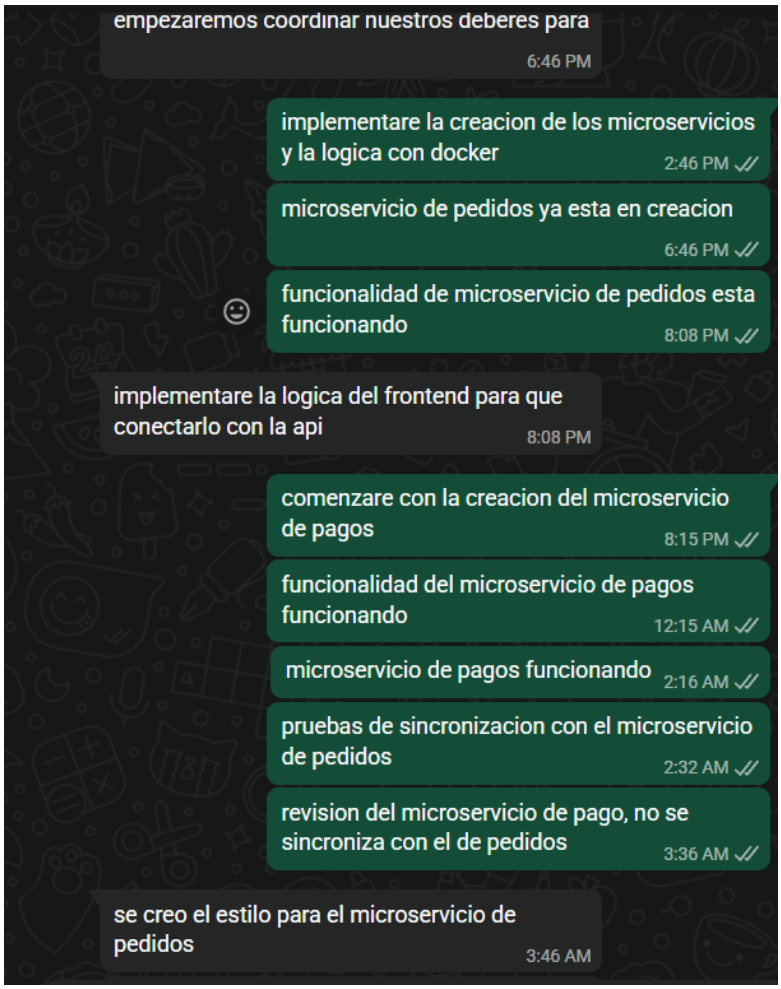
Tabla 2: Arquitectura general del sistema

Fase	Actividad	Duración estimada	Entregable
Análisis	Identificación de requisitos funcionales	Semana 1	Documento de requisitos
Diseño	Modelado de arquitectura y base de datos	Semana 2	Diagramas UML y ER
Implementación	Desarrollo de microservicios	Semanas 3-6	Código fuente en repositorio Git
Pruebas	Pruebas unitarias y de integración	Semana 7	Reporte de Pytest
Despliegue & Documentación	Contenerización con Docker, MkDocs	Semana 8	Manual técnico y despliegue local

Implementación

El proyecto se realizó siguiendo la metodología implementada Scrumban donde se definió los roles encargados para la creación de los microservicios y el frontend, la herramienta utilizada para la comunicación fue la aplicación de mensajería instantánea WhatsApp para coordinar y llevar el log de lo que se ha hecho y lo que quedo faltando (ver figura 1).

