

Guía de Laboratorio #3

Laura Aviles, Nicolás Núñez

Universidad Manuela Beltrán

Ingeniería de Software

Bogotá, Colombia

2025

**Guiding Questions**

Which one of the following options is needed in the lifecycle of software architecture process?

1. Investigation

2. Analysis

3. prerequisites

4. Induction

Which of the following are correct statements?

1. An architecture may or may not defined components

2. An architecture is not dependable on requirements

3. An architecture is foremost an abstraction of a system that suppresses details of the components

4. that do not affect how they are used.

**A. Architecture Model**

**1. Brief Description of the ERP Functionalities:**

"DRIP HAUS" is an ERP designed to optimize and centralize the operations of clothing stores. Its main functionalities are:

* **Inventory Management:** Real-time stock control, allowing centralized management across multiple warehouses or stores.
* **Sales Management:** Efficient processing of sales, integrating various channels, including physical stores and online platforms.
* **Purchasing Management:** Automation of restocking and efficient supplier management.
* **Integrated CRM:** Creation of personalized marketing actions and improved communication with customers.
* **Billing and Accounting:** Versatile tools for invoicing, due date control, collections, and payments.

**2. Trade-offs:**

When designing the architecture of "DRIP HAUS," the following trade-offs were considered:

* **Scalability vs. Complexity:** Implementing a scalable architecture may increase system complexity, requiring more sophisticated management.
* **Functionality vs. Usability:** Adding multiple functionalities may affect the platform's simplicity and ease of use.
* **Cost vs. Performance:** Integrating advanced technologies may increase costs, even if it improves system performance.

**3. Selected Technologies:**

For the implementation of "DRIP HAUS," the following technologies were selected:

* **Database:**

PostgreSQL: A free, powerful, and open-source relational database, ideal for handling large volumes of structured data.

Alternatives: MySQL (also relational) or MongoDB Community Edition (a free NoSQL database for documents).

* **Infrastructure and Deployment:**

Railway: A free platform (with limits) for deploying backend applications and databases.

Free alternatives:

Render (free web services for small projects).

Fly.io (for deploying applications in the cloud for free).

Docker + Docker Compose (for local containers or deployment on free platforms).

* **Version Control:**

Git + GitHub: A free version control system. GitHub offers free private repositories for managing source code.

* **Authentication and Security:**

Auth.js (formerly NextAuth.js): A free solution for authentication in web applications.

Alternative: Keycloak (an open-source identity management system) if a more robust approach is required.

* **Backend:**

Ruby: Clear and concise syntax, ideal for speeding up development. It has a large community and many gems (libraries) available.

**A. Architecture Model**

**1. Breve descripción de las funcionalidades del ERP:**

"DRIP HAUS" es un ERP diseñado para optimizar y centralizar las operaciones de tiendas de ropa.  
Sus principales funcionalidades son:

* **Gestión de inventario:** Control en tiempo real del stock, permitiendo trabajar de forma centralizada con múltiples almacenes o tiendas.
* **Gestión de ventas:** Procesamiento eficiente de ventas, integrando diversos canales, incluyendo tiendas físicas y plataformas en línea.
* **Gestión de compras:** Automatización de reposiciones y gestión eficiente de proveedores.
* **CRM integrado:** Elaboración de acciones de marketing personalizadas y mejora de la comunicación con clientes.
* **Facturación y contabilidad:** Herramientas versátiles para la facturación, control de vencimientos, cobros y pagos.

**2. Trade-offs:**

Al diseñar la arquitectura de "DRIP HAUS", se consideraron los siguientes compromisos:

* **Escalabilidad vs. Complejidad:** Implementar una arquitectura escalable puede aumentar la complejidad del sistema, requiriendo una gestión más sofisticada.
* **Funcionalidad vs. Usabilidad:** Añadir múltiples funcionalidades puede afectar la simplicidad y facilidad de uso de la plataforma.
* **Costos vs. Rendimiento:** Integrar tecnologías avanzadas puede incrementar los costos, aunque mejore el rendimiento del sistema.

