Workshop Architecture: MyBookStore

Tópicos Especiales en Telemática (2025-2)

Group Number: S2566-0671

Nombre Completo

Juan Manuel Young Hoyos
Yoban Stéban Nova Aranda

Tutor: Juan Carlos Montoya Mendoza

Medellín, 30 de julio de 2025

Índice

1 Arquitectura del Proyecto **2**
1.1 Introducción 2
1.2 Modelo C4 2
1.3 Diagrama de Casos de Uso 4
1.4 Vistas de Implementación 4

1 | Arquitectura del Proyecto

1.1 | Introducción

El proyecto **MyBookStore** se concibió inicialmente con una arquitectura monolítica pero evolucionó a un modelo de microservicios para mejorar la escalabilidad, la resiliencia y la velocidad de despliegue continuo. Para describir formalmente la solución empleamos el *C4 Model*[1] y un conjunto de diagramas UML que facilitan la comunicación técnica con todos los stakeholders.

1.2 | Modelo C4

1.2.1 | Nivel 1 - Contexto

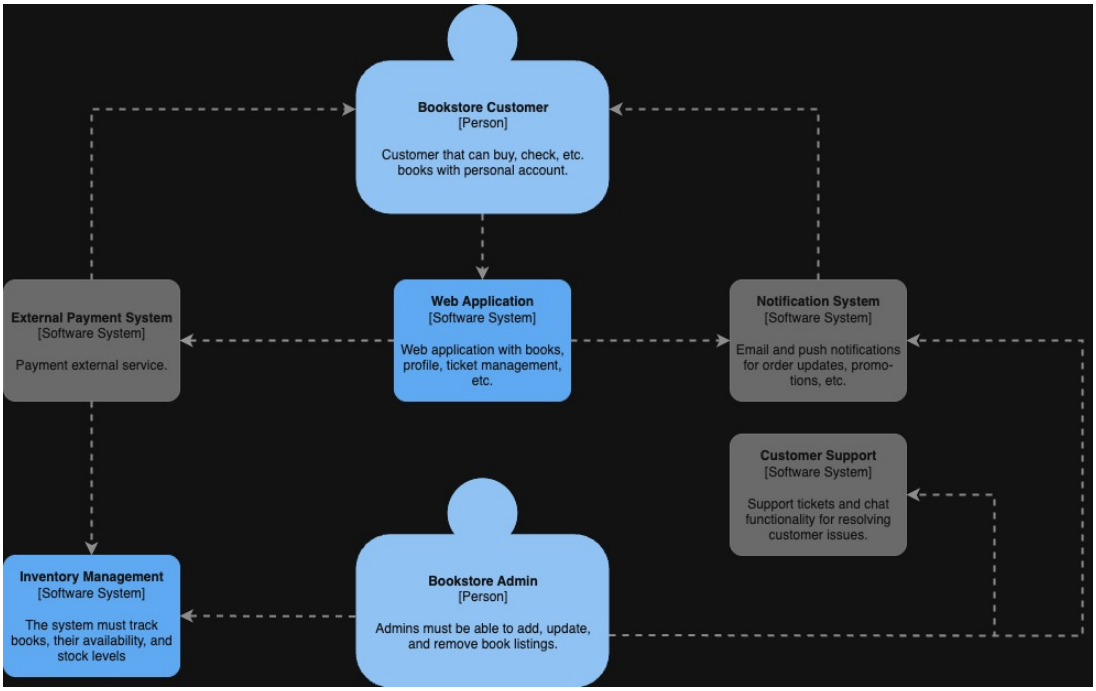

Figura 1.1: Diagrama C4 - Nivel Contexto.

La figura 1.1 muestra a los principales actores (clientes, administradores y sistemas externos como el *Payment Gateway*) y cómo interactúan de forma global con **MyBookStore**. Este nivel responde a la pregunta *"¿quién usa el sistema y para qué?"*.

[1] Desarrollado por Simon Brown, C4 permite visualizar software en cuatro niveles: Contexto, Contenedores, Componentes y Código.

