**[StyleHub]**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Identificación de Documento**

| **Identificación** |  |
| --- | --- |
| **Proyecto** | StyleHub |
| **Versión** | 1.0 |

| **Documento mantenido por** | Yuri Alexis Durán Casanueva |
| --- | --- |
| **Fecha de última revisión** | 26/11/2024 |
| **Fecha de próxima revisión** | 03/12/2024 |

| **Documento aprobado por** | Yuri Alexis Durán Casanueva |
| --- | --- |
| **Fecha de última aprobación** | 03/12/2024 |

**Historia de Revisiones**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 26/11/2024 | 1.0 | Creación inicial del documento | Yuri Durán |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**1**](#_heading=h.30j0zll) **Introducción 3**

[1.1](#_heading=h.1fob9te) Contexto del Problema 3

[1.2](#_heading=h.3znysh7) Propósito 3

[1.3](#_heading=h.2et92p0) Ámbito 3

[1.4](#_heading=h.tyjcwt) Definiciones, acrónimos y abreviaciones 3

[1.5](#_heading=h.3dy6vkm) Referencias 3

[1.6](#_heading=h.1t3h5sf) Resumen ejecutivo 4

[**2**](#_heading=h.2bn6wsx) **Representación de la Arquitectura 5**

[2.1](#_heading=h.4d34og8) Representación 5

[**3**](#_heading=h.2s8eyo1) **Metas y Restricciones de la Arquitectura 6**

[A continuación se revisan las metas y restricciones de la arquitectura. 6](#_heading=h.17dp8vu)

[3.1](#_heading=h.3rdcrjn) Metas de la arquitectura 6

[3.2](#_heading=h.26in1rg) Restricciones de la Arquitectura 6

[3.3](#_heading=h.lnxbz9) Otros antecedentes y consideraciones 6

[**4**](#_heading=h.35nkun2) **Vista de Casos de Uso y Escenarios de Calidad 8**

[4.1](#_heading=h.1ksv4uv) Modelo de Casos de Uso 8

[4.2](#_heading=h.44sinio) Especificación de Casos de Uso Relevantes 8

[4.3](#_heading=h.2jxsxqh) Especificación de los Escenarios de Calidad Relevantes 9

[**Vista Lógica 11**](#_heading=h.z337ya)

[4.4](#_heading=h.4i7ojhp) Parte Estructural 11

[4.5](#_heading=h.1y810tw) Parte Dinámica 11

[**5**](#_heading=h.4i7ojhp) **Vista de Procesos 13**

[**6**](#_heading=h.2xcytpi) **Vista de Implementación 14**

[**7**](#_heading=h.1ci93xb) **Vista de Despliegue 15**

[**8**](#_heading=h.3whwml4) **Decisiones de Diseño y Selección de Alternativas 16**

1. **Introducción**
   1. **Contexto del Problema**

**StyleHub** aborda las problemáticas comunes en el comercio electrónico de ropa:

* Falta de herramientas para PYMEs de moda para gestionar y personalizar catálogos de productos en línea.
* Limitada experiencia interactiva para consumidores al comprar ropa en plataformas digitales, donde no pueden visualizar las prendas en modelos personalizados.

* 1. **Propósito**

Crear una plataforma que permita:

* Visualizar prendas en 3D para mejorar la experiencia de compra.
* Proporcionar herramientas de gestión de PYMEs.
* Facilitar transacciones seguras mediante un portal de pagos integrado.
  1. **Referencias**

La Solución está diseñada para ser accesible a través de navegadores web y dispositivos móviles, orientada a usuarios finales (consumidores) y pequeños negocios del sector moda.

* 1. **Resumen ejecutivo**

El proyecto StyleHub busca revolucionar la experiencia de compra de ropa en línea y facilitar la digitalización de las pequeñas y medianas empresas (PYMEs) del sector moda mediante una plataforma innovadora. Este sistema integra herramientas avanzadas de modelado 3D y gestión de catálogos, ofreciendo una solución tecnológica completa que mejora tanto la experiencia del usuario final como la eficiencia operativa de las PYMEs.

**Problemas identificados:**

* Los consumidores enfrentan dificultades para visualizar prendas de manera realista y personalizada antes de realizar una compra en línea, lo que genera desconfianza y disminuye la satisfacción.
* Las PYMEs carecen de herramientas efectivas para gestionar catálogos y competir en el mercado digital, limitando su crecimiento y alcance.