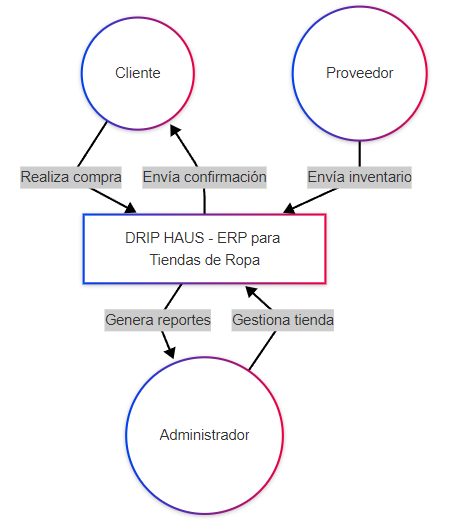
**3. Tecnologías seleccionadas:**

Para la implementación de "DRIP HAUS", se seleccionaron las siguientes tecnologías:

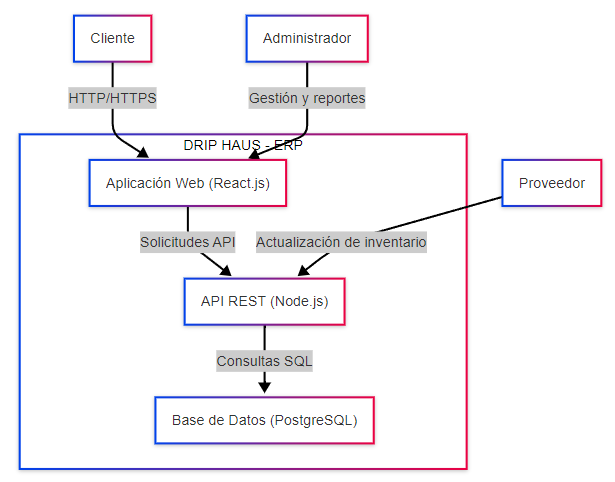
* **Base de datos:**  
  PostgreSQL: Base de datos relacional gratuita, potente y de código abierto, ideal para manejar grandes volúmenes de información estructurada.  
  Alternativa: MySQL (también relacional) o MongoDB Community Edition (base de datos NoSQL gratuita para documentos).
* **Infraestructura y Despliegue:**  
  Railway: Plataforma gratuita (con límites) para el despliegue de aplicaciones backend y bases de datos.  
  Alternativas gratuitas:  
  Render (servicios web gratuitos para proyectos pequeños).  
  Fly.io (para desplegar aplicaciones en la nube de forma gratuita).  
  Docker + Docker Compose (para contenedores locales o despliegue en plataformas gratuitas).
* **Control de versiones:**  
  Git + GitHub: Sistema de control de versiones gratuito. GitHub ofrece repositorios privados gratuitos para gestionar el código fuente.
* **Autenticación y Seguridad:**  
  Auth.js (anteriormente NextAuth.js): Solución gratuita para la autenticación en aplicaciones web.  
  Alternativa: Keycloak (sistema de gestión de identidad de código abierto) si se requiere un enfoque más robusto.
* **Backend:**  
  Ruby: Sintaxis clara y concisa, ideal para agilizar el desarrollo. Tiene una gran comunidad y muchas gemas (bibliotecas) disponibles.