Figura 2: evidencia de coordinación y log de errores o pendientes

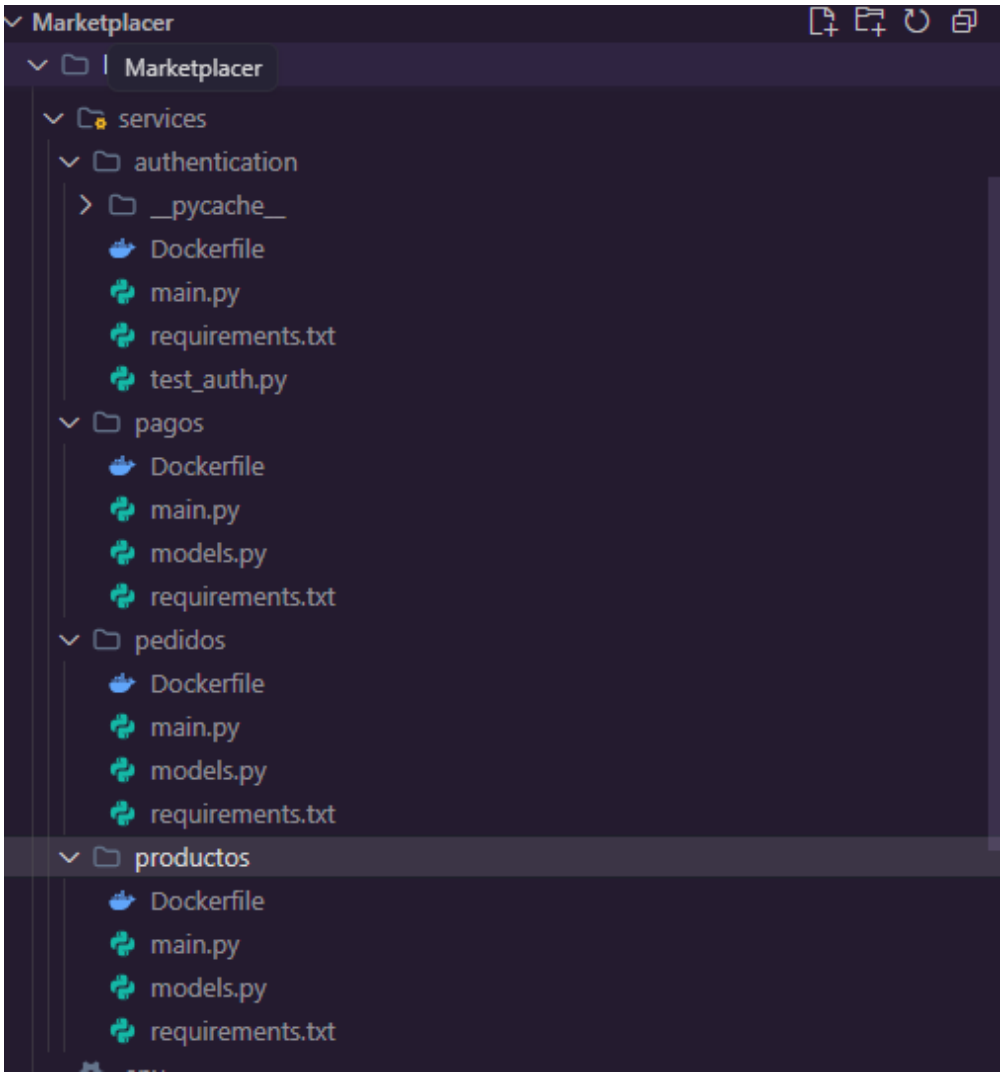


Fuente: Andrés peña, Yesid Velásquez

Construcción del Backend

Para la creación de los microservicio, se pensó en hacerlos de manera separadas para mejor mantenimiento y escalabilidad a futuro, con una estructura de carpetas mas organizadas como se ve en la siguiente imagen. (ver figura 3)

Figura 3: Organización de Microservicios



Fuente: Andrés Peña, Yesid Velásquez

Creación de los endpoint con FastAPI

Se implemento los CRUD (crear, leer, actualizar y eliminar) a todos los microservicios para tener una funcionalidad más robusta, como se muestra en el siguiente fragmento de código.

```python
@router.get("/", response_model=list[ProductoResponse])
async def get_productos(db: Session = Depends(get_db)):
    try:
        productos = db.query(Producto).all()
        return productos
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=f"Error al obtener productos: {str(e)}"
        )
```