**Solución propuesta:**

StyleHub es una plataforma web que ofrece:

* **Visualización 3D interactiva:** Los usuarios pueden personalizar y visualizar prendas en un maniquí virtual, mejorando la confianza y satisfacción en sus compras.
* **Gestión eficiente de catálogos:** Las PYMEs pueden administrar sus productos (tallas, colores , precios y stock) mediante herramientas intuitivas.
* **Filtros avanzados y recomendaciones personalizadas:** Facilitan la búsqueda de productos según las preferencias del usuario.
* **Carrito de compras y pagos seguros:** Un portal de transacciones integrado que garantiza confiabilidad y seguridad.

**Impacto esperado:**

* **Para los consumidores:** Una experiencia de compra enriquecida, interactiva y confiable que reduce la incertidumbre al seleccionar prendas.
* **Para las PYMEs:** Mayor visibilidad en el mercado digital, aumento en la eficiencia operativa y posibilidad de competir en igualdad de condiciones con grandes marcas.

* 1. **Representación**

La arquitectura del sistema **StyleHub** está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Casos de Uso y Escenarios de Calidad**: Describe actores, casos de uso y escenarios clave.
* **Vista de Metas y Restricciones**: Expone objetivos del negocio y restricciones tecnológicas.
* **Vista Lógica**: Define módulos y sus responsabilidades.
* **Vista de Procesos**: Explica flujos de datos y relaciones entre componentes.
* **Vista de Implementación**: Muestra cómo se implementarán los servicios en infraestructura técnica.

1. **Metas y Restricciones de la Arquitectura**

* 1. **Metas de la arquitectura**

* Crear un sistema ágil y seguro que permita a PYMEs gestionar productos de manera eficiente.
* Ofrecer una experiencia interactiva y personalizada a los usuarios finales mediante visualización 3D.

* 1. **Restricciones de la Arquitectura**
* Tecnologías utilizadas:
  + Frontend: HTML5, CSS3, Three.js, JavaScript.
  + Backend: PHP, MySQL.
* Seguridad: Protección de datos de usuarios y cifrado de transacciones.

1. **Vista de Casos de Uso y Escenarios de Calidad**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

* 1. **Modelo de Casos de Uso**

| Código | Nombre | Actor | Prioridad |
| --- | --- | --- | --- |
| 01 | Visualización 3D de prendas | Consumidor | Alta |
| 02 | Gestión de catálogos | PYME | Alta |
| 03 | Procesamiento de pagos | Consumidor | Alta |

* 1. **Especificación de los Escenarios de Calidad Relevantes**

Los escenarios de calidad son:

* **Escenario de rendimiento:** El sistema debe responder rápidamente a las solicitudes de los usuarios, garantizando tiempos de carga adecuados para todas la funcionalidades. El atributo de calidad sería, el rendimiento, por otro lado tenemos el estímulo, ya que un usuario realiza acciones comunes como cargar un catálogo, visualizar una prenda en 3D o procesar un pago.
* **Escenario de Usabilidad:** La plataforma debe ser fácil de usar, tanto para los consumidores como para las PYMEs que administran catálogos. El atributo de calidad sería, la usabilidad, por otro lado tenemos el estímulo, ya que un usuario intenta realizar una tarea como agregar un producto al carrito, cargar un catálogo o visualizar prendas en 3D.
* **Escenario de seguridad:** Proteger los datos sensibles de los usuarios y garantizar la seguridad en las transacciones. El atributo de calidad sería la seguridad, por otro lado tenemos el estímulo, un usuario realiza una transacción o proporciona información personal al registrarse.
* **Escenario de escalabilidad:** El sistema debe ser capaz de manejar incrementos en la carga de usuarios sin degradar su rendimiento. El atributo de calidad sería la escalabilidad, por otro lado tenemos el estímulo, un aumento en el tráfico web de 500 a 5,00 usuarios concurrentes.
* **Escenario de mantenibilidad:** El sistema debe ser fácil mantener y permitir actualizaciones sin afectar las funcionalidades existentes. El atributo de calidad sería la mantenibilidad, por otro lado tenemos un desarrollador realiza cambios en el código o actualiza una funcionalidad.

1. **Vista Lógica**

La vista lógica describe los componentes principales del sistema y sus interacciones a nivel de software. Esta perspectiva se centra en los módulos que componen el sistema, sus responsabilidades y cómo colaboran para cumplir los objetivos del proyecto.