1.2.2 | Nivel 2 - Contenedores

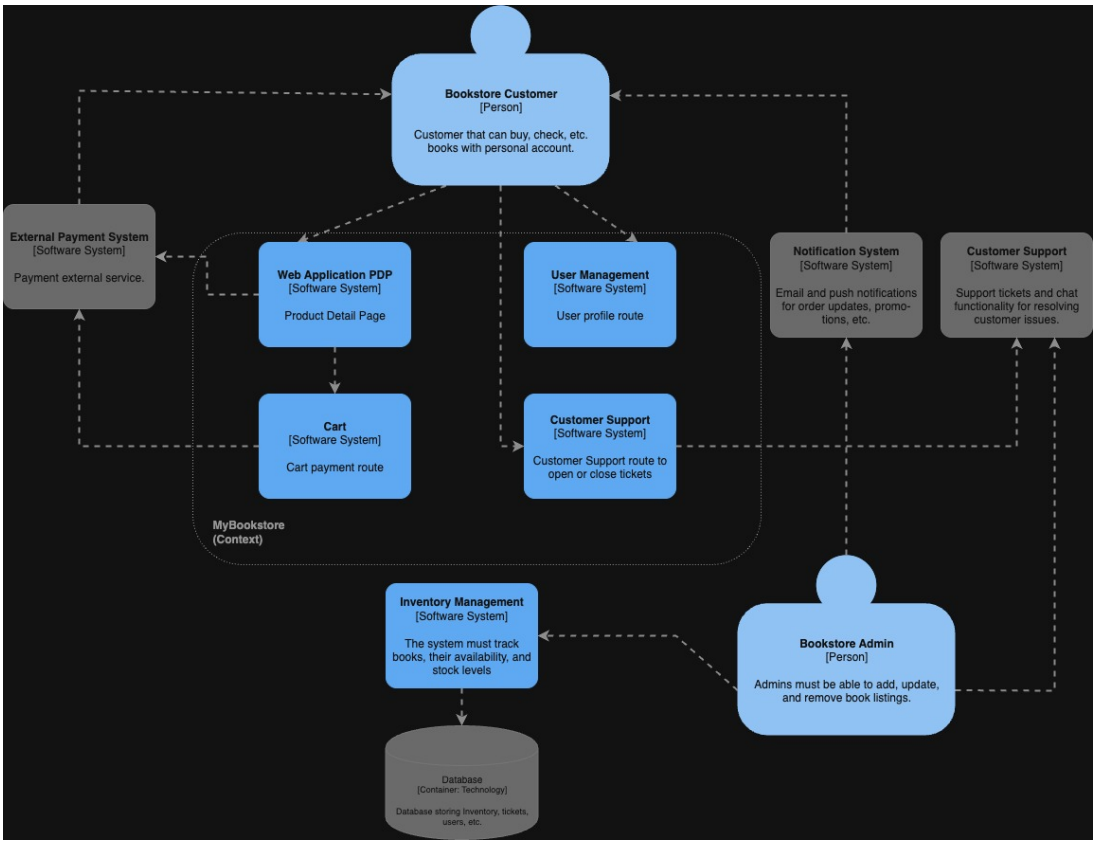

Figura 1.2: Diagrama C4 - Nivel Contenedores.

El diagrama de la figura 1.2 descompone el sistema en contenedores desplegables (aplicación Web, Carrito, Gestión de Inventario, Soporte, etc.). Cada contenedor encapsula una responsabilidad de negocio clara y expone interfaces (REST o eventos) que favorecen el bajo acoplamiento.

