**DOCUMENTACIÓN INTEGRACIÓN**

**PONENTES**

**JUAN DAVID SERRANO**

**MIGUEL ANGEL ROMERO**

**LAURA GAONA CASTAÑO**

**INGENIERIA DE SOFTWARE**

**UNIVERSIDAD MANUELA BELTRAN**

**2025**

**1. Introducción**

Esta documentación describe el proceso de integración del sitio web de la tienda online de libros **Entrelineas**, centrándose en la conexión entre el **FrontEnd, BackEnd y la base de datos**, además de la gestión del control de versiones con **Git**. La correcta integración de estos elementos es esencial para garantizar la estabilidad, eficiencia y sincronización del sistema, facilitando el desarrollo colaborativo.

**2. Estructura del Proyecto**

El repositorio del proyecto se organiza en distintas ramas, cada una con un propósito específico:

* **FrontEnd:** Contiene los archivos HTML, CSS y JavaScript, utilizando frameworks como **React** o **Vue.js** para la construcción de la interfaz de usuario.
* **BackEnd:** Incluye la lógica del servidor desarrollada en **Node.js con Express.js**, así como la configuración y conexión con la base de datos.
* **Documentación:** Almacena toda la información relevante sobre el desarrollo y la estructura del proyecto.
* **Integración:** Es la rama donde se unen todas las partes del sistema y se realizan pruebas antes del despliegue de nuevas versiones.

**3. Pruebas de Integración**

Antes de lanzar una nueva versión, se ejecutan pruebas en la rama de integración para validar el correcto funcionamiento del sistema. Algunas pruebas clave incluyen:

* **Verificación de la comunicación entre FrontEnd y BackEnd:** Se comprueba que las solicitudes y respuestas sean correctas.
* **Validación del almacenamiento de datos:** Se revisa que la base de datos almacene y recupere información correctamente.
* **Revisión del control de versiones con Git:** Se garantiza que todas las modificaciones se integren sin conflictos y de manera organizada.

**4. Pruebas con Postman**

Para validar la funcionalidad del BackEnd, se usa **Postman**, una herramienta que permite realizar pruebas sobre APIs RESTful de manera eficiente. A través de Postman, se ejecutan solicitudes **HTTP (GET, POST, PUT, DELETE)** para verificar el correcto funcionamiento del servidor antes de integrarlo con el FrontEnd.

**Ventajas del uso de Postman:**

* Permite probar el BackEnd sin depender del FrontEnd.
* Facilita la identificación y corrección de errores en la comunicación con la base de datos.
* Posibilita la automatización de pruebas mediante colecciones de solicitudes.

**5. Integración de APIs y Comunicación entre Componentes**

Para conectar la base de datos con el BackEnd, se desarrolla una **API REST** en **Node.js con Express.js**, encargada de gestionar autenticaciones, validación de datos y operaciones **CRUD**.

