

**Entrega final – Escenario 7**

**Carlos Eduardo Guzmán Torres**

**Politécnico Grancolombiano**

**Facultad de Ingeniería, Diseño e Innovación**

**Front-End**

**John Alirio Olarte Ramos**

**Bogotá D.C., 04 de octubre de 2025**

## Tabla de Contenidos

|       |  |    |
|-------|--|----|
| 1.    | Introducción .....                                       | 4  |
| 1.1   | Planteamiento del problema.....                          | 4  |
| 1.2   | Objetivos .....  | 5  |
| 1.2.1 | Objetivo general.....                                    | 5  |
| 1.2.2 | Objetivos específicos .....                              | 5  |
| 1.3   | Definición del ambiente de desarrollo .....              | 5  |
| 1.4   | Planificación y alcance .....                            | 6  |
| 1.4.1 | Cronograma de desarrollo.....                            | 6  |
| 1.4.2 | Alcance .....  | 6  |
| 1.4.3 | Entregables del proyecto.....                            | 7  |
| 2.    | Marco Teórico.....                                       | 8  |
| 2.1   | Diseño Web .....   | 8  |
| 2.1   | Tecnologías Web .....                                    | 8  |
| 2.1.1 | HTML .....   | 8  |
| 2.1.2 | CSS .....  | 9  |
| 2.1.3 | JavaScript.....  | 9  |
| 2.2   | Mockups en diseño de interfaces .....                    | 10 |
| 2.3   | Diseño centrado en el usuario (UX).....                  | 10 |
| 3.    | Metodología .....  | 11 |
| 3.1   | Análisis de los requerimientos .....                     | 11 |
| 3.1.1 | Requerimientos funcionales.....                          | 11 |
| 3.1.2 | Requerimientos no funcionales.....                       | 12 |
| 3.2   | Diseño de la Solución .....                              | 13 |
| 3.2.1 | Desarrollo de los MockUps. ....                          | 13 |
| 3.3   | Implementación del Frontend .....                        | 16 |
| 3.3.1 | Configuración del entorno .....                          | 16 |
| 3.3.2 | Frontend: Explicación de las páginas desarrolladas ..... | 17 |
| 3.4   | Repositorio y Código Fuente .....                        | 28 |
| 3.5   | Persistencia de información con localStorage .....       | 30 |
| 3.5.1 | Implementación del Sistema de Persistencia Local .....   | 30 |
| 3.5.2 | Arquitectura de Almacenamiento de Datos .....            | 31 |

|       |  |    |
|-------|--|----|
| 3.5.3 | Estructura de Datos implementada .....       | 32 |
| 3.5.4 | Simulación de Operaciones CRUD .....         | 32 |
| 3.5.5 | Funcionalidades avanzadas implementadas..... | 33 |
| 3.5.6 | Manejo de Estados y Validación.....          | 34 |
| 3.5.7 | Ventajas y Limitaciones.....                 | 34 |
| 3.6   | Despliegue en la Nube .....                  | 35 |
| 3.6.1 | Selección de Plataforma de Hosting .....     | 35 |
| 3.6.2 | Preparación del Proyecto para Deploy.....    | 35 |
| 3.6.3 | Proceso de configuración de Vercel .....     | 36 |
| 3.6.4 | Configuración de Routing.....                | 37 |
| 3.6.5 | Automatización del Deployment .....          | 38 |
| 3.6.6 | Configuración de Dominio y Acceso.....       | 39 |
| 3.7   | Documentación de la aplicación .....         | 41 |
| 3.7.1 | Manual de Usuario.....                       | 41 |
| 3.7.2 | Estructura de la Documentación .....         | 41 |
| 3.7.3 | Documentación Técnica.....                   | 41 |
| 4.    | Conclusiones .....                           | 42 |
| 5.    | Referencias.....                             | 43 |

## **1. Introducción**

La empresa Kibu, dedicada a la prestación de servicios tecnológicos, ha identificado la necesidad de ampliar su cobertura comercial y fortalecer su presencia digital mediante el uso de soluciones web modernas. En la actualidad, gran parte de los procesos de promoción y comunicación empresarial se realizan de forma digital, lo que convierte a las aplicaciones web en herramientas estratégicas para la competitividad y el posicionamiento en el mercado. Ante este panorama, se plantea la creación de una aplicación web que no solo actúe como vitrina de servicios, sino que también permita una gestión más eficiente de la información interna y la interacción con los usuarios.

El desarrollo de esta aplicación no solo responde a una necesidad de crecimiento empresarial, sino que también busca consolidar una solución tecnológica flexible, intuitiva y accesible. A través de esta plataforma se pretende optimizar la difusión de los servicios, garantizar una administración adecuada mediante un sistema CRUD y ofrecer una experiencia de usuario atractiva que fortalezca la identidad digital de Kabul. El presente documento describe la propuesta metodológica, los objetivos, el diseño de los mockups, así como la planificación del proyecto, con el fin de sentar las bases para su implementación y futura escalabilidad.

### **1.1 Planteamiento del problema**

La empresa Kibu no cuenta con una aplicación web que le permita mostrar de forma clara su portafolio de servicios ni gestionar eficientemente la información interna. Esta ausencia limita su alcance comercial, dificulta la interacción con los clientes y la actualización de datos, lo que reduce su competitividad frente a empresas del sector que ya utilizan soluciones digitales.

## **1.2 Objetivos**

### ***1.2.1 Objetivo general***

Desarrollar una aplicación web para la empresa Kibu que permita la visualización y gestión de sus servicios tecnológicos, garantizando una experiencia de usuario intuitiva y una administración eficiente mediante un sistema CRUD conectado a una base de datos.

### ***1.2.2 Objetivos específicos***

1. Diseñar la interfaz gráfica de la aplicación web mediante mockups que integren las funcionalidades requeridas.
2. Implementar un sistema de gestión de servicios (CRUD) que permita al administrador crear, actualizar, eliminar y consultar los servicios de la empresa.
3. Configurar y estructurar una base de datos relacional que permita almacenar información de usuarios y servicios, garantizando la integridad y seguridad de los datos.

## **1.3 Definición del ambiente de desarrollo**

Para la construcción de la aplicación, se empleará un ambiente de desarrollo web basado en tecnologías modernas y de código abierto. El frontend se desarrollará con HTML5, CSS3 y JavaScript, mientras que el backend se implementará en Node.js con Express, lo que permitirá la gestión dinámica de la información. La base de datos se diseñará en MySQL, garantizando una adecuada administración de usuarios y servicios.

El ambiente de desarrollo se complementará con herramientas de control de versiones como Git y GitHub, además de entornos de prueba locales que simulen la ejecución en un servidor de producción.

## 1.4 Planificación y alcance

La planificación del proyecto se centra en definir las etapas de análisis, diseño, implementación y prueba, asegurando que el producto final cumpla con los requerimientos establecidos por la empresa Kibu.

### 1.4.1 Cronograma de desarrollo

El proyecto se desarrollará en un periodo de 5 semanas, distribuidas de la siguiente manera:

**Tabla 1**  
*Cronograma de desarrollo*

| Semana | Fecha de entrega | Descripción   |
|--------|------------------|---|
| 1      | 9 de septiembre  | Análisis de requerimientos, documento parcial con Normas APA y Mockups  |
| 2      | 16 de septiembre | Configuración del ambiente de desarrollo, diseño de la base de datos  |
| 3      | 23 de septiembre | Desarrollo del Front-End, documento parcial con Normas APA, Directorio Raíz, Código fuente del repositorio de GitHub. |
| 4      | 30 de septiembre | Desarrollo del Back-End integrada a la base de datos.   |
| 5      | 7 de octubre     | Aplicación funcional, repositorio de GitHub, documento final en Normas APA y manual de la aplicación                  |

Nota. Datos elaborados por el autor del documento.

### 1.4.2 Alcance

El proyecto contempla la construcción de un prototipo funcional de aplicación web con las siguientes características:

- Página de inicio con slider, nombre de la empresa, descripción, navbar y footer.
- Catálogo de servicios con al menos 10 elementos que incluyan imagen, nombre y precio.
- Vista de detalle de cada servicio, con información ampliada (cantidad, descripción, promociones).
- Login de administrador con autenticación básica.
- Panel de administración con funcionalidades CRUD sobre los servicios.
- Base de datos relacional con tablas para usuarios y servicios.

El sistema no contempla en esta primera fase módulos de pago en línea, integración con plataformas externas ni escalabilidad a aplicaciones móviles.

#### ***1.4.3 Entregables del proyecto***

Los entregables definidos para el proyecto son:

1. Documento técnico con análisis de requerimientos y mockups.
2. Prototipo funcional de la aplicación web.
3. Base de datos estructurada con tablas de usuarios y servicios.
4. Manual de la aplicación de 20 páginas, para usuario y administrador.
5. Repositorio de GitHub con el código fuente de la aplicación.

## **2. Marco Teórico**

### **2.1 Diseño Web**

El diseño web es la disciplina que combina creatividad, tecnología y estrategia para crear experiencias digitales efectivas y atractivas. Va más allá de hacer que una página "se vea bonita": implica planificar cómo los usuarios interactuarán con el contenido, cómo navegarán por la información y cómo lograrán sus objetivos de manera intuitiva. En el diseño web se combinan diferentes aspectos como la experiencia de usuario, el diseño visual, la organización del contenidos y la implementación de las tecnologías como HTML, CSS y JavaScript.