* Componentes principales
  + Frontend:
  + Responsabilidad:
    - Proveer una interfaz intuitiva y amigable para los usuarios finales (consumidores y PYMEs).
    - Soportar las funcionalidades interactivas, como la visualización de prendas en 3D y el manejo de catálogos.
  + Tecnologías:
    - HTML5, CSS3, JavaScript, Three.js (para modelado 3D).
  + Subcomponentes:
    - **Interfaz de usuario (UI):** Pantallas para navegación, compra, personalización y administración.
    - **Gestión de interacciones:** Scripts para manejar la interacción del usuario con el modelo 3D, formularios y otros elementos dinámicos.
  + Backend:
    - **Responsabilidad**:
      * Gestionar la lógica de negocio y las interacciones entre el frontend y la base de datos.
      * Manejar la autenticación, la administración de catálogos y las transacciones.
    - **Tecnologías:**
      * Node.js (framework backend), Express.js (para APIs).
    - **Subcomponentes**:
      * Módulo de gestión de Usuario: Autenticación, creación y manejo de roles.
      * Módulo de gestión de catálogos: CRUD (Crear, Leer, Actualizar, Eliminar) de productos.
      * Módulo de transacciones: Procesamiento de pagos y confirmación de órdenes.
      * Api para modelado 3D: Punto de conexión para manejar datos relacionados con los modelos 3D.
  + Base de datos:
    - **Responsabilidad:**
      * Almacenar de manera estructurada toda la información del sistema, incluyendo datos de usuarios, productos, transacciones y configuraciones.
    - **Tecnologías:**
      * MySQL o PostgreSQL (base de datos relacional).
    - **Subcomponentes:**
      * Tabla de Usuarios: Información de registro, roles y preferencias.
      * Tabla de productos: Datos de catálogo, tallas, colores, imágenes y stock.
      * Tabla de transacciones: Historial de compras y estados de pagos.
  + Servicios externos:
    - **Responsabilidades:**
      * Integrar funcionalidades de terceros para mejorar la experiencia del sistema.
    - **Ejemplos:** 
      * Pasarela de Pagos: Stripe o Paypal para transacciones seguras.
      * CDN para imágenes: Almacenamiento eficiente de imágenes de alta resolución para prendas.

**Diagrama Lógico**

Un diagrama lógico incluiría los siguientes elementos:

* **Frontend:** Interfaz que interactúa con el usuario y se comunica con el backend mediante APIs.
* **Backend:** Intermediario que procesa solicitudes del frontend y responde con datos de la base de datos o servicios externos.
* **Base de datos:** Contiene tablas relacionadas con los usuarios, catálogos y transacciones.
* **Servicios externos:** APIs de terceros para pagos, almacenamiento o cualquier otra funcionalidad externa.

**Interacción entre componentes**

**1.- Ciclo de compra:**

* El usuario selecciona un producto desde el catálogo en el frontend.
* El backend consulta la base de datos para obtener los detalles del producto.
* Si el usuario completa la compra, el backend se comunica con la pasarela de pagos para procesar la transacción.