Fuente: Andrés Peña, Yesid Velásquez

El fragmento anterior se puede ver el método Get que nos permitirá leer todo los productos que tengamos (en el fragmento anterior son los endpoint de productos pero son los mismos para pedidos y pagos), también se visualiza la ruta donde podemos visualizar todo nuestro catálogo de producto en formato JSON y por ultimo tenemos nuestro código de estados donde responderá 200 que hace referencia a que todo está bien y mostrara todos los productos que tengamos disponibles o 500 si no hay conexión con el servidor

```python
@router.post("/", response_model=ProductoResponse)
async def create_producto(producto: ProductoCreate,
             db: Session = Depends(get_db)):
    new_producto = Producto(**producto.dict())
    db.add(new_producto)
    db.commit()
    db.refresh(new_producto)
    return new_producto
```

Fuente: Andrés Peña, Yesid Velásquez

El fragmento anterior se puede ver el método Post, que nos permite crear nuevos productos (lo mismo aplica para pedidos o pagos lo que cambia es la ruta) los códigos de estado que tienen que salir al crear un producto son 201 hace referencia a producto creado exitosamente o 500 error del servidor.

```
@router.get("/{id}", response_model=ProductoResponse)
async def get_producto(id: int, db: Session = Depends(get_db)):
    db_producto = db.query(Producto).filter(Producto.id == id).first()
    if not db_producto:
        raise HTTPException(status_code=404, detail="Producto no
encontrado")
    return db_producto
```

Fuente: Andrés Peña, Yesid Velásquez

En el anterior fragmento de código se puede ver el método Get por id, nos permite visualizar solamente solo los detalles de 1 solo producto (lo mismo esta para pedidos y pagos) el código de estado son 200: éxito devuelve el producto, 404: producto no encontrado y 500: error del servidor

```
@router.put("/{id}", response_model=ProductoResponse)
async def update_producto(
       id: int, producto: ProductoUpdate,
       db: Session = Depends(get_db)):
    db_producto = db.query(Producto).filter(Producto.id == id).first()
    if not db_producto:
        raise HTTPException(status_code=404,
            detail="Producto no encontrado")
    for key, value in producto.dict(exclude_unset=True).items():
        setattr(db_producto, key, value)
    db.commit()
    db.refresh(db_producto)
    return db_producto
```

Fuente: Andrés Peña, Yesid Velásquez

En el anterior fragmento de código se puede ver el método Put por id, nos permite cambiar solamente solo los detalles de 1 solo producto (lo mismo esta para pedidos y pagos) el código de estado son 200: producto actualizado exitosamente, 404: producto no encontrado y 500: error del servidor

```
@router.delete("/{id}")
async def delete_producto(id: int, db: Session = Depends(get_db)):
    db_producto = db.query(Producto).filter(Producto.id == id).first()
    if not db_producto:
        raise HTTPException(status_code=404,
             detail="Producto no encontrado")
    db.delete(db_producto)
    db.commit()
    return {"message": "Producto eliminado"}
```