### **2.1 Tecnologías Web**

#### **2.1.1 *HTML***

HTML ha sido una tecnología que ha estado presente desde incluso antes del auge del internet accesible a nivel mundial. Es importante precisar que HTML no es como tal un lenguaje de programación sino que consiste en un lenguaje de etiquetas. Según Tabarés (2012):

Es un lenguaje de marcación de texto que permite al navegador conectado interpretar la página que solicita al servidor. No es un lenguaje de programación y no tiene compiladores, por lo tanto si hay algún error en los documentos que interpreta, lo visualiza de la manera en que no lo ha entendido. (p. 59)

Esto indica que HTML es interpretado por los navegadores como Chrome, Edge, Firefox o Safari para leer e interpretar el contenido de una página web a través de la estructura de etiquetas con la cual fue construida, para poder visualizarla en el navegador. Esto es definido por Tabarés (2012), “el lenguaje HTML se basa principalmente en un sistema de etiquetas que indica



al navegador dónde está el cuerpo de un documento cuándo hay que colorear un texto, etc.” (p. 59).

### **2.1.2 CSS**

Al igual que la estructura o el esqueleto de un sitio web construido con HTML, es muy importante contar con una presentación y un diseño visual en los sitios web. CSS es una tecnología web orientada a estos dos aspectos, tal como lo define Álvarez et al. (2017):

CSS es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas.

Por tanto, podemos entender a CSS como una tecnología que permite trabajar sobre el contenido de una página web, más no sobre la forma ya que esto lo hace HTML. CSS por tanto se centra en la apariencia de los contenidos en cualquier tipo de sitio web.

### **2.1.3 JavaScript**

Para el desarrollo de las aplicaciones web, JavaScript es una tecnología web muy popular que inicialmente se enfocó en brindar interacción a los diferentes recursos de una página web, añadiendo un comportamiento más dinámico e interactivo. Según Eguíluz (2012):

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario (p. 5)

Es decir que JavaScript es una tecnología que aporta efectos y animaciones que aportan una experiencia de usuario más enriquecedora gráficamente.

## **2.2 Mockups en diseño de interfaces**

En el diseño de interfaces orientados a la presentación de los componentes gráficos de una página web, hay dos conceptos de gran importancia: los Wireframes y los Mockups. Según Nikiforova et al. (2024), “los mockups son representaciones gráficas de alta fidelidad del producto final, que detallan componentes y elementos con colores, pero carecen de interactividad” (p. 102). En este sentido, se pueden comprender como una herramienta que brinda una representación más detallada y realista de la interfaz de usuario, ya que presenta componentes gráficos con color, tipografías, imágenes y elementos visuales.

## **2.3 Diseño centrado en el usuario (UX)**

Las páginas web además de ser productos tecnológicos desarrollados con diferentes lenguajes y tecnologías web o además incluso de ser una solución específica a un problema dado, también se deben considerar como productos que resuelven necesidades del usuario y que se centran especialmente en su contexto, necesidades e intereses. Es por ello que, la Experiencia de Usuario ha sido un campo que ha cobrado importancia en las últimas décadas. Según Galarza (2016), “el UX tiene por objetivo la creación de productos que resuelvan necesidades concretas de sus usuarios, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo” (p. 27).

En este sentido, el uso de las tecnologías web como HTML, CSS y JavaScript deben enfocarse no solamente en el desarrollo técnico de las aplicaciones, sino también en la forma en que el usuario consume estos contenidos y estos, se adapten a sus necesidades puntuales.

### 3. Metodología

#### 3.1 Análisis de los requerimientos

Previo al diseño de los Mockups, se realizó un análisis de los requerimientos definidos con el objetivo de establecer claramente las funcionalidades y características que debe cumplir la aplicación web de la empresa Kibu. Este proceso permite asegurar que el producto final responda a las necesidades del negocio y a las expectativas de los usuarios. El análisis se realizó en dos categorías principales: Requerimientos funcionales (Relacionados con lo que el sistema debe hacer y Requerimientos no funcionales (condiciones de calidad y restricciones del sistema).

##### 3.1.1 *Requerimientos funcionales*

Los requerimientos funcionales definen las acciones y los procesos principales que debe ofrecer la aplicación:

**Tabla 2**

*Listado de requerimientos funcionales*

| Código | Requerimiento funcional        | Descripción   |
|--------|--------------------------------|---|
| RF-01  | Página de inicio               | La aplicación debe mostrar un <i>home</i> con slider de imágenes, nombre de la empresa, descripción, barra de navegación y pie de página.         |
| RF-02  | Listado de servicios           | Se debe presentar un catálogo con al menos 10 servicios, incluyendo imagen, nombre y precio.  |
| RF-03  | Vista de detalle de servicio   | Al seleccionar un servicio, se debe mostrar información ampliada: imagen, nombre, precio, cantidad disponible, descripción y estado de promoción. |
| RF-04  | Autenticación de administrador | Debe existir una ventana de inicio de sesión exclusiva para el administrador.   |

|       |                                |  |
|-------|--------------------------------|--|
| RF-05 | Gestión de servicios (CRUD)    | El administrador podrá crear, leer, actualizar y eliminar los servicios ofrecidos.                                     |
| RF-07 | Persistencia de la información | Se debe almacenar información de los servicios en un archivo en formato .json y mostrados localmente por el navegador. |

Nota. Datos tomados del documento guía de la actividad.

### 3.1.2 *Requerimientos no funcionales*

Los requerimientos no funcionales garantizan la calidad, el desempeño y la usabilidad de la aplicación

**Tabla 3**

*Listado de requerimientos no funcionales*

| <b>Código</b> | <b>Requerimiento no funcional</b> | <b>Descripción</b>  |
|---------------|-----------------------------------|---|
| RNF-01        | Usabilidad                        | La interfaz debe ser intuitiva, clara y fácil de usar para los usuarios finales y administradores.          |
| RNF-02        | Accesibilidad                     | La aplicación debe ser accesible desde computadores.  |
| RNF-04        | Mantenibilidad                    | El código y la base de datos deben estar organizados de forma modular, facilitando actualizaciones futuras. |
| RNF-05        | Escalabilidad                     | La aplicación debe estar diseñada para permitir la integración de nuevos servicios en el futuro.            |

Nota. Datos elaborados por el autor del documento.

A partir los requerimientos, se identifican los dos perfiles de usuario principales y sus interacciones con la aplicación:

- **Usuario final (Cliente):** Puede navegar en la página de inicio, visualizar los servicios disponibles y consultar los detalles de cada uno.
- **Administrador:** Puede autenticarse en el sistema y, mediante el panel de control, realizar operaciones CRUD sobre los servicios, además de gestionar la información básica de la empresa.

Este análisis de requerimientos constituye una base para la construcción de los Mockups, asegurando que el diseño visual corresponda con las funcionalidades esperadas.

## **3.2 Diseño de la Solución**

### ***3.2.1 Desarrollo de los MockUps.***

Con el propósito de dar cumplimiento a los requerimientos planteados para la aplicación web de Kibu, se desarrollaron cinco mockups empleando la herramienta **Figma**. Esta plataforma de diseño colaborativo permitió crear representaciones visuales de las interfaces, manteniendo consistencia en los estilos, tipografías y colores definidos para la identidad corporativa de la empresa. Los mockups diseñados se encuentran al final del presente documento.

A continuación, se describen en detalle los cinco mockups elaborados, indicando los elementos que los componen y la funcionalidad esperada en la aplicación final.

#### **3.2.1.1 Mockup 1. Home de la empresa**

Este diseño corresponde a la página principal y cumple la función de presentar la identidad de la compañía.

- **Funcionalidad:** Permite la primera interacción con el usuario, facilitando la navegación hacia las secciones clave y transmitiendo confianza mediante una estructura visual clara.

- **Elementos integrados:**

- Barra de navegación con enlaces a Home, Servicios, Contacto y Login.
- Slider con imágenes representativas.
- Nombre y descripción de la empresa.
- Sección de presentación rápida de servicios destacados.
- Footer con información de contacto.

### 3.2.1.2 Mockup 2. Catálogo de servicios

Corresponde a la vista en la que se listan los diez servicios ofrecidos por la empresa.

- **Funcionalidad:** Permite visualizar el catálogo de servicios de manera organizada y facilita el acceso a la información individual de cada uno mediante interacción directa.

- **Elementos integrados:**

- Título de sección, grilla de tarjetas que incluyen imagen, nombre y precio de cada servicio.
- Botón de “Ver Más” para acceder al detalle.

### 3.2.1.3 Mockup 3. Detalle del servicio

Diseño destinado a profundizar en la información de un servicio específico.

- **Funcionalidad:** Posibilita que el usuario consulte características completas del servicio y reconozca fácilmente promociones o disponibilidad.

- **Elementos integrados:**

- Imagen principal, nombre, precio, descripción detallada y etiqueta de si el servicio está en promoción.

- Botón para regresar al catálogo.

#### 3.2.1.4 Mockup 4. Login del administrador

Vista enfocada en el acceso al sistema por parte del administrador.

- **Funcionalidad:** Habilita el inicio de sesión seguro del administrador, quien posteriormente tendrá acceso a funcionalidades de gestión.
- **Elementos integrados:**
  - Formulario centrado con campos de usuario y contraseña.
  - Botón “Ingresar”.
  - Enlace de recuperación de contraseña
  - Imagen representativa.

#### 3.2.1.5 Mockup 5. Panel de administrador (CRUD de servicios)

Corresponde al área de gestión exclusiva del administrador.

- **Funcionalidad:** Permite la gestión completa del catálogo de servicios, aplicando las operaciones CRUD (Crear, leer, actualizar y eliminar). Este panel constituye el núcleo del área administrativa de la aplicación.
- **Elementos integrados:**
  - Barra de navegación con enlaces a Home, Servicios, Contacto y Login.
  - Tabla central con los campos Nombre, Precio, Promoción, Porcentaje de descuento, Descripción, Información adicional, características.
  - Botones de acción (Crear, editar y eliminar servicios).