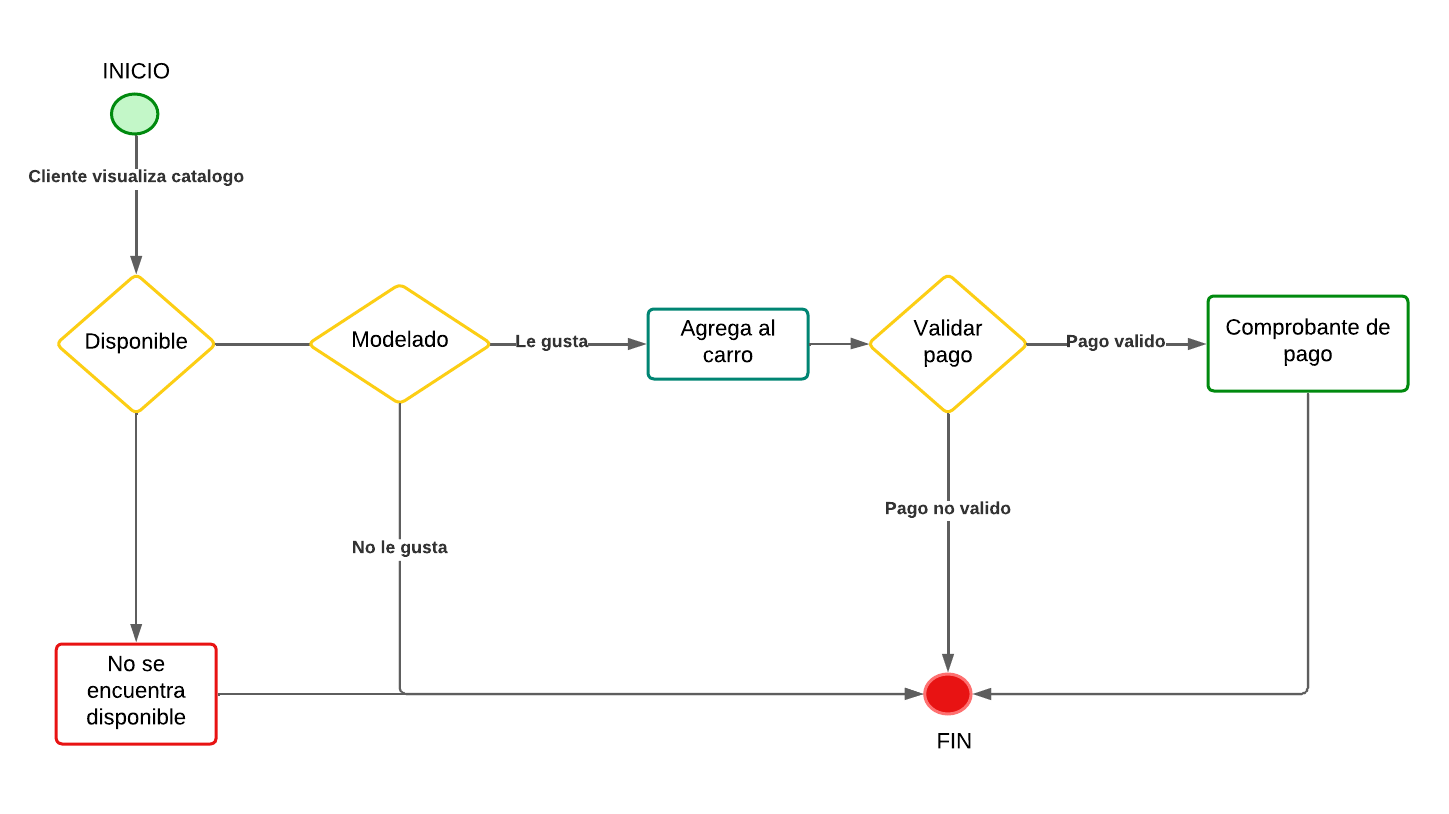
**2.- Gestión de Catálogos:**

* Una PYME sube un nuevo producto desde el frontend.
* El backend valida los datos y los almacena en la base de datos.
* Las imágenes del producto son almacenadas en un CDN externo.