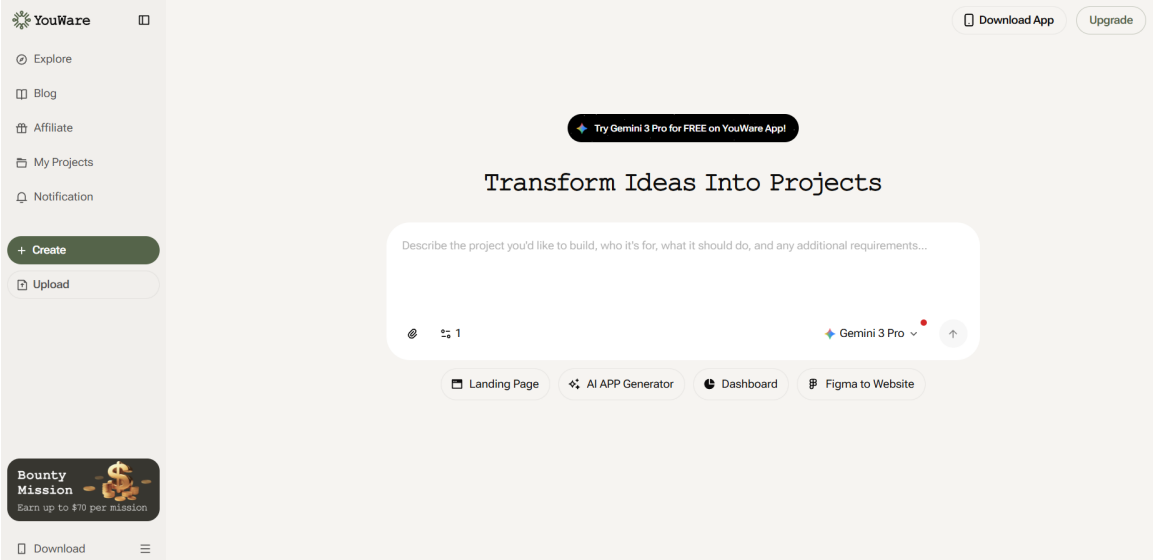
Fuente: Andrés Peña, Yesid Velásquez

En el anterior fragmento de código se puede ver el método Delete por id, nos permite borrar solamente solo 1 solo producto (lo mismo esta para pedidos y pagos) el código de estado son 200: producto eliminado exitosamente, 404: producto no encontrado y 500: error del servido

Construcción del Frontend

Para la creación del frontend se utilizó una herramienta de inteligencia artificial llamada YouWare que nos permitió agilizar la creación del estilo del frontend como se muestra en la siguiente imagen. (ver figura 4)

Figura 4: página principal de la herramienta YouWare



Fuente: Andrés Peña, Yesid Velásquez

El resultado final se verá en las siguientes imágenes a continuación, se verá la página de inicio del Marketplace (ver figura 5)

Figura 5: página principal del Marketplace

En la imagen anterior se puede ver la página principal del Marketplace, se puede ver en la siguiente url [http://localhost:5173/.](http://localhost:5173/)