En conjunto, los mockups representan el flujo de interacción principal del sistema, abarcando tanto la experiencia del cliente (Home, Catálogo, Servicio) como la del administrador

(Login y Panel de Control). Esta propuesta servirá de base para la fase de codificación en HTML, CSS y JavaScript.

Es importante comentar que debido al tamaño de los mockups, no fue conveniente insertarlos como imágenes dentro del cuerpo de esta sección. Las imágenes del Mockup se encuentran disponibles en la carpeta *Docs*, del repositorio de la aplicación.

### **3.3 Implementación del Frontend**

#### ***3.3.1 Configuración del entorno***

La configuración del entorno de desarrollo para el desarrollo del Frontend requiere la instalación y configuración de herramientas específicas que garanticen un flujo de trabajo eficiente y consistente. La configuración esencial requiere:

- **Node.js y npm:** Se requiere Node.js versión 18.0 o superior el cual incluye npm como gestor de paquetes predeterminado. Esta versión garantiza compatibilidad con las características modernas de JavaScript y TypeScript usadas en el proyecto.
- **Git para Control de Versiones:** El sistema de control de versiones Git se configura para mantener un historial completo de cambios y facilitar la colaboración en equipo. La configuración incluye la instalación del programa, la configuración de las credenciales y la configuración de la rama principal.
- **Editor de código:** Para este proyecto se utilizó Cursor como entorno de desarrollo integrado, configurado con extensiones específicas para React, TypeScript y Tailwind CSS que proporcionan autocompletado, verificación de sintaxis y formateo automático.



- **Inicialización con Vite:** El proyecto se inicializa usando Vite como herramienta de construcción, seleccionando la plantilla React con TypeScript para aprovechar las ventajas del tipado estático.
- **Instalación de dependencias principales:** Las dependencias se organizan en dos categorías: dependencias de producción necesarias para el funcionamiento de la aplicación, y dependencias de desarrollo usadas durante el proceso de construcción.
- **Configuración de Tailwind CSS:** La configuración de Tailwind CSS se personaliza para reflejar la identidad visual de Kibu, incluyendo colores corporativos, tipografía específica y breakpoints responsivos.

### ***3.3.2 Frontend: Explicación de las páginas desarrolladas***

#### **3.3.2.1 Página principal (Home)**

La página principal constituye el punto de entrada principal de la aplicación, diseñada para proporcionar una primera impresión sólida de la empresa y sus servicios. Esta página define una estructura que presenta la propuesta de valor de la empresa de manera inmediata y atractiva. La sección principal incluye un título llamativo "Soluciona con Kibu", acompañado de una descripción que explica los servicios de desarrollo de software personalizado. Se incorpora un carrusel de imágenes interactivo que muestra diferentes aspectos de la empresa, incluyendo el equipo de trabajo, tecnologías utilizadas y proyectos exitosos. Este carrusel implementa navegación manual mediante flechas direccionales y reproducción automática cada 4 segundos.

Los elementos mencionados en el desarrollo de los mockups, se encuentra integrados.

**Figura 1**

Página Home de la aplicación



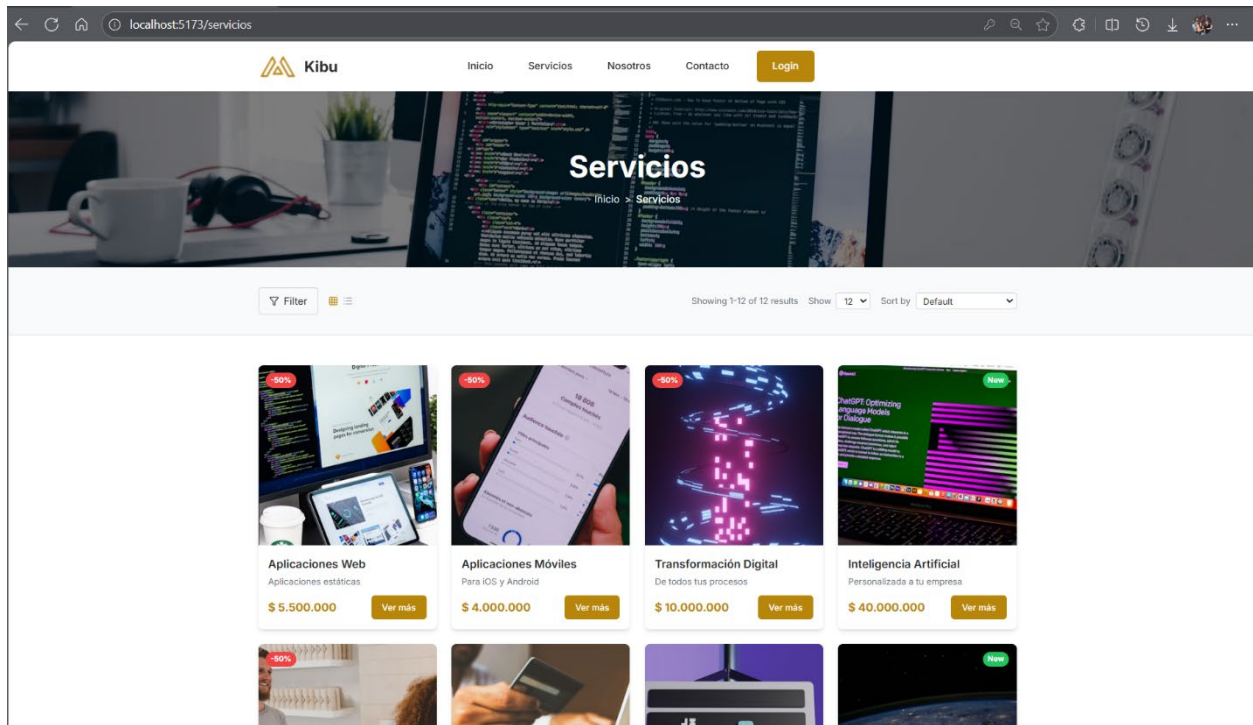
*Nota:* Imagen elaborada por el autor del documento.

### 3.3.2.2 Catálogo de Servicios

La página de catálogo de servicios implementa una vista comprensiva de todos los servicios ofrecidos, organizados en una estructura de cuadrícula responsiva que se adapta automáticamente según el tamaño de pantalla. El sistema de paginación divide los resultados en páginas manejables, mostrando 12 servicios por página por defecto, con opciones para modificar esta cantidad. Cada servicio se presenta con indicadores visuales distintivos: etiquetas de promoción en color rojo para descuentos activos, etiquetas verdes para servicios nuevos, y precios claramente destacados. La información incluye título, descripción breve, categoría, precio y botón de acción para acceder al detalle completo.

**Figura 2**

## Página de Servicios



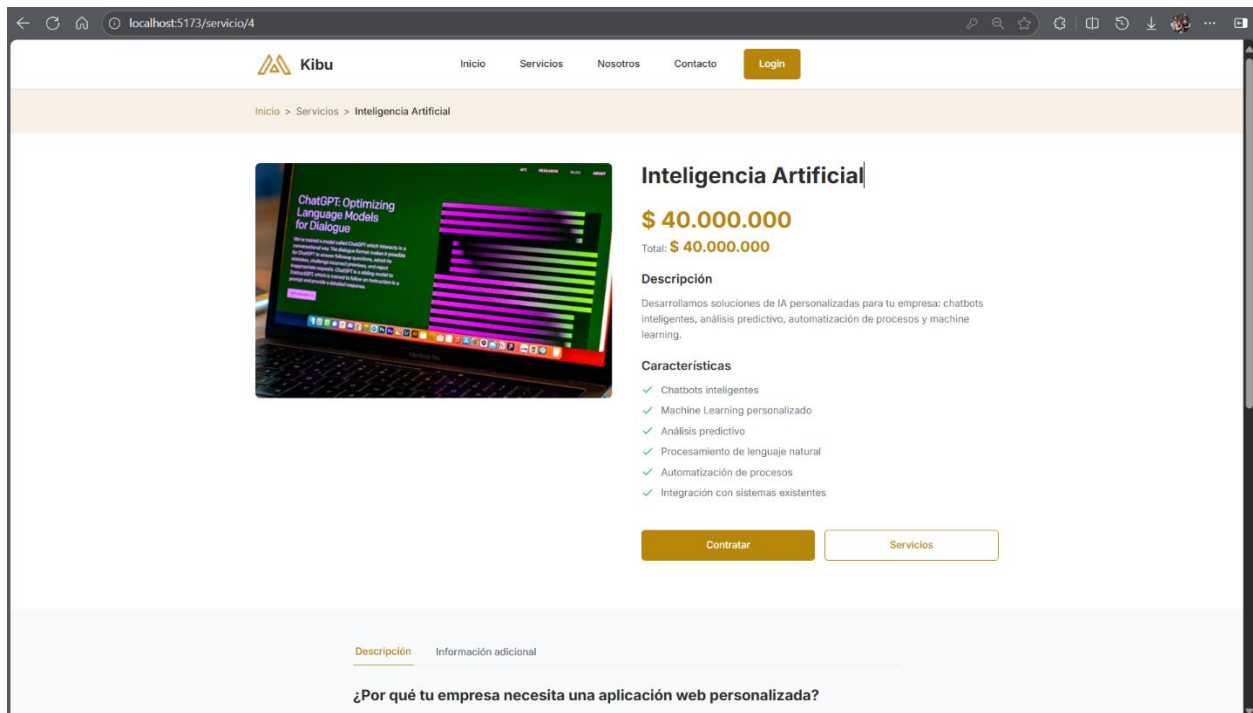
*Nota:* Imagen elaborada por el autor del documento.