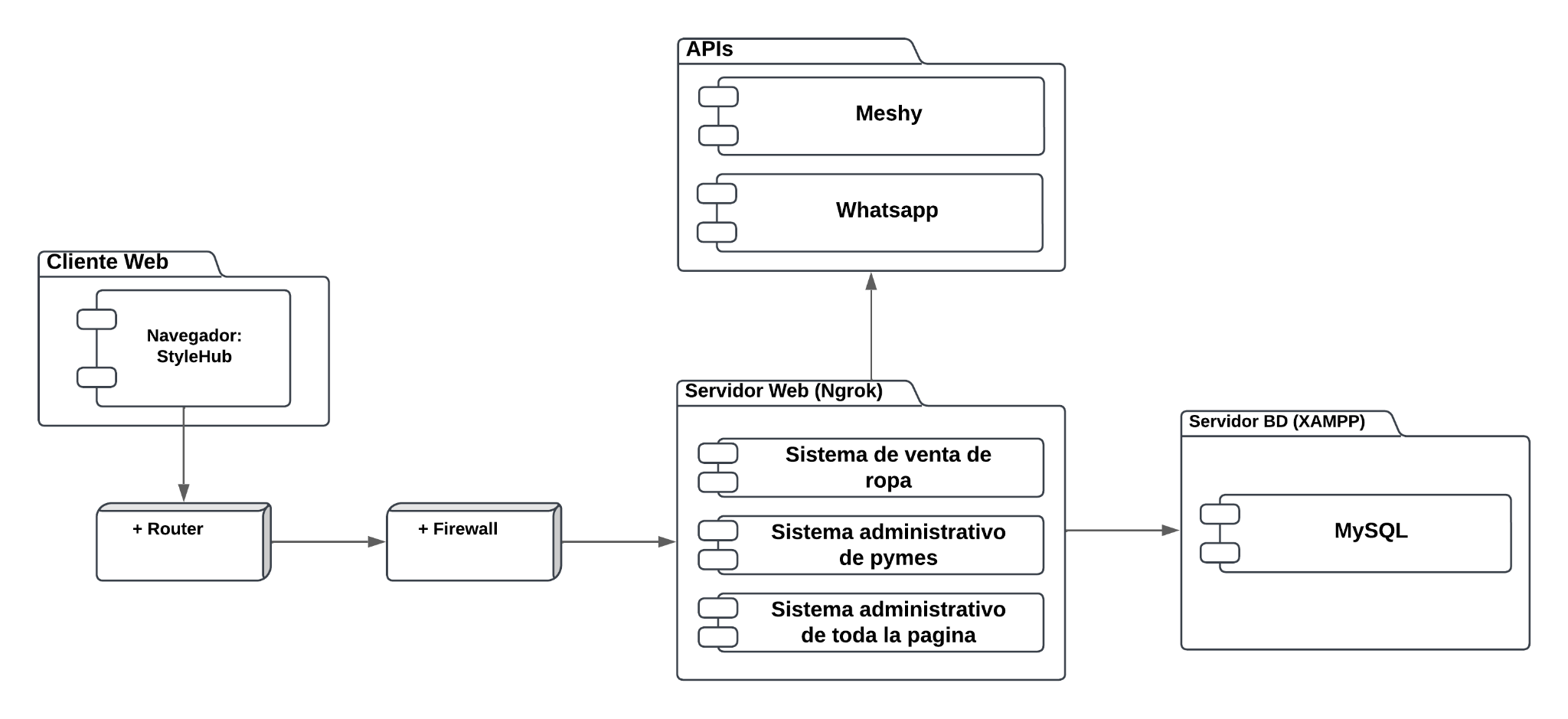
**3.- Visualización 3D:**

* El usuario interactúa con el modelo 3D en el frontend mediante Three.js.
* El backend proporciona los datos y ajustes necesarios para la configuración del modelo.