**4. Diagramas Modelo C4**

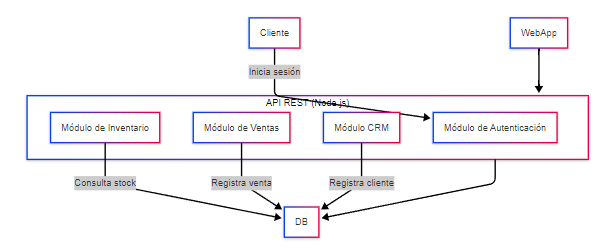
4.1 Diagrama de Contexto del Sistema

****

4.2 Diagrama de Contenedores

****

4.3 Diagrama de Componentes



4.4 Diagrama de Código



**Plan de Proyecto**

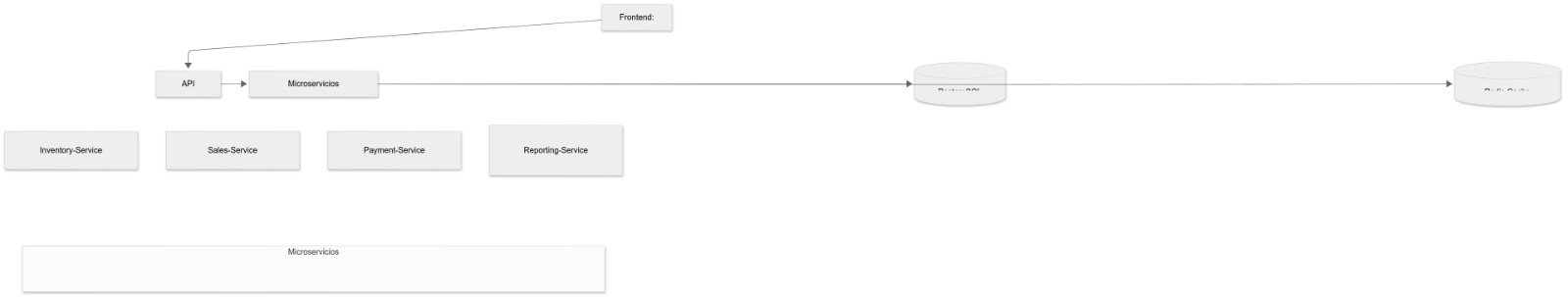
Versión 1.0 | Fecha: 03/02/2025

**1. Objetivos del Sistema**

Centralizar operaciones de inventario, ventas, pagos y nómina para tiendas de ropa.

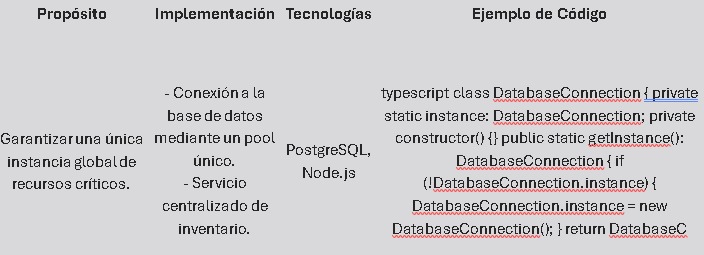
Cumplir normativas legales de facturación electrónica y seguridad de datos.

Escalar a 10,000 transacciones diarias con disponibilidad 24/7.