### 3.3.2.3 Detalle del Servicio

La página de detalle de un servicio, proporciona información exhaustiva sobre servicios individuales, implementando un diseño de dos columnas que optimiza la presentación de contenido visual y textual. La columna izquierda presenta una galería de imágenes con imagen principal y miniaturas adicionales, mientras la columna derecha contiene toda la información descriptiva. Se incluyen características técnicas listadas con iconos de verificación, información adicional expandida, y botones de acción prominentes para contratar el servicio o regresar al catálogo.

**Figura 3**

Página de detalle de un servicio



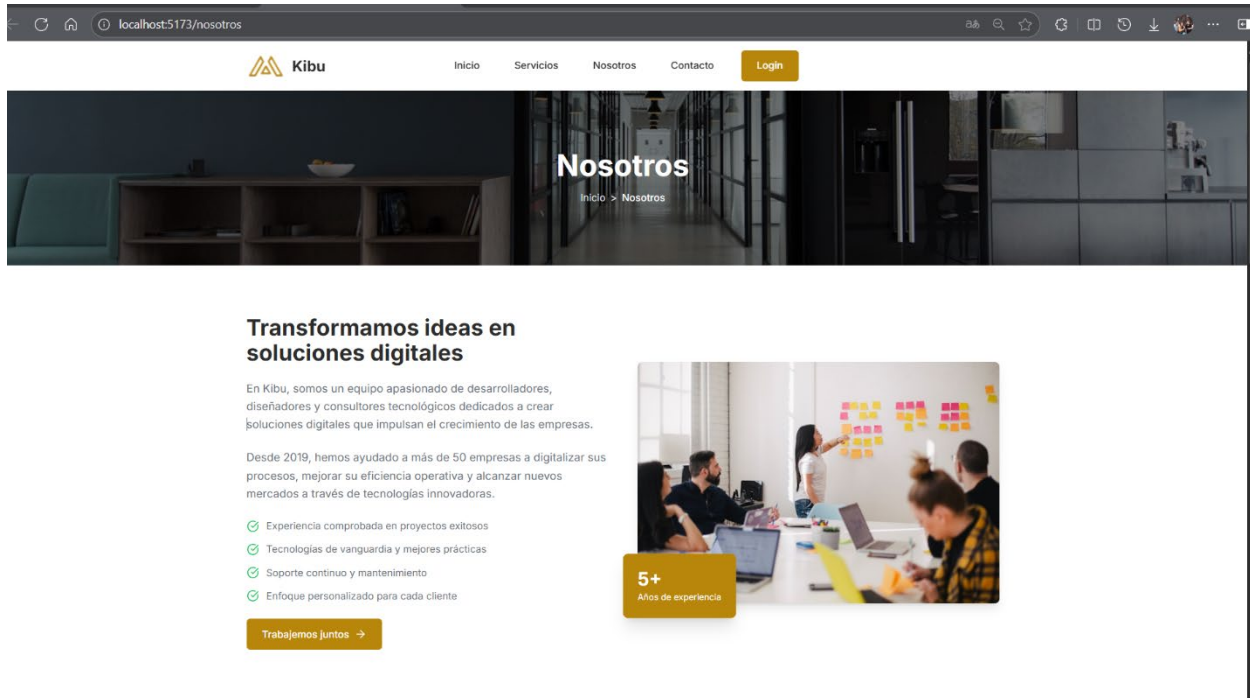
*Nota:* Imagen elaborada por el autor del documento

#### 3.3.2.4 Página Nosotros (Bonus)

Esta página institucional presenta la identidad corporativa de Kibu mediante una estructura narrativa que incluye historia, valores, estadísticas y equipo de trabajo. La sección principal implementa un diseño de dos columnas con contenido textual y elemento visual destacado. Las estadísticas se presentan en formato de métricas destacadas (500+ proyectos, 50+ clientes, 5+ años de experiencia, soporte 24/7), utilizando tipografía de gran tamaño y colores corporativos para maximizar el impacto visual. La sección de valores corporativos emplea iconografía consistente y descripciones concisas para comunicar los principios fundamentales de la empresa. El directorio del equipo presenta tarjetas individuales para cada miembro, incluyendo fotografía profesional, nombre, cargo y descripción breve de experiencia. La sección concluye con un call-to-action prominente que dirige hacia la página de contacto.

**Figura 4**

Página "Nosotros"



*Nota:* Imagen elaborada por el autor del documento

### 3.3.2.5 Página de Contacto (Bonus)

La página de contacto implementa un formulario funcional de múltiples campos con validación en tiempo real y manejo de estados de carga. El formulario incluye campos obligatorios (nombre, email, teléfono, servicio de interés, mensaje) y opcionales (empresa), con validación específica para cada tipo de dato. Se incluye una sección de información de contacto complementaria que presenta dirección física, números telefónicos, direcciones de correo electrónico y horarios de atención. Se proporciona información detallada sobre horarios de atención organizada por días de la semana.

El sistema de confirmación presenta una pantalla de éxito tras el envío exitoso del formulario, incluyendo mensajes de agradecimiento y tiempo estimado de respuesta.

**Figura 5**

## Página de Contacto

localhost5173/contacto

Kibu Inicio Servicios Nosotros Contacto Login

### Contacto

Inicio > Contacto

#### Envíanos un mensaje

Nombre completo \*

Tu nombre completo

Email \*

tu@email.com

Teléfono \*

+57 300 123 4567

Empresa (opcional)

Nombre de tu empresa

Servicio de interés \*

Selecciona un servicio

Mensaje \*

Cuéntanos más sobre tu proyecto...

#### Información de contacto

Estamos aquí para ayudarte. Contáctanos a través de cualquiera de estos medios y te responderemos lo antes posible.

**Dirección**  
Calle 61A#9-15 Bogotá D.C.  
ZIP 11110 33134 COL

**Teléfono**  
+57 699 67 83  
Lun - Vie: 8:00 AM - 6:00 PM

**Email**  
servicios@kibu.com.co  
info@kibu.com.co

**Horario de atención**  
Lunes - Viernes: 8:00 AM - 6:00 PM  
Sábados: 9:00 AM - 2:00 PM

**Horario de atención**  
Lunes - Viernes: 8:00 AM - 6:00 PM

*Nota:* Imagen elaborada por el autor del documento.

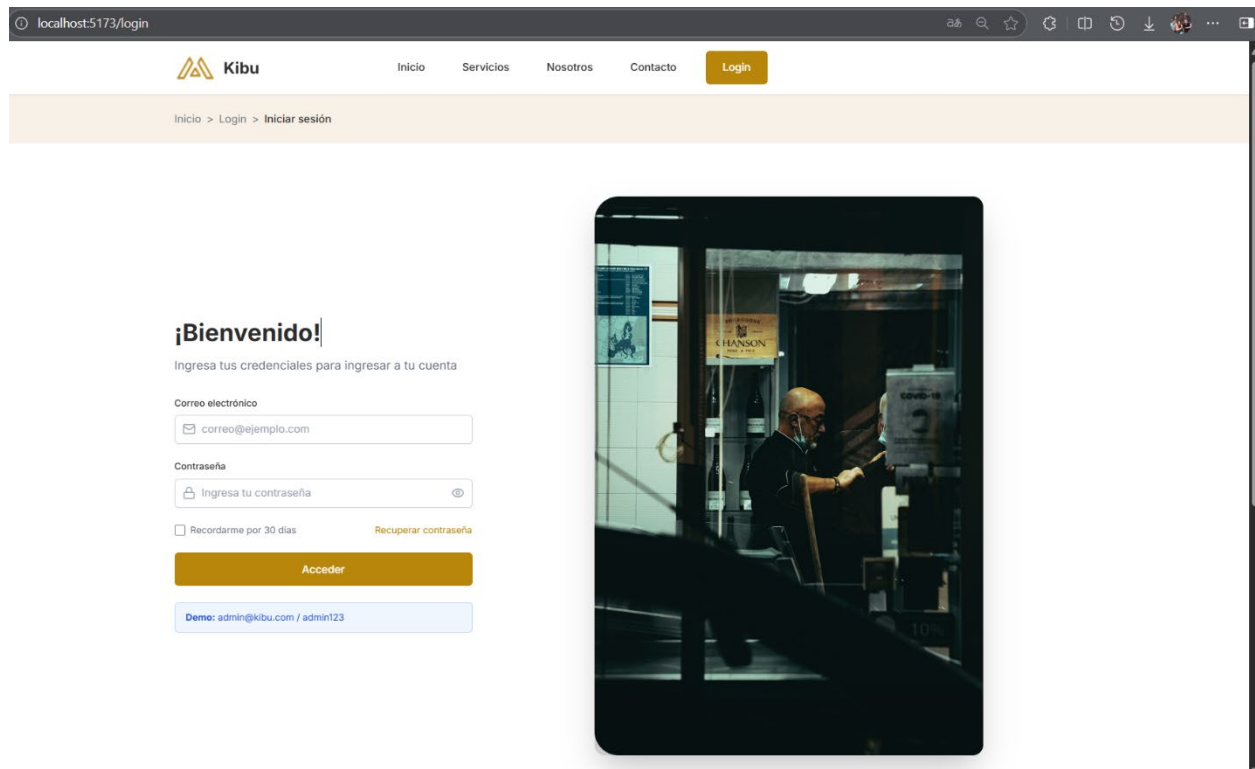
### 3.3.2.6 Login de administrador

La página de login implementa un formulario seguro de autenticación administrativa con campos para correo electrónico y contraseña. Se incluyen funcionalidades de mostrar/ocultar contraseña, opción de "recordarme", y enlaces para recuperación de contraseña.