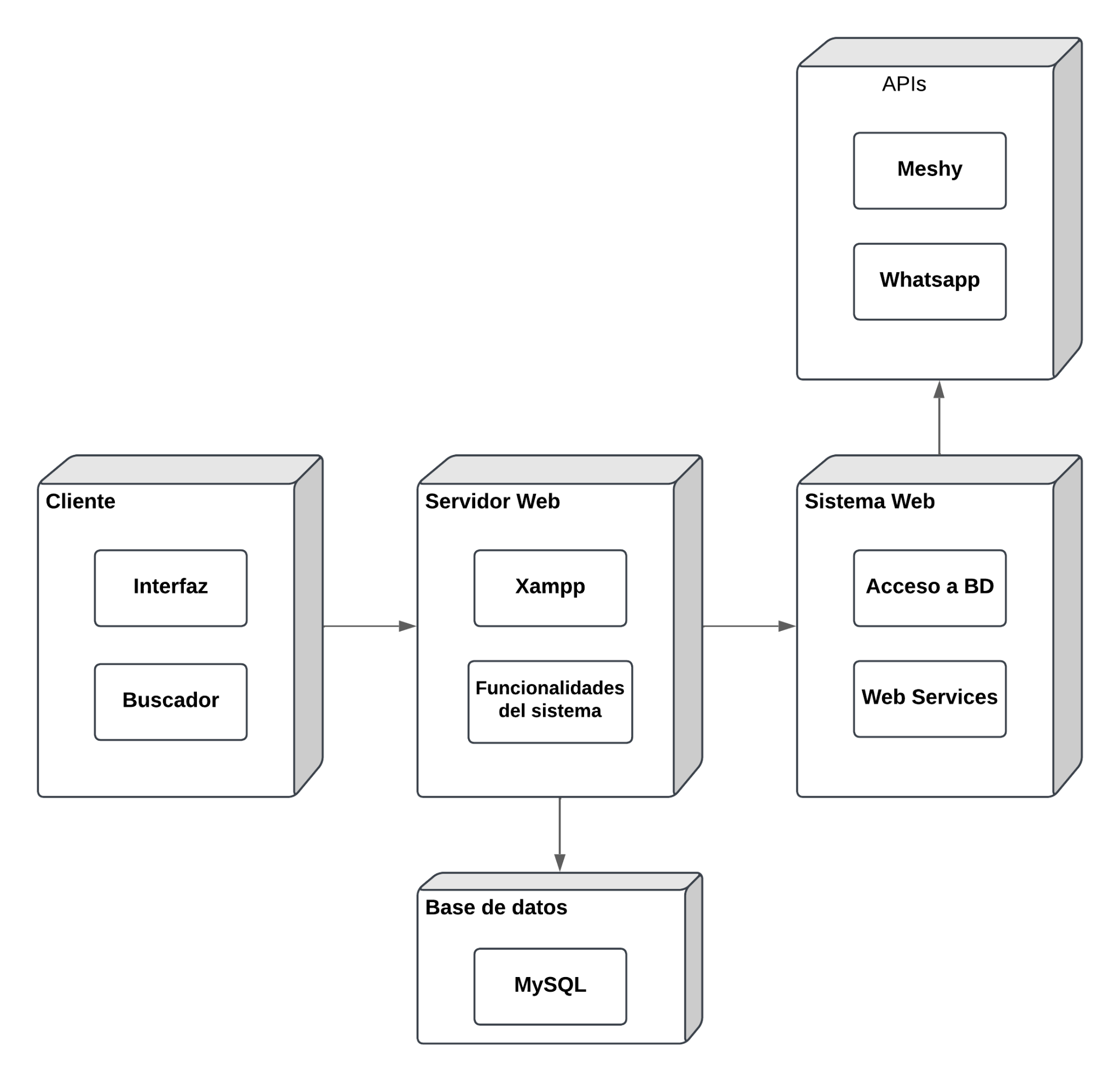
1. **Vista de Procesos**

****

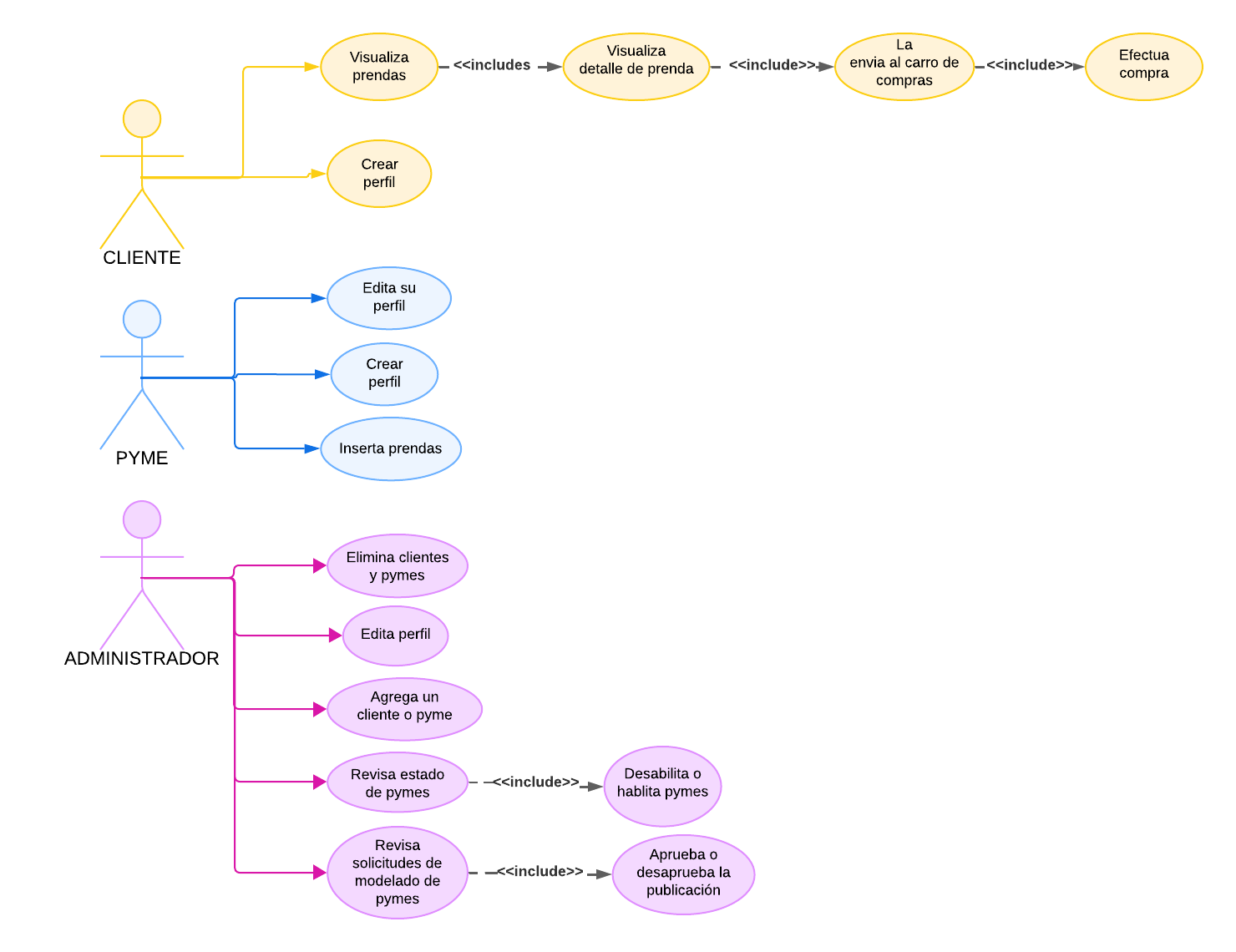
1. **Vista de Implementación**



1. **Vista de Despliegue**

****

1. **Vista de escenarios**

****

1. **Decisiones de Diseño y Selección de Alternativas**

En el diseño del sistema StyleHub, se tomaron decisiones clave para garantizar la efectividad , escalabilidad y facilidad de mantenimiento de la plataforma. Estas decisiones se basaron en las necesidades del proyecto, las capacidades del equipo y los recursos disponibles.

1. **Arquitectura del Sistema**
   * Decisión: Adoptar una arquitectura basada en Modelo-Vista-Controlador (MVC) para organizar el sistema.
   * Razón: Facilita la separación de responsabilidades entre el frontend, la lógica del negocio y la base de datos.
   * Mejora la mantenibilidad y escalabilidad del sistema.
2. **Tecnologías seleccionadas**
   * **Frontend:**
     + HTML5, CSS3 y JavaScript:
       - Permiten crear una interfaz de usuario moderna y accesible.
       - Se integran fácilmente con bibliotecas especializadas como Three.js para visualizar 3D.
     + Three.js:
       - Proporciona herramientas para modelado y renderizado 3D en navegadores web.
       - Ideal para la funcionalidad de visualización de prendas en 3D.
   * **Backend:**
     + PHP:
       - Utilizado para gestionar la lógica del negocio, los controladores del sistema y las API.
       - Permite una integración eficiente con bases de datos relacionales como MySql, optimizando el manejo de datos.
   * **Base de datos:**
     + MySQL:
       - Sistema de gestión de bases de datos relacional.
       - Justificación: Es confiable, rápido y ampliamente soportado por PHP.
   * **Servicios Externos:**
     + Almacenamiento de imágenes y modelos:
       - Decisión: Uso de carpetas locales como uploads para guardar archivos de los usuarios.
       - Razón: Centraliza el almacenamiento en el servidor, simplificando la recuperación de datos.