Figura 6: página de autenticación

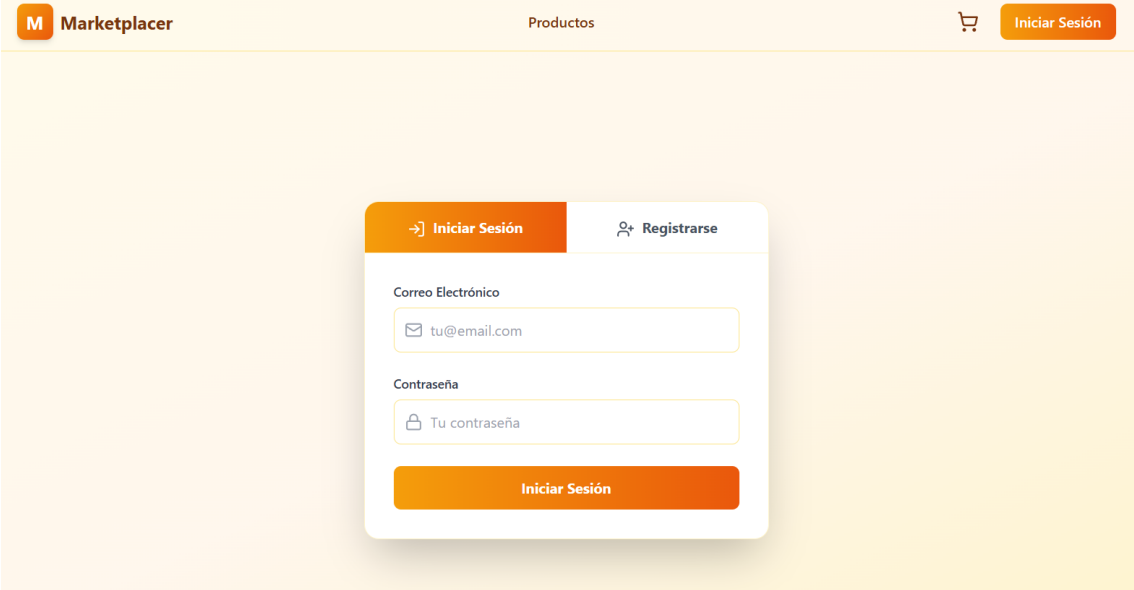
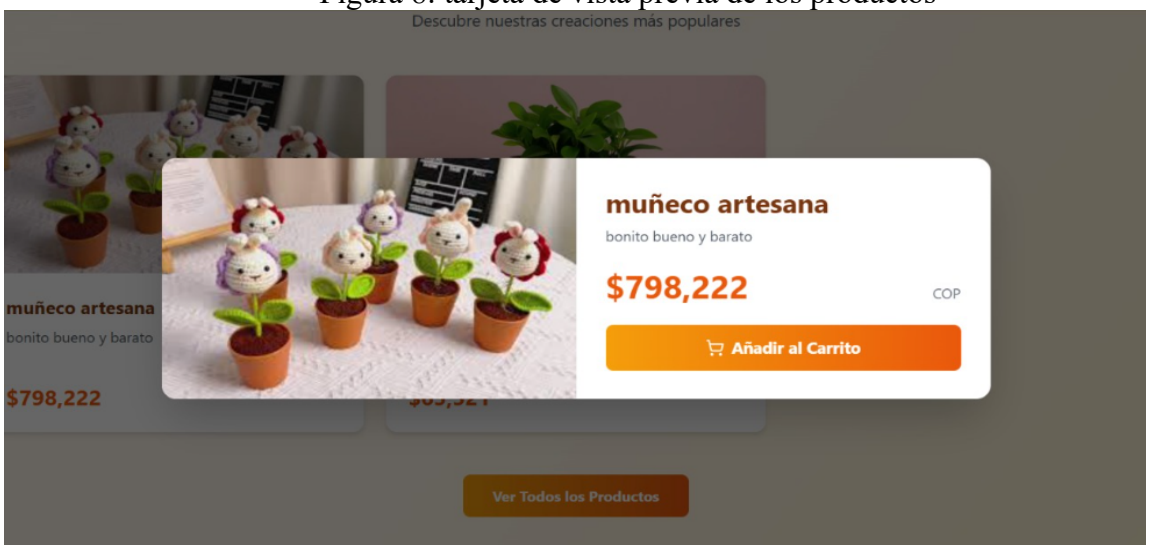