El sistema incluye validación tanto del lado cliente como simulación de validación del servidor, con manejo apropiado de errores y estados de carga. Para efectos de demostración, se proporcionan credenciales de prueba (**admin@kibu.com** / **admin123**) claramente indicadas en la interfaz.

**Figura 6**

Página de Loggin de Administrador



*Nota:* Imagen elaborada por el autor del documento.

### 3.3.2.7 Panel de Administración

El panel administrativo proporciona funcionalidades completas de gestión de servicios (CRUD - Crear, Leer, Actualizar, Eliminar) mediante una interfaz tabular responsiva. La tabla principal presenta todos los servicios con columnas para nombre, precio, estado de promoción, porcentaje de descuento, descripción, información adicional y características técnicas. Se incluyen controles de búsqueda en tiempo real, opciones de filtrado, y botones de acción individuales para cada servicio (editar, eliminar). El sistema implementa modales de confirmación para acciones destructivas y formularios emergentes para creación y edición de servicios. La navegación del panel mantiene coherencia con el resto del sitio, incluyendo enlaces principales y funcionalidad de cierre de sesión.

**Figura 7**

## Página de Administración CRUD de Servicios

The screenshot displays the Kibu administration interface. At the top, there's a navigation bar with links for 'Inicio', 'Servicios', 'Nosotros', 'Contacto', and a 'Login' button. Below this, the 'Panel de Administración' is visible, with a user profile 'Admin' and a 'Salir' link. The main content area features a large banner with the text 'Panel de Administración' and a breadcrumb 'Inicio > Admin'. Below the banner, there are two side-by-side forms. The left form, titled 'Agregar servicio', includes fields for 'Nombre del servicio', 'Precio', 'Promoción' (a dropdown menu set to 'No'), 'Porcentaje de descuento' (a text input with '0'), 'Descripción', and 'Características'. The right form, titled 'Seleccionar servicio', has a dropdown menu for 'Escoja un servicio' with 'Transformación Digital' selected. Below this are buttons for 'Editar', 'Eliminar', and 'Consultar'. The 'Transformación Digital' section shows pricing: 'Precio normal' at 'COP 20.000.000', 'Cuenta con promoción' at '50%', and a 'Total' of 'COP 10.000.000'. It also includes a 'Descripción' and a 'Características' section with a list of bullet points: 'Análisis de procesos actuales', 'Diseño de arquitectura digital', 'Implementación por fases', 'Capacitación del personal', and 'Migración de datos'.

*Nota:* Imagen elaborada por el autor del documento.

### 3.3.2.8 Funcionalidades implementadas en páginas

- **Navegación y Routing:** El sistema de navegación implementa **React Router DOM** para gestión de rutas del lado del cliente (SPA), proporcionando transiciones fluidas entre páginas sin recargas completas del navegador. La barra de navegación permanece activa y consistente a través de todas las páginas. El Sistema de breadcrumbs se implementa dinámicamente en páginas de detalle y formularios proporcionando contexto de navegación y enlaces directos para retroceder en la jerarquía del sitio.
- **Interacción del usuario:** El carrusel de imágenes de la página principal permite interacción a través de navegación manual usando flechas, indicadores de



posición, reproducción automática y contador de total de imágenes. Los

**formularios implementan validación de campos** proporcionando

retroalimentación sobre errores de formato o campos requeridos. Se implementan

**efectos de *hover* en elementos interactivos**, animaciones de escala en botones y

transiciones suaves entre estados. Las tarjetas de servicio incluyen efectos de

elevación y escala de imagen al pasar el mouse.

- **Responsividad y adaptabilidad:** El diseño es responsivo para navegadores y usando un enfoque de *mobile-first* usando Tailwind CSS. La página se adapta automáticamente en una columna en móvil, dos en Tablet y lo correspondiente a escritorio.
- **Gestión de estado:** La aplicación utiliza React Hooks para gestión de estado local en componentes individuales. El estado de autenticación se maneja mediante localStorage para persistencia entre sesiones, con verificación automática al cargar componentes protegidos.

### 3.3.2.9 Uso de tecnologías Frontend

- **HTML:** La estructura HTML se implementa a través de JSX (JavaScript XML), una extensión de sintaxis que permite escribir elementos HTML dentro de código JavaScript/TypeScript. Esta aproximación proporciona ventajas de HTML tradicional con la potencia de JavaScript moderno.
- **CSS:** La estilización se implementa mediante Tailwind CSS, un framework de CSS utility-first que proporciona clases predefinidas para construcción rápida de interfaces. La configuración personalizada se define en tailwind.config.js.

- **JavaScript/TypeScript:** La lógica de la aplicación se desarrolla en TypeScript, proporcionando tipado estático y detección temprana de errores. La implementación abarca múltiples patrones y funcionalidades como la Gestión de Estado con Hooks, Validación de Formularios, Tipado Estricto y Manejo de Eventos y Efectos.
- **Frameworks y librerías:** Se usó React como Framework principal para la construcción de interfaz de usuario mediante componentes reutilizables y gestión de estado declarativa y se implementan patrones modernos. En cuanto a React Router DOM, la librería para gestión de rutas del lado del cliente, implementando navegación SPA sin recargas de página.

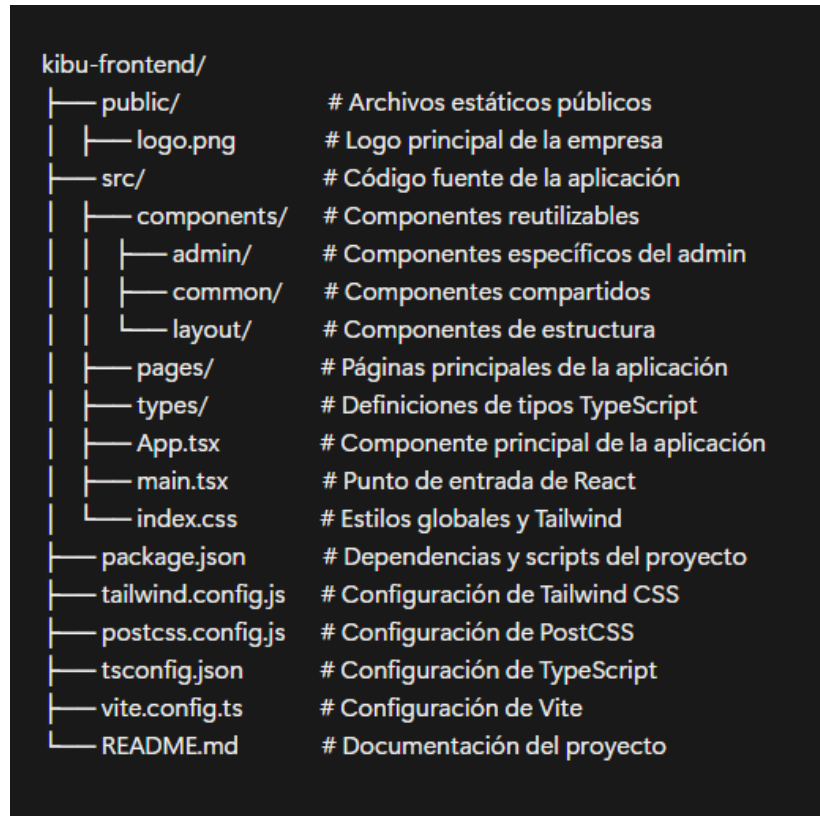
En cuanto a Tailwind CSS, se usa este framework CSS utility-first para diseño rápido y consistente, el cual proporciona un sistema de diseño coherente, Responsividad integrada, personalización extensa y optimización automática de CSS. Otros componentes usados con Lucide React para la iconografía, TypeScript que añade tipado estática.

### **3.3.2.10 Organización del Directorio Raíz**

El Frontend se organiza siguiendo prácticas recomendadas de Reactor y Desarrollo Web moderno, implementando una arquitectura escalable y mantenible La estructura del directorio raíz, se detalla en la siguiente figura:

**Figura 8**

Directorio Raíz de la aplicación



*Nota:* Imagen elaborada por el autor del documento.