1.3 | Diagrama de Casos de Uso

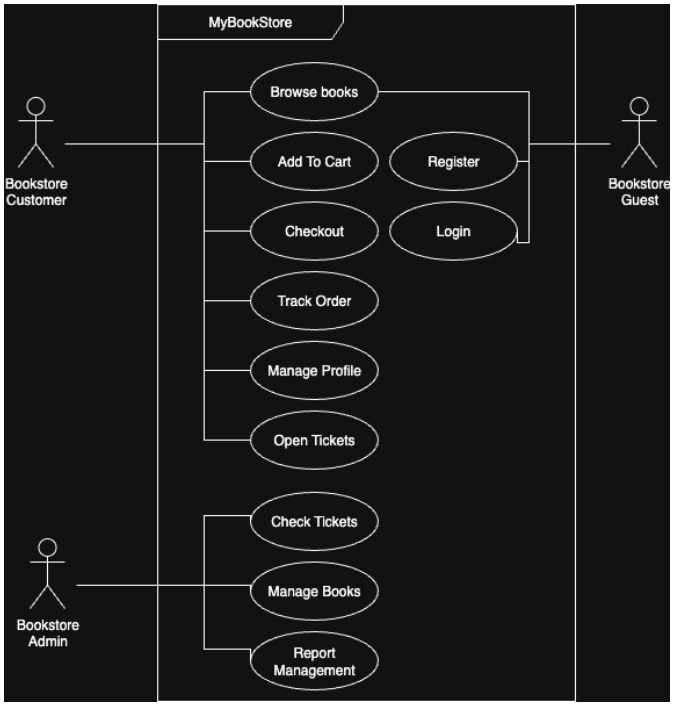

Figura 1.3: Casos de uso principales de **MyBookStore**.

El diagrama de la figura 1.3 resume las funcionalidades nucleares que guían el diseño de contenedores y componentes: desde la navegación y el checkout hasta la gestión de inventario y la atención al cliente.

1.4 | Vistas de Implementación

1.4.1 | Arquitectura Monolítica Inicial

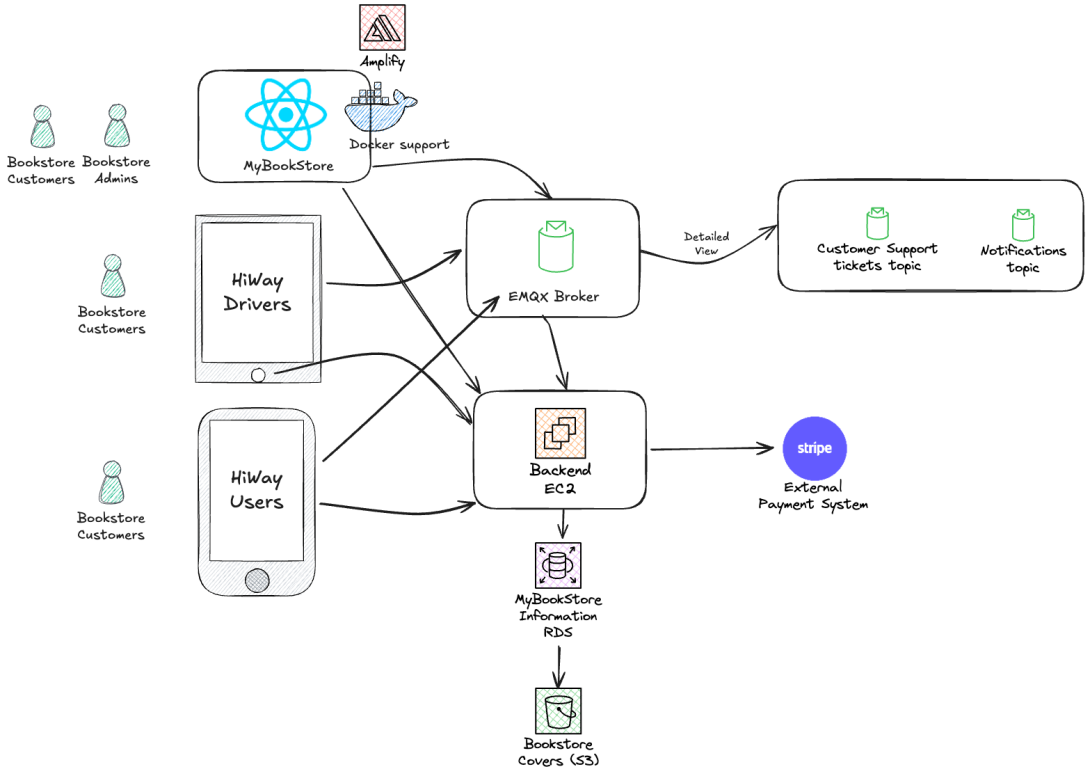

Figura 1.4: Vista de la arquitectura monolítica desplegada en una instancia EC2.