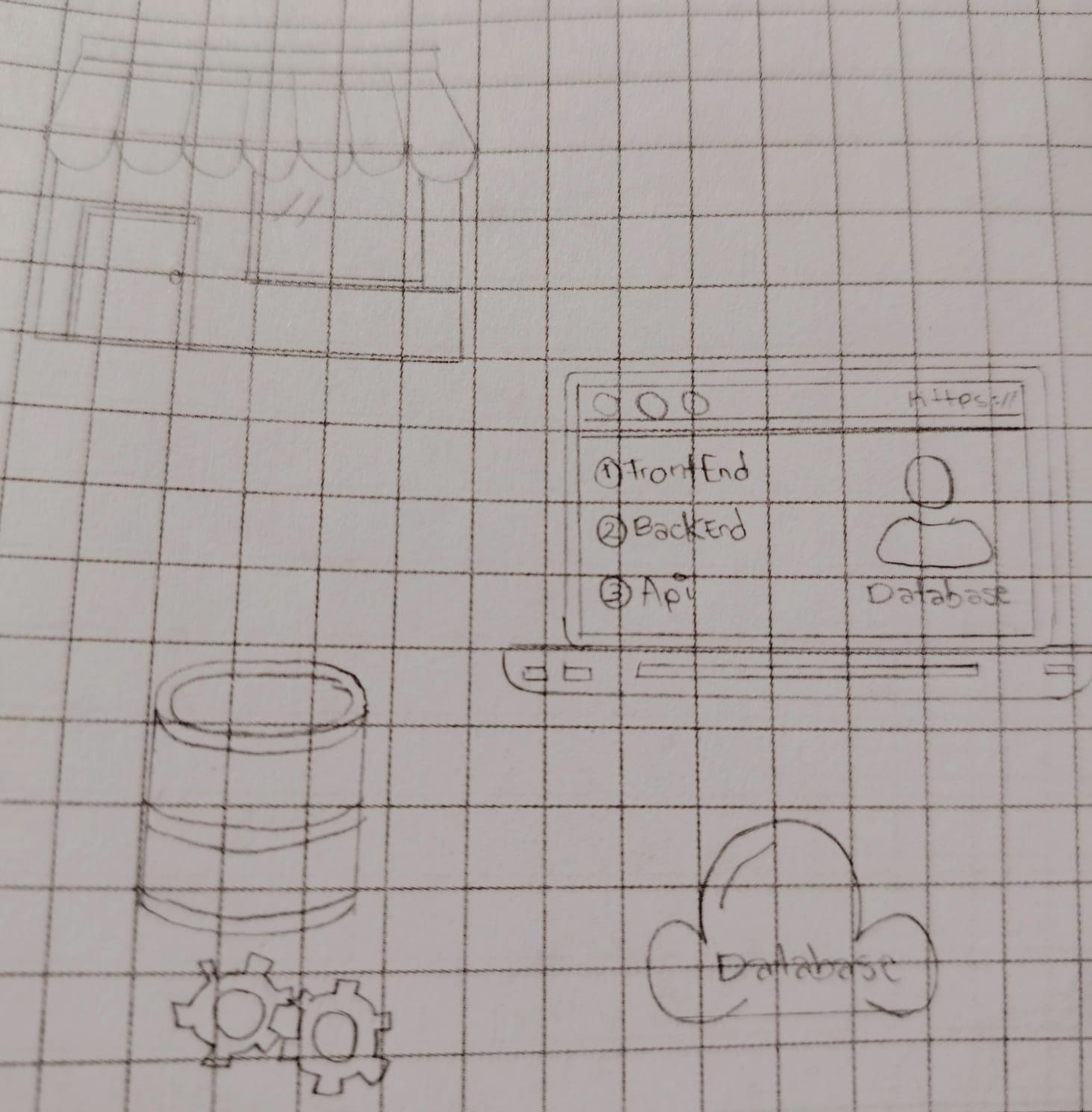
Para la conexión entre el FrontEnd y el BackEnd, se usa la función **fetch()** de JavaScript, que permite el envío y recepción de datos de manera asincrónica. Esto es crucial para un flujo de información eficiente y una mejor experiencia de usuario.

**Consideraciones clave en la integración:**

* Configuración de solicitudes HTTP desde el FrontEnd al BackEnd.
* Manejo adecuado de respuestas para actualizar dinámicamente la interfaz.
* Implementación de **JWT (JSON Web Token)** para una autenticación segura.

Una integración efectiva entre el FrontEnd y el BackEnd es fundamental para el buen funcionamiento del sistema. El uso de herramientas como **Postman** para pruebas de API y **fetch ()** para la comunicación asincrónica optimiza la validación de la funcionalidad antes del despliegue.

A continuación de muestran los diagramas en la base de datos y La utilización de API’s:



**Figura 1**

1. **Interfaz de usuario**

* Representado por la pantalla del navegador.
* Es el par
* Útil HTML, CSS y JavaScript.

1. **Backend (Lógica y Procesamiento)**

* Maneja las reglas de negocio y la lógica del sistema.
* Proceso
* Pue Python, Java, Node.js, etc.