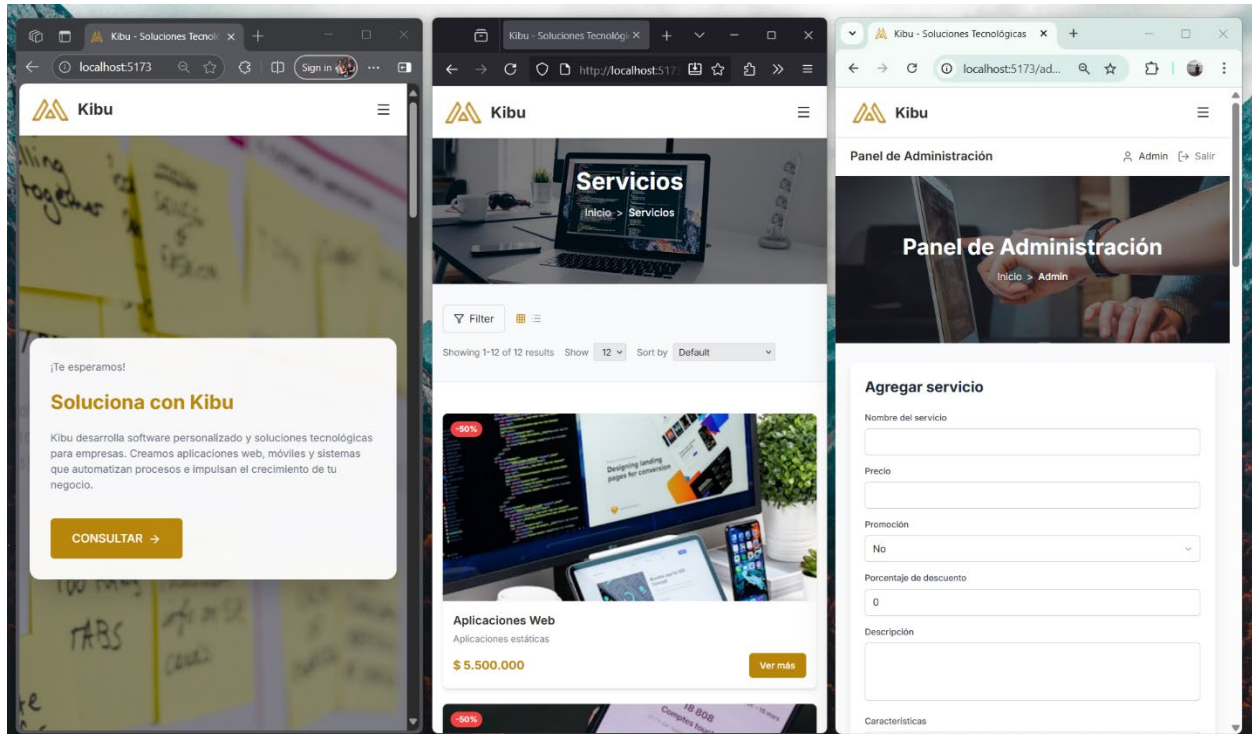
### 3.3.2.11 Prueba de visualización en navegadores

Como parte del desarrollo del frontend de la aplicación, esta fue ejecutada en los navegadores de Microsoft Edge, Google Chrome y Mozilla Firefox, para validar la compatibilidad de las páginas y los componentes. Durante las pruebas se verificó la correcta carga de estilos, la disposición de los elementos en la interfaz y la funcionalidad de los formularios. Los resultados evidenciaron que la aplicación mantiene un comportamiento consistente en los tres navegadores evaluados, sin presentar fallos de renderizado ni diferencias significativas en el diseño o en la experiencia de usuario. De esta manera, se garantiza que el

sistema es accesible y funcional desde los principales navegadores utilizados actualmente. Esto se puede apreciar en la siguiente figura, de izquierda a derecha se validó en Microsoft Edge, Mozilla Firefox y Google Chrome:

**Figura 9**

Ejecución de la aplicación en navegadores



*Nota:* Imagen elaborada por el autor del documento.

### 3.4 Repositorio y Código Fuente

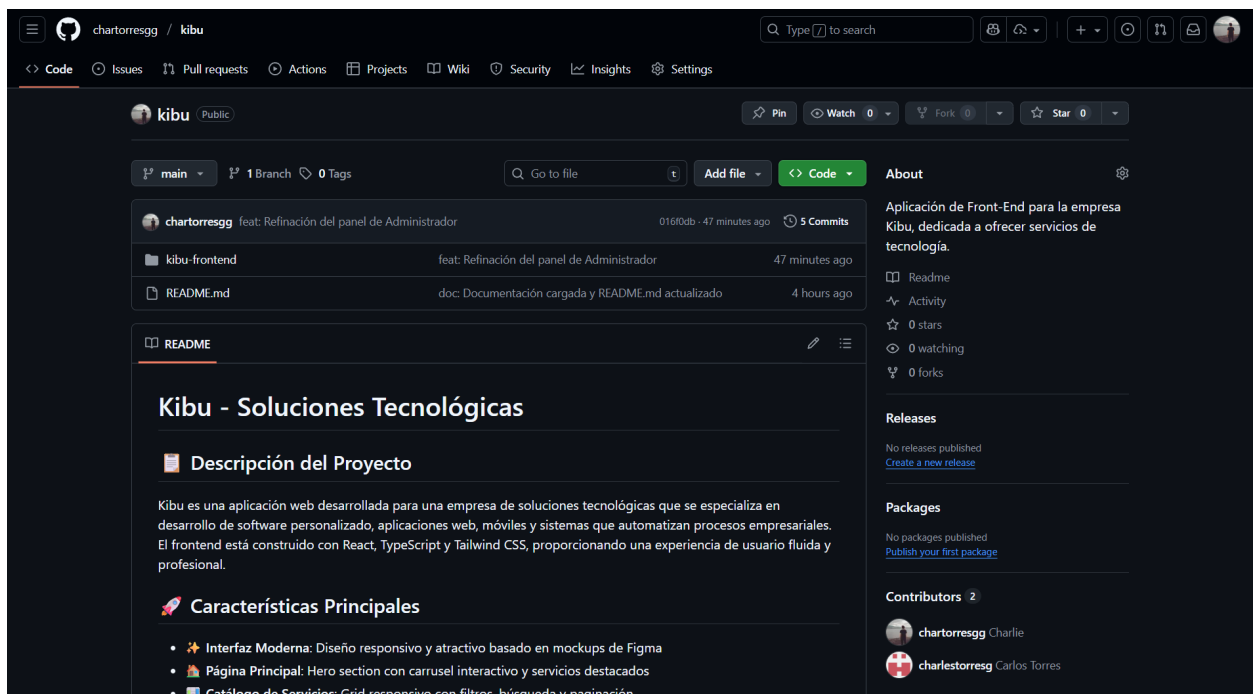
El código fuente completo del proyecto se encuentra alojado en el repositorio público de GitHub, al cual se puede acceder mediante la URL: <https://github.com/chartorresgg/kibu>. El repositorio fue implementado siguiendo una organización estándar con las siguientes características:

- **Rama principal (*main*):** Contiene la versión estable y desplegable del proyecto.

- **README.md:** Brinda una presentación general de la aplicación con las características principales, tecnologías usadas, estructura, páginas y funcionalidades, instalación e información adicional.
- **Gitignore:** Archivo de configuración para excluir archivos temporales, dependencias y builds.
- **Commits descriptivos:** Todas las confirmaciones de cambios incluyen mensajes claros que describen los cambios implementados.

**Figura 10**

Repositorio GitHub de la aplicación



*Nota:* Imagen elaborada por el autor del documento.

### 3.5 Persistencia de información con localStorage

#### 3.5.1 Implementación del Sistema de Persistencia Local

Para la simulación de una base de datos en el frontend, se implementó un sistema de persistencia basado en localStorage del navegador web. Esta solución permite mantener la funcionalidad completa de la aplicación sin requerir un servidor backend, cumpliendo con los objetivos académicos del proyecto mientras proporciona una experiencia de usuario realista.

#### Figura 11

Información predefinida de servicios en dataService.js

```
// src/services/dataService.ts
import type { Service, ServiceFormData } from '../types/service';

// Simulación de base de datos en memoria
let servicesData: Service[] = [
  {
    id: 1,
    nombre: "Aplicaciones Web",
    precio: "5500000",
    promocion: true,
    porcentajeDescuento: 50,
    descripcion: "Creamos aplicaciones web a medida que se adaptan perfectamente a las necesidades de tu empresa",
    informacionAdicional: "En un mundo donde el 86% de los clientes investigan online antes de comprar, tener",
    características: "Diseño UX/UI personalizado, Desarrollo responsivo (móvil y desktop), Panel de administración",
    imagen: "https://images.unsplash.com/photo-1547658719-da2b51169166?w=600&h=400&fit=crop&crop=center",
    categoria: "web",
    disponible: 5
  },
  {
    id: 2,
    nombre: "Apps Móviles",
    precio: "4000000",
    promocion: true,
    porcentajeDescuento: 50,
    descripcion: "Desarrollamos aplicaciones móviles nativas y multiplataforma para iOS y Android, con interfaz
```

*Nota:* Información predefinida para la visualización de los Servicios en la aplicación.

Imagen elaborada por el autor del documento.

El desarrollo de esta funcionalidad inició con la creación del archivo dataService.ts ubicado en src/services/, que actúa como capa de abstracción entre los componentes React y el almacenamiento local.

### 3.5.2 Arquitectura de Almacenamiento de Datos

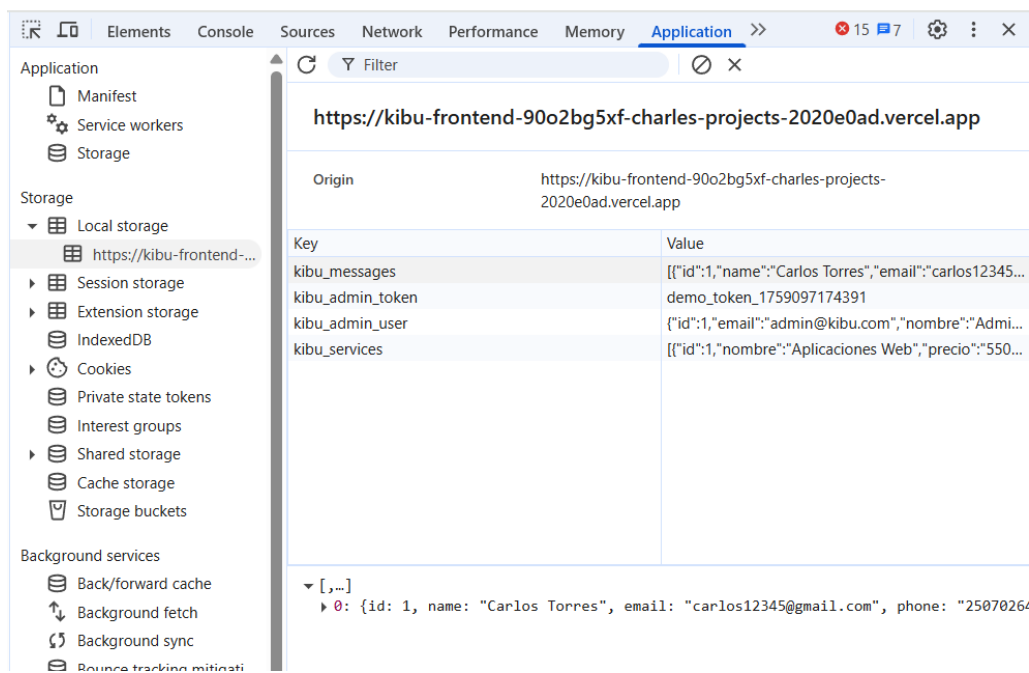
La aplicación Kibu utiliza el API localStorage nativo del navegador para almacenar datos en formato JSON. Se definieron cuatro claves principales de almacenamiento:

- **kibu\_services:** Array de servicios con información completa incluyendo id, nombre, precio, descripción, características, promociones y disponibilidad.
- **kibu\_messages:** Mensajes de contacto enviado por usuarios con campos de nombre, email, teléfono, empresa, servicio de interés y mensaje
- **kibu\_admin\_token:** Token de autenticación administrativa.
- **kibu\_admin\_user:** Datos del usuario administrador activo.