Figura 7: página de productos destacados

Figura 8: tarjeta de vista previa de los productos

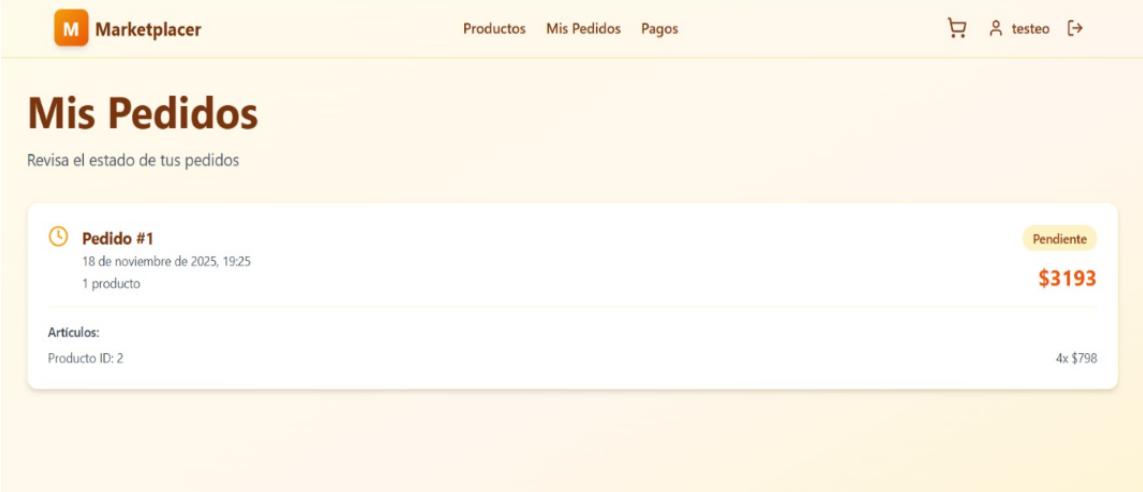


Vista de tarjetas para los productos con animación por frame-motion y estilos con tailwindcss

Figura 9: página de carritos de compra

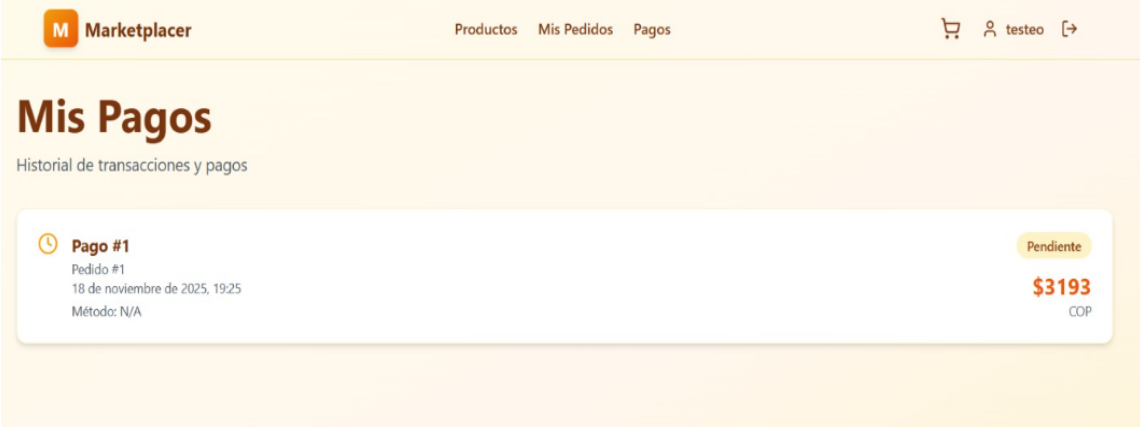
Frontend para el microservicio de pedidos:

Figura 10: página de pedidos



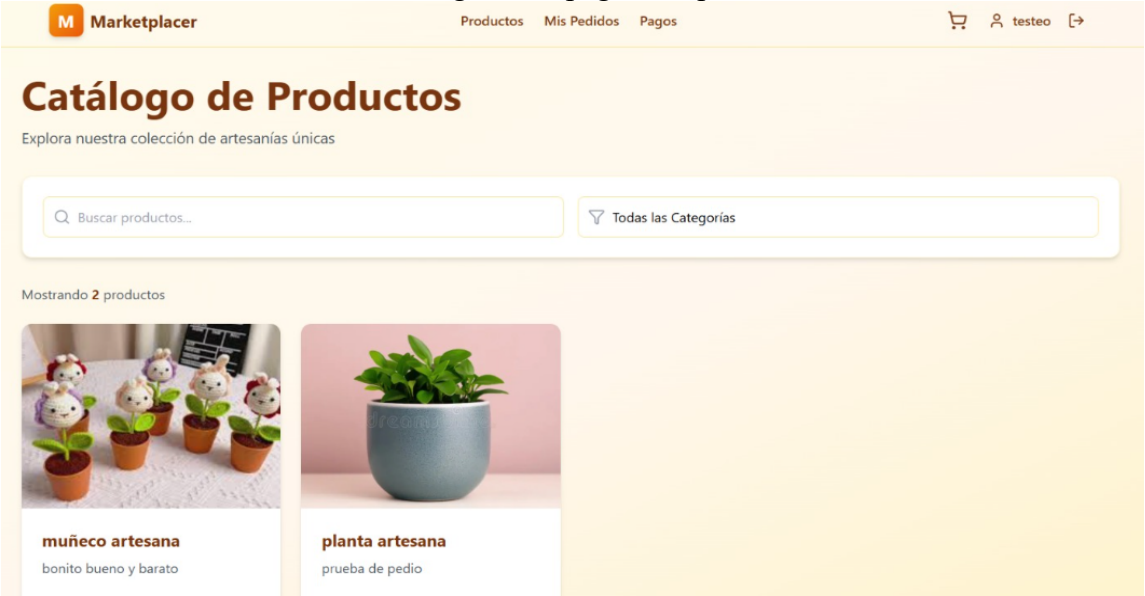
Se visualiza el apartado de los pedidos con su información