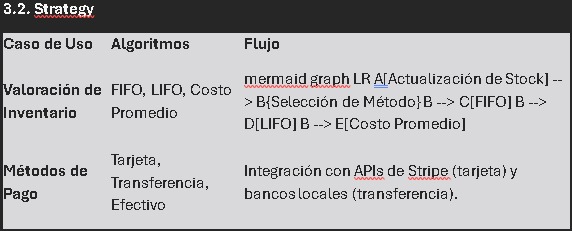
**2. Arquitectura General  
**

**3. Patrones de Diseño** **Detallados**

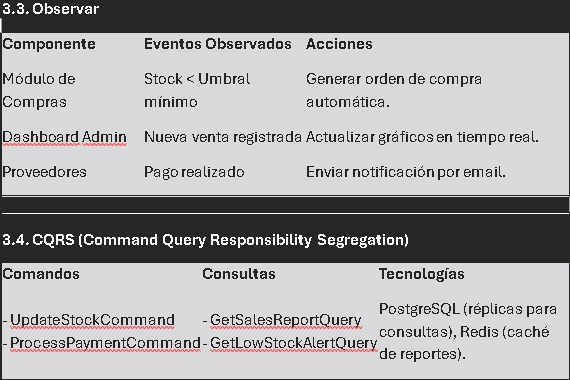
3.1 Singleton



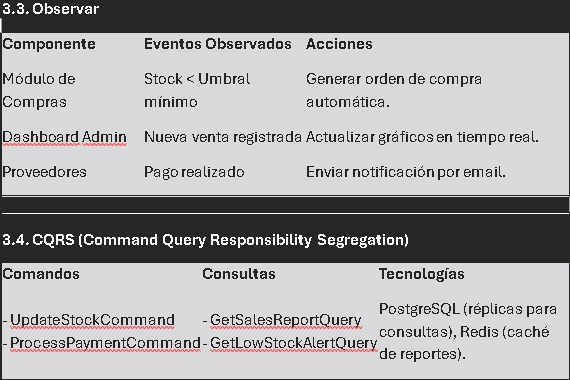
3.2 Strategy



3.3 Observar



3.4 CQRS (Command Query Responsibility Segregation)



**4. Implementación por Módulos**

4.1. Módulo de Inventario

Flujo de Actualización de Stock:

1. Recepción de mercancía → Registrar entrada (Strategy: Costo Promedio).

2. Venta → Reservar items (Observer notifica a dashboard).

3. Alerta stock bajo → Trigger para reposición automática.

4.2. Módulo de Ventas

Flujo de Checkout:

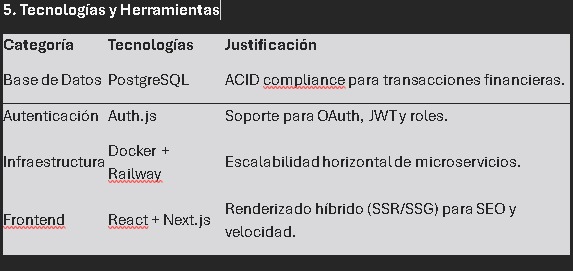
1. Cliente agrega productos al carrito (Factory: CarritoSession).

2. Validación de stock (Chain of Responsibility).

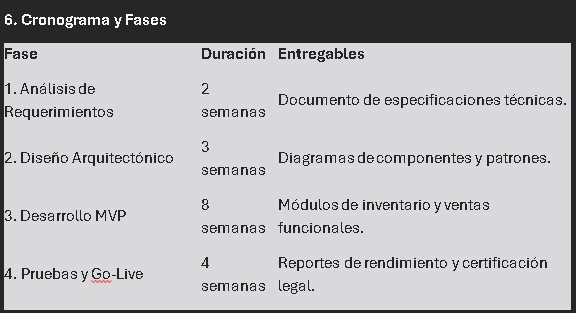
3. Procesamiento de pago (Strategy + Decorator para seguridad).