1.4.2 | Arquitectura Basada en Microservicios

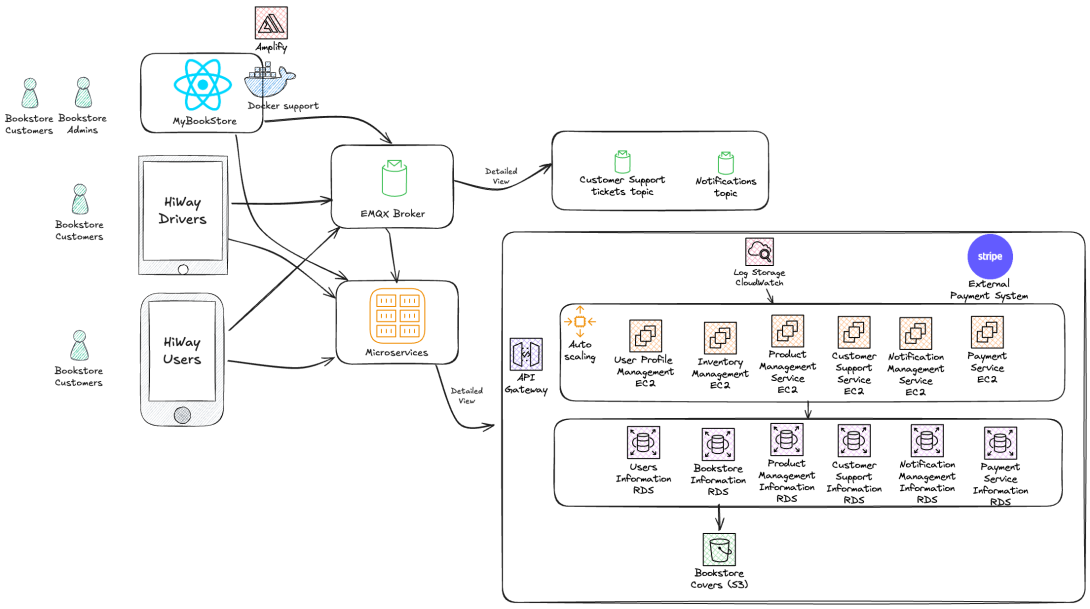

Figura 1.5: Vista de la arquitectura de microservicios con *API Gateway*, autoescalado y persistencia distribuida.

1.4.3 | Comparativa Monolito vs Microservicios

Dimensión	Monolito (Fig. 1.4)	Microservicios (Fig. 1.5)
Simplicidad	Una base de código y un único artefacto facilitan el desarrollo inicial y la depuración.	Mayor complejidad operativa: múltiples repositorios, canalizaciones de CI/CD y orquestación (p. ej. Kubernetes, ECS).
Despliegue	Cambio atómico: todo el sistema se libera como un bloque.	Despliegue independiente por servicio, permitiendo *canary releases* y rollback selectivo.
Escalabilidad	Escalado vertical (más CPU/RAM) o duplicando el monolito completo.	Escalado horizontal fino (autoescalado de los servicios críticos, p. ej. Inventario o Checkout).
Tiempo de arranque / pruebas	Arranque y testeo integrados pero pueden volverse lentos conforme crece el código.	Servicios pequeños arrancan rápido; las pruebas de integración exigen herramientas de *contract testing*.
Evolución del equipo	Óptimo para equipos pequeños (< 5–8 desarrolladores).	Permite escuadras por dominio de negocio y libertad tecnológica (polyglot).
Resiliencia	Fallo de un módulo puede comprometer todo el sistema.	Falla aislada al servicio afectado; uso de *circuit breakers* y reintentos.
Costo Operativo	Menor al inicio (menos infraestructura y monitorización).	Mayor inversión en observabilidad, redes internas y seguridad de servicio a servicio.

Cuadro 1.1: Ventajas, retos y trade-offs entre monolito y microservicios.

Conclusión. El monolito es idóneo para *MVPs* y equipos reducidos; sin embargo, la demanda prevista de **MyBookStore** (tráfico variable, picos de comercio electrónico y necesidad de despliegue continuo) justificó la transición a microservicios, respaldada por un API Gateway, autoescalado y observabilidad centralizada.