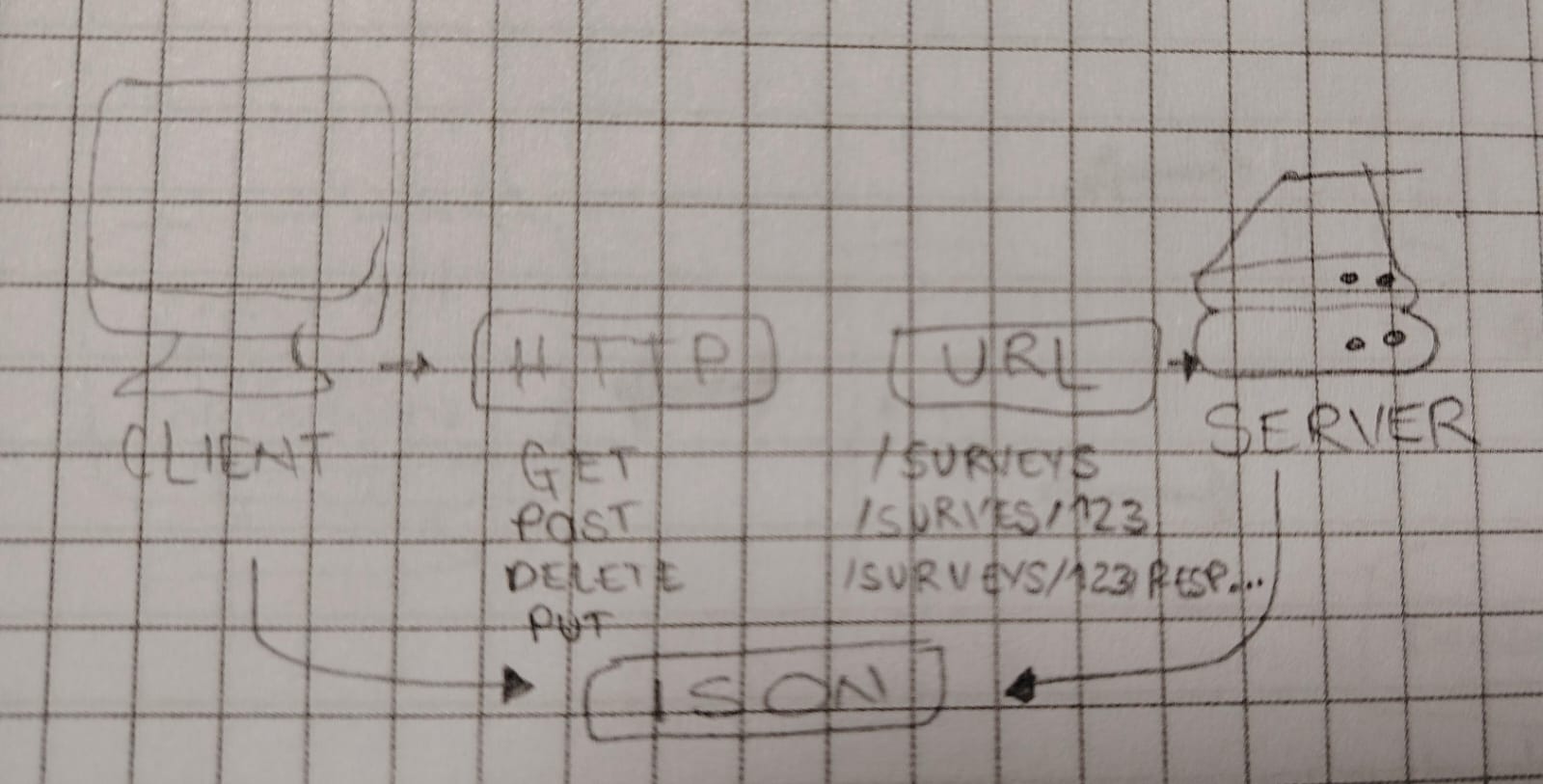
1. **API (Intermediario de Comunicación)**

* Permite la comunicación entre el frontend y el backend.
* Usa protocolos como HTTP y formatos como JSON para el intercambio de datos.

1. **Base de Datos (Almacenamiento de Información)**

* Representada por el ícono de almacenamiento.
* Guarda y gestiona la información del sistema.
* Puede ser SQL (MySQL, PostgreSQL) o NoSQL (MongoDB, Firebase).

1. **uso**
2. Un usuario ingresa a una tienda en línea (frontend), navega por los productos y realiza una compra.
3. El backend procesa la compra y la API comunica los datos a la base de datos.
4. La base de datos almacena la compra y devuelve la confirmación al usuario.