Figura 11: página de pagos



A continuación, se mostrará el resultado del frontend para los microservicios. (ver figura 11)

Figura 12: página de productos



Se visualiza el catálogo que los usuarios tienen para escoger

Despliegue de la Solución

Figura 13: despliegue de los microservicios

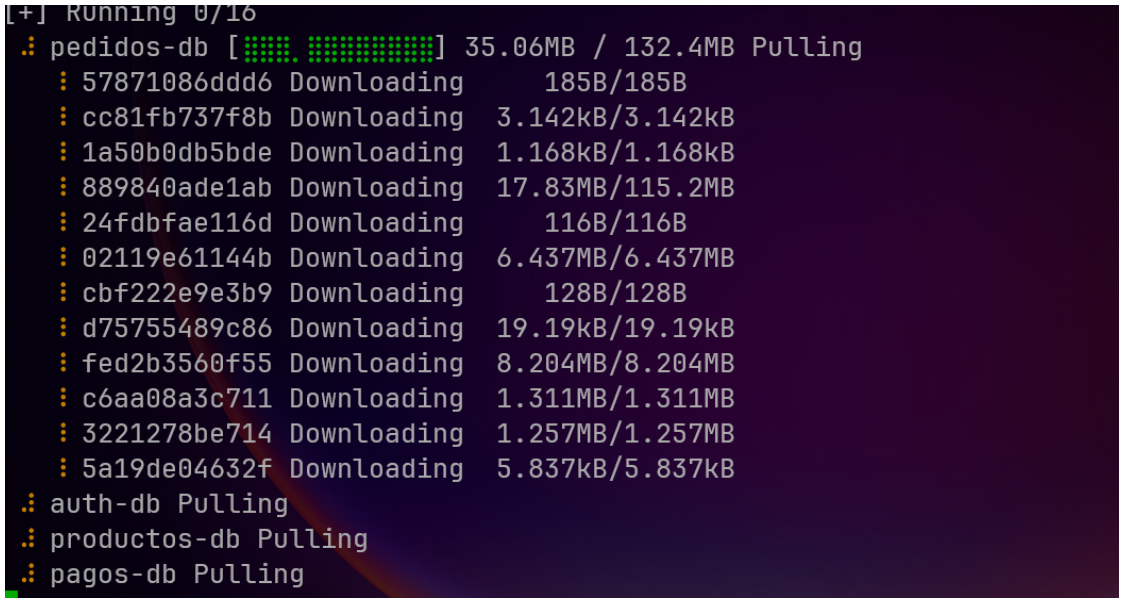


Figura 14: documentacion automatica de FastAPI

Figura 15: documentación automática de con Mkdocs

Se pudo llevar a cabo un gran a avance a pesar de las dificultades que se ha tenido en la creación del proyecto y el límite de entrega, siguiendo las indicaciones de la metodología scrumban, se realizó la coordinación y log, el seguimiento de roles, la construcción, la documentación, el testo y el resultado final.