Estas cuatro claves se pueden observar en la siguiente figura:

**Figura 12**

Claves usadas por el navegador



*Nota:* Estas claves evidencian la persistencia de la información enviada y almacenada por navegador. Imagen elaborada por el autor del documento.

### 3.5.3 Estructura de Datos implementada

Los servicios se almacenan siguiendo una estructura TypeScript definida que garantiza la consistencia de tipos de datos:

**Figura 13**

Datos solicitados para un servicio

```
export interface ServiceData {  
  id?: number;  
  nombre: string;  
  precio: string;  
  promocion: boolean;  
  porcentajeDescuento: number;  
  descripcion: string;  
  informacionAdicional: string;  
  características: string;  
  imagen: string;  
  categoria: string;  
  disponible: number;  
}
```

*Nota:* Datos alojados en la clase src/types/service.ts. Imagen elaborada por el autor del documento.

Los mensajes de contacto enviados por los usuarios, siguen una estructura similar que incluyen metadatos de fecha de envío y estado de lectura para gestión administrativa.

### 3.5.4 Simulación de Operaciones CRUD

El archivo dataService.js implementa funciones asíncronas que simulan llamadas a una API real, incluyendo delays artificiales para replicar la latencia de red. Las operaciones implementadas incluyen:



- **Create:** La función `createService()` genera IDs únicos mediante `Math.max()` sobre los IDs existentes, valida los datos de entrada, construye el objeto completo del servicio y persiste inmediatamente en `localStorage`. Se implementa un delay de 800ms para simular tiempo de procesamiento del servidor.
- **Read:** Las funciones `getAllServices()` y `getServiceById()` recuperan datos desde `localStorage` con delays de 500ms y 300ms respectivamente. Se implementa clonación profunda de arrays para evitar mutaciones accidentales del estado.
- **Update:** La función `updateService()` localiza el servicio por ID, valida los nuevos datos, actualiza el registro y persiste los cambios. Incluye manejo de errores para servicios no encontrados y validación de integridad de datos.
- **Delete:** La función `deleteService()` implementa eliminación segura con verificación de existencia previa, actualización del array de servicios y persistencia inmediata de los cambios.

### 3.5.5 *Funcionalidades avanzadas implementadas*

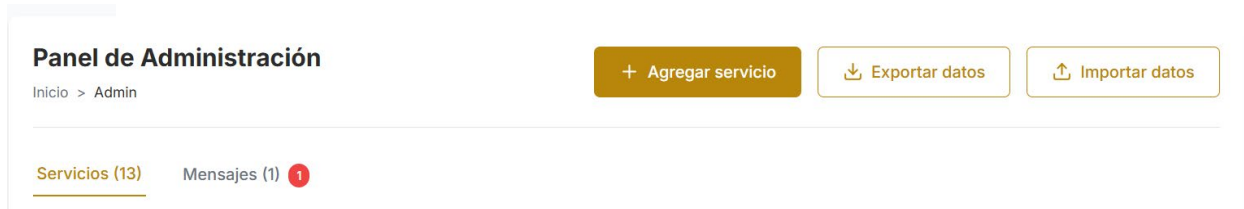
Se desarrollaron funcionalidades adicionales que enriquecen la experiencia de gestión:

- **Inicialización de datos:** La función `initializeData()` carga datos predefinidos si `localStorage` está vacío, garantizando que la aplicación tenga contenido inicial para demostración. Se incluyen 12 servicios precargados con información completa.
- **Backup y restauración:** Las funciones `exportData()` e `importData()` permiten exportar todos los datos en formato JSON e importar desde archivos externos. Esta funcionalidad facilita la transferencia de datos entre navegadores e información de respaldo.

- **Gestión de mensajes:** Se implementó un sistema para manejo de mensajes de contacto, incluyendo persistencia automática, marcado de lectura, y visualización administrativa.

### Figura 14

Funcionalidad para importar o exportar servicios



*Nota:* Imagen elaborada por el autor del documento.

#### 3.5.6 Manejo de Estados y Validación

La aplicación incluye un manejo de estados de carga y error. Todas las operaciones asíncronas proporcionan feedback visual mediante estados de carga, y se implementan validaciones tanto del lado del cliente como simulaciones de validación del servidor. La validación incluye verificación de tipos de datos, campos requeridos, teléfono, email. Los errores se capturan y se presentan de manera amigable al usuario.

#### 3.5.7 Ventajas y Limitaciones

Esta implementación ofrece ventajas para el desarrollo de la aplicación, en este contexto académico: no requiere configuración de servidor, proporciona persistencia real durante la sesión de navegación, permite demostrar funcionalidades completas de gestión de datos, y facilita la evaluación sin dependencias externas.

Las limitaciones incluyen ámbito local por el navegador, capacidad limitada de almacenamiento, ausencia de sincronización entre dispositivos, y dependencia del

almacenamiento del cliente. Estas limitaciones son aceptables para el contexto académico del proyecto.

### **3.6 Despliegue en la Nube**

#### ***3.6.1 Selección de Plataforma de Hosting***

Para el despliegue de la aplicación Kibu en la Nube, se seleccionó Vercel como plataforma de hosting, debido a su integración con aplicaciones React, proceso de deploy automatizado desde GitHub, tier robusto que cumple con los requerimientos del proyecto, y soporte nativo para Single Page Applications (SPA).

Vercel proporciona características específicamente diseñadas para aplicaciones frontend modernas, incluyendo optimizaciones automáticas de performance, CDN global para distribución de contenido, y manejo inteligente de rutas para aplicaciones React Router.

#### ***3.6.2 Preparación del Proyecto para Deploy***

El proceso de preparación requirió la resolución de incompatibilidades en la configuración del proyecto. Inicialmente, el package.json contenía configuraciones mixtas entre Create React App y Vite, causando errores de Build. Se realizó una auditoría completa de la configuración:

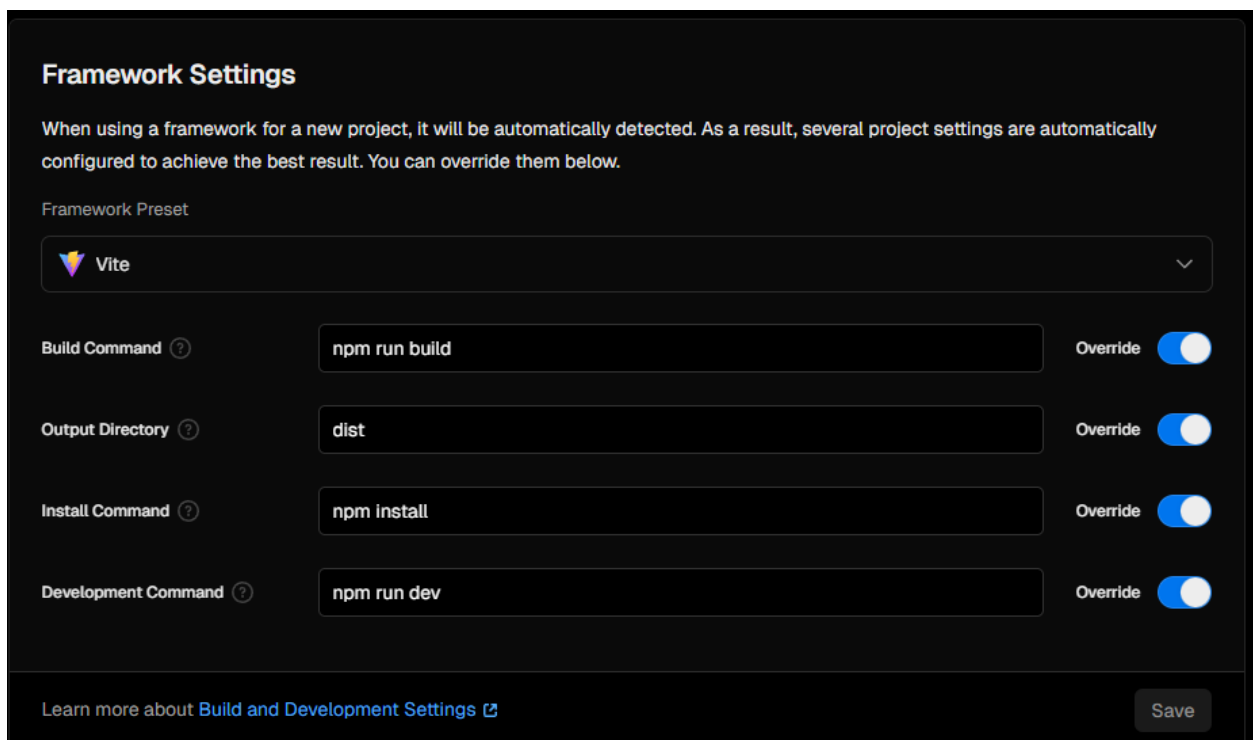
- **Actualización de package.json:** Se corrigieron los scripts de build para usar vite build en lugar de react-scripts build, se actualizó el output directory a dist (requerido por Vite), y se eliminaron dependencias conflictivas de Create React App.