4. Generación de factura (Facade: Adaptación a normativas locales).

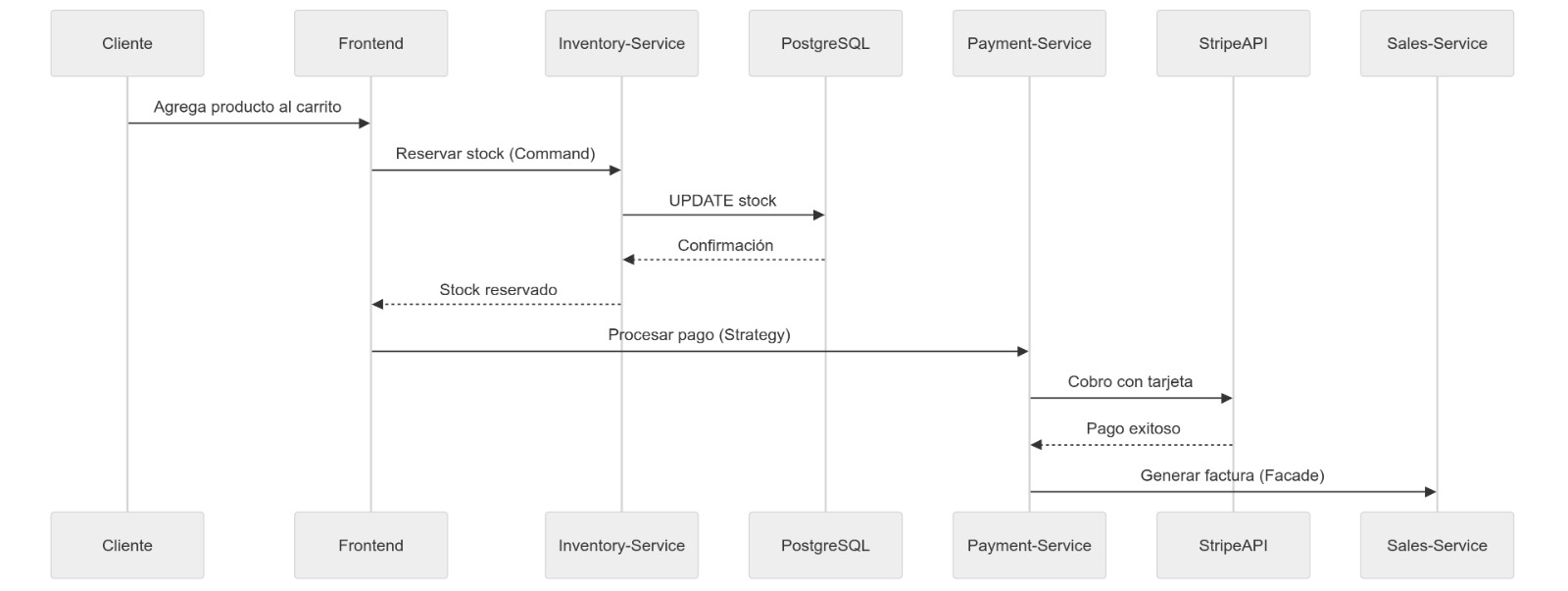
**5. Tecnologías y Herramientas**



**6. Cronograma y Fases**

****

**7. Diagramas Adicionales**

7.1 Secuencia de una venta

**8. Aplicación de Principios SOLID**

**Planeación de Pruebas del Sprint**

**1. Alcance**

* Módulo de Gestión de Inventario
* Pruebas en entorno de desarrollo.
* Pruebas manuales iniciales, con planificación de automatización futura.

**2. Criterios de Aceptación**

* Los usuarios pueden agregar productos con información completa.
* Los productos pueden ser editados y los cambios reflejados correctamente.
* Se pueden eliminar productos y estos desaparecen de la lista de inventario.
* La lista de productos se visualiza correctamente con paginación y filtros.

**3. Casos de Prueba**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Nombre del Caso de Prueba | Descripción | Datos de Prueba | Resultado Esperado | Estado |
| TC001 | Registro de Producto | Registrar un producto con datos válidos | Nombre: Camiseta Roja, Precio: $20, Stock: 10 | Producto registrado correctamente | Pendiente |
| TC002 | Edición de Producto | Editar información de un producto existente | Modificar precio de $20 a $25 | Producto actualizado con éxito | Pendiente |
| TC003 | Eliminación de Producto | Eliminar un producto del inventario | ID Producto: 3 | Producto eliminado y no visible en la lista | Pendiente |
| TC004 | Visualización de Inventario | Ver la lista de productos | Inventario con 5 productos registrados | Productos visibles con paginación | Pendiente |
| TC005 | Validación de Campos Requeridos | Intentar registrar un producto sin nombre | Nombre: (vacío) | Error indicando que el campo es obligatorio | Pendiente |

**4. Herramienta de prueba**

* TestRail

**5. Criterios de Salida**

* Todos los casos de prueba han sido ejecutados y validados. a
* No se presentan errores críticos en la funcionalidad básica.