       - Futuro: Se considera migrar a un servicio en la nube para mejorar la escalabilidad.
3. **Diseño de la interfaz de usuario (UI)**
   * Decisión: Adoptar un diseño moderno y minimalista.
   * Razón: Facilita la navegación y experiencia del usuario en dispositivos móviles y de escritorio.
4. **Seguridad**
   * Validación de entradas:
     + Decisión: implementar validaciones tanto como en frontend como backend.
     + Razón: Previene ataques con inyecciones SQL o cargas maliciosas.
   * Autenticación y autorización:
     + Decisión: Uso de sesiones para autenticar usuarios registrados y roles (Administradores, Usuarios, Pymes).
     + Razon: Asegurar el acceso de funcionalidades según los permisos asignados.
5. **Estrategia de Escalabilidad**
   * Decisión: Modularización del código para facilitar futuras integraciones y escalabilidad.
   * Razón: Permite incorporar nuevas funcionalidades, como recomendaciones personalizadas o manejo avanzado de datos.
6. **Selección de Frameworks y Librerías**
   * Decisión: Usar bibliotecas nativas como Three.js y librerías ligeras para optimizar el rendimiento.
   * Razón: Asegurar que el sistema se mantenga rápido, incluso en dispositivos de menor capacidad.
7. **Diseño de la Experiencia 3D**
   * Decisión: uso de modelos GLTF y Three.js para la renderización en tiempo real.
   * Razón: Estos formatos son ligeros y ampliamente soportados, permitiendo una visualización fluida.
8. **Estrategia de Pruebas**
   * Pruebas unitarias:
     + Decisión: Validar la funcionalidad de módulos clave (e.g., carga de modelos, autenticación).
     + Razón: Garantiza la fiabilidad de componentes individuales.
   * Pruebas de integración:
     + Decisión: Verificar la interacción entre el backend, base de datos y el frontend.
     + Razón: Asegurar que los componentes trabajen de manera conjunta sin problemas.
   * Pruebas de usuario final:
     + Decisión: Realizar pruebas con usuarios representativos.
     + Razón: Identifica posibles problemas en la experiencia del usuario.

