Arquitectura del sistema – Proyecto Medusa JS con Gráficas para Administrador

UNIVERSIDAD CENTROAMERICANA "JOSÉ SIMEÓN CAÑAS"

Facultad de Ingeniería y Arquitectura

Aplicaciones de código abierto - Ciclo 01/2025

Grupo 06

Integrantes:

Orellana Vividor, Gerardo Andre	00053520	Product Owner
Christian Alejandro Paz Escobar	00132720	Scrum Master
Rodriguez Rodriguez, David Neftali	00218621	Tech Lead
Alas Moscoso, Noe Bladimir	00262020	Developer
Zacatales López, César Adonay	00223021	Developer

Antiguo Cuscatlán, lunes 26 de mayo de 2025

MEDUSAJS V 2.8.2

Índice

1. Introducción

Este documento describe la arquitectura técnica del sistema personalizado basado en MedusaJS, una plataforma de comercio electrónico headless, extendida para incluir un panel administrativo con gráficas analíticas.

La solución propuesta consiste en la creación de un dashboard administrativo para Medusa.js que se conectará al backend mediante APIs RESTful. El dashboard permitirá visualizar métricas de productos, órdenes y clientes de forma eficiente. La solución incluirá la creación de nuevas rutas API para gestionar funcionalidades específicas, como la visualización en tiempo real de las órdenes. Esto permitirá una experiencia dinámica y reactiva para los administradores del sistema.

2. Objetivo del sistema

La implementación de un dashboard administrativo avanzado en MedusaJS tiene un impacto potencial significativo en la eficiencia operativa y la toma de decisiones dentro del entorno de comercio electrónico. Al centralizar la visualización de métricas clave y alertas en una interfaz clara e intuitiva, se estima una reducción en el tiempo necesario para acceder y analizar datos críticos del negocio.

El sistema está compuesto por tres capas principales:

- Backend (API REST – MedusaJS)
- Frontend Administrador (React con dashboard personalizado)
- Base de datos (PostgreSQL + opcional Redis)

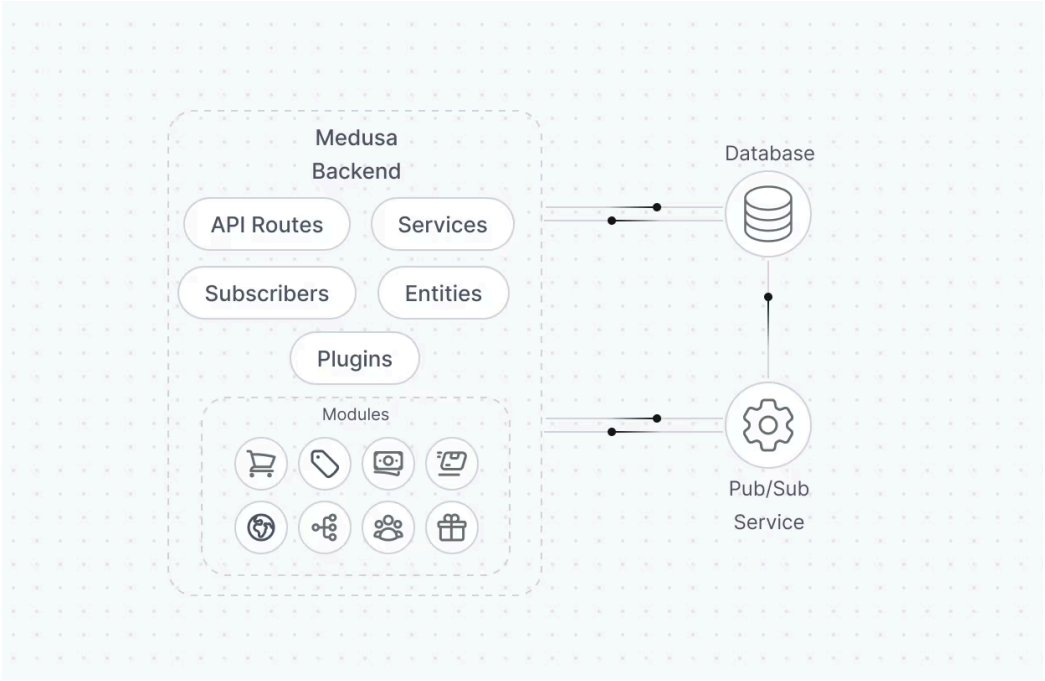
Estas capas se comunican mediante HTTP y consultas SQL, con un enfoque desacoplado (headless) que permite flexibilidad de desarrollo y escalabilidad.

3. Arquitectura general

El sistema está compuesto por tres capas principales:

- Backend (API REST – MedusaJS)
- Frontend Administrador (React con dashboard personalizado)
- Base de datos (PostgreSQL + opcional Redis)

Estas capas se comunican mediante HTTP y consultas SQL, con un enfoque desacoplado (headless) que permite flexibilidad de desarrollo y escalabilidad.



Headless es una arquitectura que separa el frontend del backend.

- El frontend es lo que ve el usuario (interfaz gráfica).
- El backend maneja la lógica de negocio y la base de datos.

Esta separación permite que el backend le envíe la información al frontend a través de una conexión llamada API. Así, el frontend puede mostrar esos datos de distintas maneras, ya sea en una página web, una aplicación móvil o incluso en otros dispositivos. Esta forma de trabajar da más libertad para crear experiencias personalizadas, adaptables y fáciles de ampliar en el futuro.

Apoyándose de la arquitectura basada en eventos que es una forma de diseñar aplicaciones en la que los distintos componentes se comunican a través de eventos.

4. Componentes detallados

4.1 Backend API (MedusaJS)

- Basado en Node.js y Express.
- Gestiona operaciones de ecommerce: productos, pedidos, clientes.
- Expone API REST.
- Seguridad mediante JWT.
- Conectado a PostgreSQL.

4.2 Frontend Administrador

- Aplicación en ReactJS.
- Panel visual para administradores.
- Integración con Chart.js o Recharts para visualización de datos.
- Consume endpoints del backend.
- CORS habilitado en backend para permitir acceso seguro.

4.3 Base de Datos

- PostgreSQL: base de datos principal, contiene entidades como productos, órdenes, usuarios.

4.4 Seguridad

- Autenticación JWT para usuarios administradores.
- Control de acceso a rutas sensibles.
- Variables de entorno para protección de credenciales.
- Comunicación por HTTPS recomendada.

5. Tecnologías utilizadas

Componente	Tecnología
Backend API	Node.js, MedusaJS, Express
Frontend Admin	React, Tailwind, Chart.js
Base de Datos	PostgreSQL, Redis
Seguridad	JWT, Dotenv
Control de Versión	Git, GitHub
DevOps	Docker, PM2, Nginx
Analítica	Chart.js, Recharts

6. Escalabilidad y consideraciones finales

La arquitectura propuesta permite una operación robusta, segura y preparada para escalar. El panel de gráficas aporta un valor agregado clave para el monitoreo y la toma de decisiones basada en datos:

- El sistema está preparado para futuras integraciones con sistemas ERP, pasarelas de pago o herramientas externas de análisis.
- La documentación de APIs y endpoints está disponible para facilitar el trabajo de otros desarrolladores.
- Se recomienda mantener pruebas automatizadas y pipelines CI/CD para despliegues seguros.
- Separación de frontend y backend permite despliegue independiente.
- Compatible con Docker y entornos en la nube.
- El sistema puede adaptarse fácilmente a arquitecturas de microservicios.
- Es posible la integración con CDNs, balanceadores de carga y almacenamiento en la nube (AWS, GCP, Azure).