**Figura 2**

**Flujo de comunicación**

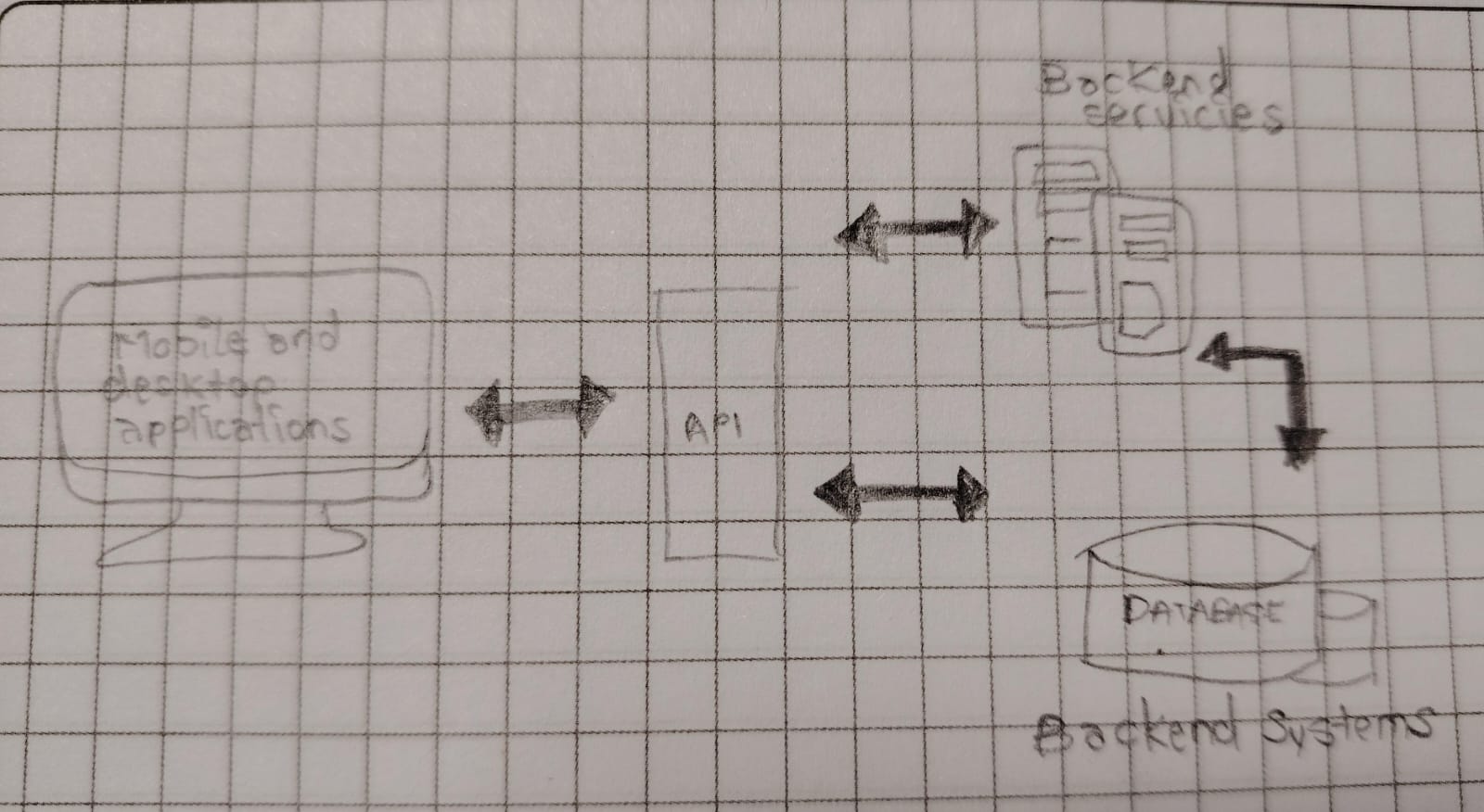
1. El Cliente (navegador, aplicación móvil, etc.) realiza una petición al servidor.
2. La comunicación se hace vía HTTP, utilizando métodos como:

* OBTENER
* CORREO
* ELIMINAR
* PONER

1. Las solicitudes usan URL específicas, como:

* /surveys
* /surveys/123
* /surveys/123/resp

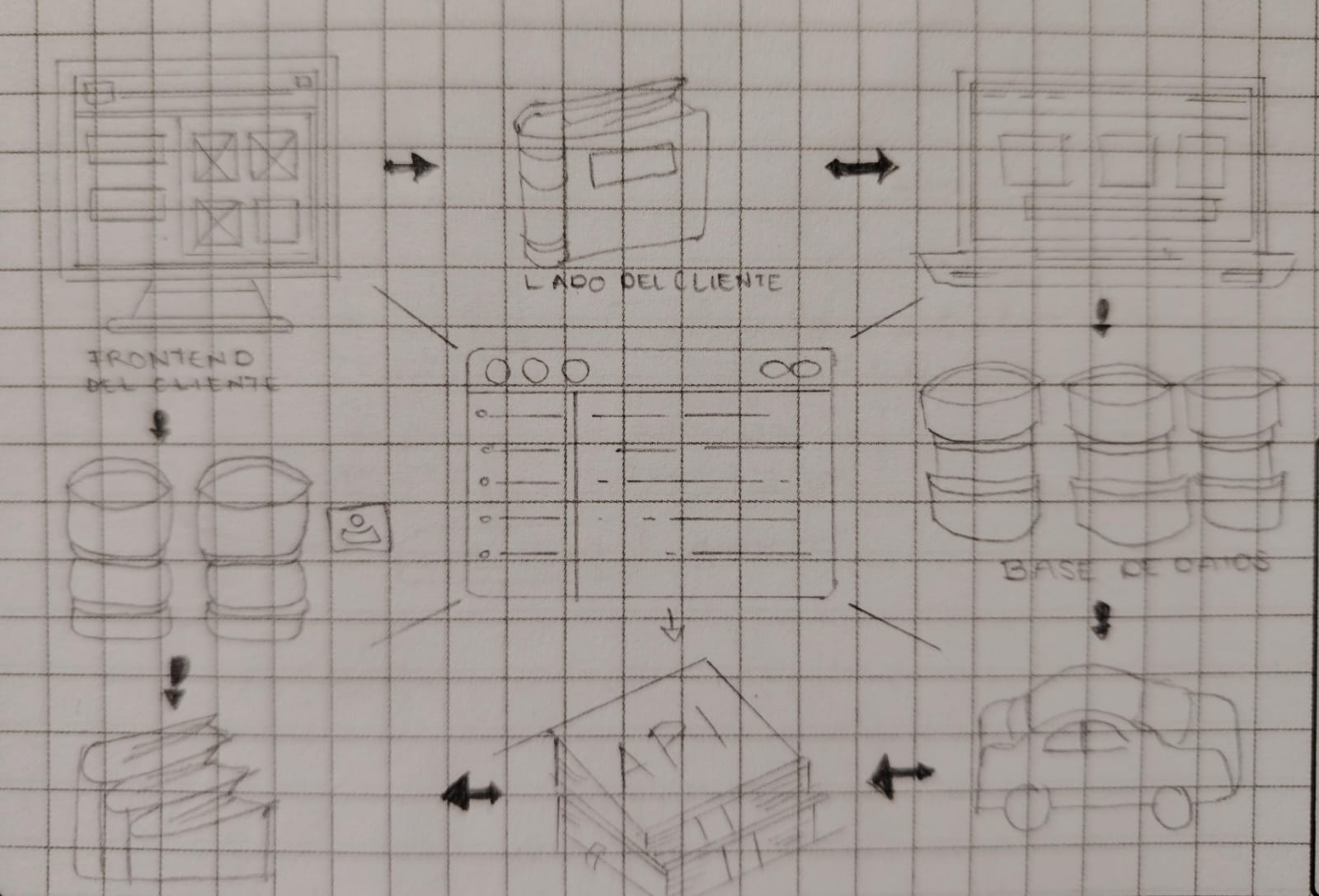
1. El servidor procesa la solicitud y devuelve los datos en formato JSON.



**Figura 3**

**Flujo de datos**

1. Las aplicaciones móviles y de escritorio envían solicitudes a la API.
2. La API actúa como intermediaria, gestionando las solicitudes y redirigiéndolas a los servicios del BackEnd.
3. Los servicios del BackEnd procesan la información y, si es necesario, consultan la Base de Datos.
4. La Base de Datos responde con la información requerida.
5. El BackEnd devuelve los datos a la API, que los formatea y envía de vuelta a la aplicación.



**Figura 4**

1. **Componentes principales**

* FrontEnd (Cliente) → InterfazReact/Vue.js.
* BackEnd (Servidor) → Procesa solicitudes con Node.js/Express.js, maneja la lógica y consulta la base de datos.
* Base de Datos → Almacena información sobre libros y usuarios. Gestión de operaciones CRUD.
* Documentación → Contiene información clave del proyecto para facilitar el desarrollo colaborativo.
* Flujo de Datos → Representa el intercambio de información entre los componentes.
* Pruebas de Integración → Se validan las conexiones con Postman antes del despliegue.
* Control de Versiones (Git) → Organiza el desarrollo y evita conflictos en el código.

1. **Flujo de integración**
2. El usuario interactúa con el FrontEnd (ej. búsqueda de libros).
3. El FrontEnd envía una solicitud al BackEnd vía fetch ().
4. El BackEnd procesa la solicitud y consulta la Base de Datos.
5. La Base de Datos responde con los datos al BackEnd.
6. El BackEnd devuelve la información al FrontEnd en formato JSON.
7. El FrontEnd actualiza la interfaz sin recargar la página.
8. Se realiza pruebas con Postman y se gestiona el código con Git.