Conclusiones

La revisión teórica y el desarrollo del prototipo confirman que un marketplace especializado puede ayudar a los artesanos locales a superar diversas dificultades asociadas a las altas comisiones y a las barreras técnicas de entrada que imponen muchas plataformas de comercio electrónico (Artesanías de Colombia, 2024; Cortés Martínez, 2023). Estas condiciones limitan la visibilidad de sus productos y afectan la estabilidad de sus ingresos.

La propuesta de un marketplace basado en una arquitectura de microservicios contribuye a reducir la dependencia de intermediarios y a facilitar una relación más directa entre artesanos y consumidores, promoviendo una comercialización más justa, transparente y sostenible.

El uso de tecnologías modernas como FastAPI, Docker y Pytest impulsa a que este tipo de proyectos alcancen altos niveles de calidad, mantenibilidad, seguridad y confiabilidad. La separación en microservicios permite que cada componente funcione de manera independiente, lo que facilita el despliegue, el mantenimiento y la incorporación de mejoras futuras en el sistema..

Referencias

Abarca, P., & Hunneus, T. (2024). Estudio de casos: Modelos actuales de comercialización de las artesanías en Chile. https://www.indap.gob.cl/sites/default/files/2024-04/Estudio-de-casos-Modelos-actuales-de-comercializacion-de-las-artesanias-en-Chile.pdf

Artesanías de Colombia. (2024). Informe de gestión. https://artesaniasdecolombia.com.co/Documentos/Contenido/48815_info_gestion_2024_v9_18022025_rdo_jv_compressed_(1).pdf

Baracaldo, P. E. (n.d.). Problemas y objetivos del sector artesano. https://artesaniasdecolombia.com.co/Documentos/Contenido/1792_plan_estrategico_sector_artesano.pdf

Cabrera Cevallos, M. R. (2025). Intermediación comercial y su impacto en la agricultura familiar campesina. https://repositorio.uasb.edu.ec/bitstream/10644/10402/1/T4521-MDS-Cabrera-Intermediacion.pdf

Cortés Martínez, J. A. (2023). Impacto de exportaciones artesanales y TLC sobre el PIB. https://sired.udenar.edu.co/16329/1/2023013.pdf

El Mercado Nacional de Artesanías. (2018). El mercado nacional de artesanías. https://tesiunamdocumentos.dgb.unam.mx/ptd2018/marzo/0772210/0772210.pdf

FastAPI. (n.d.). Documentación oficial de FastAPI. https://fastapi.tiangolo.com/es/

Manquillo Astaiza, N. A. (2019). La vulnerabilidad del sector artesanal colombiano en el ámbito nacional e internacional como “negocio de conocimiento tradicional”. Revista La Propiedad Inmaterial. https://revistas.uexternado.edu.co/index.php/propin/article/view/6346/8405

Okken, B. (2022). Python testing with pytest: Simple, rapid, effective, and scalable. The Pragmatic Programmers LLC.

Scrumban/XP. (n.d.). Scrumban/XP: Propuesta para mejorar la eficiencia de la gestión de proyectos ágiles en el desarrollo de software. ProQuest. https://www.proquest.com/openview/629e1e39dbb4ddec7e004ead26cb0808/1?pq-origsite=gscholar&cbl=1006393

Urtiaga, G. G. (2020). Docker para novatos: Aprende a administrar esta tecnología en tiempo récord. AprendeIT.

Vásquez Ramírez, A. F. (n.d.). Optimización de la infraestructura on-premise de la Facultad de Ingeniería para escalabilidad y la entrega continua del software con prácticas DevOps. https://bibliotecadigital.udea.edu.co/server/api/core/bitstreams/08dec97f-0fa9-45c2-9679-61bcdc6c7990/content

Yoris, A. C. (n.d.). Primeros pasos con FastAPI. Andrés Cruz.