Estas decisiones refuerzan la estructura del proyecto y aseguran que StyleHub sea eficiente, escalable y fácil de mantener, cubriendo las funcionalidades principales con un diseño moderno y seguro.

1. **Análisis de Reutilización**

En el desarrollo de StyleHub, se adoptó una estrategia prioriza la reutilización de componentes, tecnologías y conocimientos, lo que permite ahorrar tiempo, reducir costos y aumentar la eficiencia del proyecto. A continuación, se detallan los aspectos clave de reutilización implementados:

**1.- Código Reutilizable**

* Componentes de Frontend:
  + Las plantillas de HTML y CSS para las vistas de la página se desarrollaron de manera modular, permitiendo su reutilización en múltiples secciones del sitio web (e.g., catálogo, perfil de usuario, carrito de compras).
* Scripts JavaScript:
  + Las funciones desarrolladas para interactuar con Three.js (e.g, renderizado de modelos GLTF, personalización de prendas) son reutilizables en cualquier vista que requiera visualización 3D.
* Componentes del Backend:
  + Funciones de PHP Reutilizables:
    - Módulos como conexion.PHP, que gestiona la conexión a la base de datos, y funciones comunes para consultas SQL, son reutilizables en diferentes operaciones (e.g, gestión de usuarios, productos y transacciones).
  + APIs y Endpoints:
    - Las APIs desarrolladas para interactuar con el frontend están diseñadas de forma genérica para soportar futuras funcionalidades.

**2.- Tecnologías y Librerías de código abierto**

* Three.js:
  + Librería ampliamente utilizada para renderizado 3D en navegadores web, que proporciona una solución probada y adaptable para el manejo de gráficos 3D.
  + Reutilizada en todo el sistema para la visualización de prendas en 3D, facilitando el mantenimiento y la extensión del sistema.
* MySQL:
  + Base de datos relacional utilizada para gestionar los datos del sistema. Su naturaleza reutilizable permite manejar diferentes tipos de datos como usuarios, productos y compras con un enfoque unificado.

**3.- Reutilización de modelos y recursos digitales**

* **Modelado 3D (GLTF):**
  + Los modelos de prendas almacenados en la carpeta modelos 3D son reutilizables para diferentes usuarios y escenarios de visualización.
  + Los archivos GLTF son livianos y optimizados para su uso en plataformas web, reduciendo la necesidad de desarrollar modelos exclusivos para cada prenda.
* **Imágenes y archivos de usuario:**
  + El sistema de almacenamiento en la carpeta de uploads permite reutilizar imágenes subidas por los usuarios para diferentes funciones, como personalización, historial de compras y marketing.

**4.- Estándares y buenas prácticas**

* **Estandarización de código:**
  + Uso de patrones de diseño como MVC para separar la lógica del sistema, facilitando la reutilización de controladores, vistas y modelos.
  + Código modular y organizado que permite escalar el sistema sin reescribir las funciones existentes.
* **Uso de librerías comprobadas:**
  + Reutilización de librerías de terceros, como Bootstrap (si fue usado) para estilos y diseño responsivo, lo cual acelera el desarrollo sin necesidad de crear estilos desde cero.

**5.- Escalabilidad y Futuro**

* **Reutilización en nuevas funcionalidades:**
  + Las APIs desarrolladas y los modelos 3D actuales pueden adaptarse para futuras expansiones, como recomendación personalizada o generación de modelos mediante inteligencia artificial.
* **Manejo de datos:**
  + La estructura actual de base de datos es reutilizable para soportar nuevos módulos como gestión avanzada de usuarios, integración con sistemas de pagos o analítica de datos.

**6.- Beneficios de la reutilización**

* **Reducción de Costos y tiempo:**
  + Al reutilizar librerías, plantillas y funciones, el tiempo de desarrollo se reduce significativamente, disminuyendo también los costos asociados.
* **Consistencia:**
  + El uso de componentes comunes asegura una experiencia consistente para los usuarios y facilita el mantenimiento del sistema.
* **Mantenimiento y Extensibilidad:**
  + La modularidad del código y la reutilización de tecnologías probadas facilitan la incorporación de nuevos desarrolladores al equipo y la expansión del sistema en el futuro.

En resumen, el enfoque en la reutilización ha sido fundamental para maximizar la eficiencia y sostenibilidad del desarrollo de StyleHub, permitiendo construir una plataforma sólida y escalable con un uso inteligente de los recursos disponibles.