- **Creación de vite.config.ts:** Se estableció la configuración específica para Vite incluyendo plugins de React, configuración de build con output directory correcto, y configuración de base path para deployment.
- **Configuración de vercel.json:** Se creó un archivo de configuración específico para Vercel que incluye especificación del framework como "vite", definición del build command como "npm run build", configuración del output directory como "dist", e implementación de rewrites para manejo de rutas de React Router.

### 3.6.3 Proceso de configuración de Vercel

**Figura 15**

Configuración interna del Deployment



The screenshot shows the 'Framework Settings' panel in Vercel. At the top, it says 'When using a framework for a new project, it will be automatically detected. As a result, several project settings are automatically configured to achieve the best result. You can override them below.' Below this, the 'Framework Preset' is set to 'Vite'. There are four rows of settings, each with a label, a value input field, and an 'Override' toggle switch. All toggle switches are currently turned on (blue). The settings are: Build Command (npm run build), Output Directory (dist), Install Command (npm install), and Development Command (npm run dev). At the bottom left, there is a link 'Learn more about Build and Development Settings' with an external link icon. At the bottom right, there is a 'Save' button.

| Setting             | Value         | Override |
|---------------------|---------------|----------|
| Build Command       | npm run build | On       |
| Output Directory    | dist          | On       |
| Install Command     | npm install   | On       |
| Development Command | npm run dev   | On       |

*Nota:* Datos alojados en la clase src/types/service.ts. Imagen elaborada por el autor del documento.

La configuración en el Dashboard de Vercel requirió ajustes específicos para garantizar el funcionamiento correcto:

- **Framework Detection:** Se configuró manualmente el framework preset como "Vite" para asegurar que Vercel use las optimizaciones correctas y los comandos de build apropiados.
- **Build Settings:** Se especificó el build command como `npm run build`, se configuró el output directory como `dist`, se estableció el install command como `npm install`, y se definió el development command como `npm run dev`.
- **Environment Variables:** Aunque la aplicación no requiere variables de entorno específicas, se configuró la estructura para futuras expansiones que puedan requerir configuraciones específicas de producción.

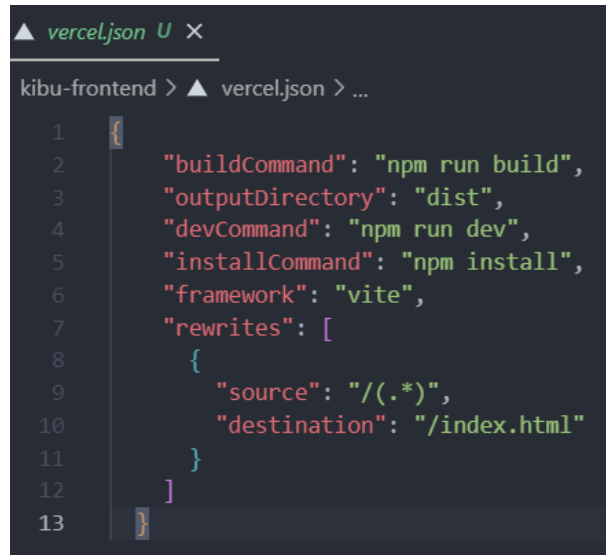
#### ***3.6.4 Configuración de Routing***

Una de las configuraciones más críticas fue el manejo correcto de rutas para Single Page Applications. React Router maneja las rutas del lado del cliente, pero el servidor necesita configuración específica para servir el archivo `index.html` para todas las rutas no reconocidas. Se implementó la configuración en `Vercel.json`.

Esta configuración garantiza que todas las rutas (`/`, `/servicios`, `/contacto`, `/login`, `/admin`) funcionen correctamente cuando se accede directamente desde el navegador, manteniendo la funcionalidad completa de navegación de la aplicación. Esta configuración mencionada se puede apreciar en la siguiente figura:

**Figura 16**

Configuración de vercel.json



```
▲ vercel.json U ×
kibu-frontend > ▲ vercel.json > ...
1  {
2    "buildCommand": "npm run build",
3    "outputDirectory": "dist",
4    "devCommand": "npm run dev",
5    "installCommand": "npm install",
6    "framework": "vite",
7    "rewrites": [
8      {
9        "source": "/(.*)",
10       "destination": "/index.html"
11     }
12   ]
13 }
```

*Nota:* Imagen elaborada por el autor del documento.

### 3.6.5 Automatización del Deployment

Se estableció una integración continua que conecta el repositorio de GitHub con Vercel, permitiendo deployments automáticos en cada push a la rama principal. Este flujo garantiza que la versión en línea esté siempre sincronizada con el código fuente más reciente. El proceso automatizado incluye:

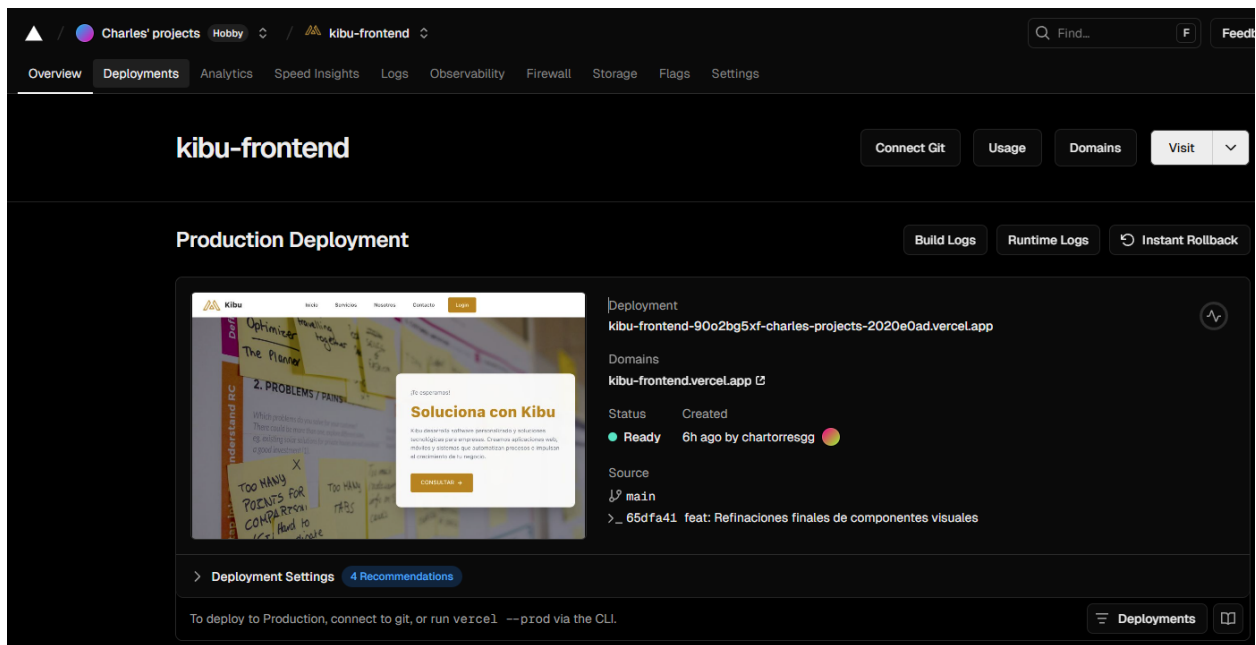
- **Triggers Automáticos:** Cada push a la rama main dispara automáticamente un nuevo deployment, permitiendo que los cambios en el código se reflejen inmediatamente en la aplicación en línea.
- **Preview Deployments:** Los pushes a ramas secundarias generan deployments de preview con URLs únicas, facilitando la revisión de cambios antes de mergear a la rama principal.
- **Build Logs:** Vercel proporciona logs detallados de cada proceso de build, permitiendo identificar y resolver problemas de deployment de manera eficiente.

- **Rollback Capabilities:** La plataforma mantiene un historial completo de deployments, permitiendo rollback inmediato a versiones anteriores.

El deployment de la aplicación y su configuración general, se puede apreciar en la siguiente imagen:

**Figura 17**

### Deployment en Vercel



*Nota:* Imagen elaborada por el autor del documento

### 3.6.6 Configuración de Dominio y Acceso

La aplicación se encuentra disponible públicamente en la URL: <https://kibu-frontend.vercel.app>. Esta URL proporciona:

- **Acceso Global:** Disponibilidad 24/7 desde cualquier ubicación con conexión a internet, sin restricciones geográficas o de autenticación para el contenido público.

- **HTTPS Automático:** Certificado SSL/TLS automático y renovación automática, garantizando conexiones seguras para todos los usuarios.
- **Performance Optimizada:** Tiempo de carga optimizado a través de la red global de CDN y técnicas de optimización automática.

La configuración permite que cualquier persona pueda acceder y evaluar la aplicación sin requerir credenciales de Vercel o configuraciones adicionales, cumpliendo perfectamente con los objetivos de demostración académica del proyecto.

### Figura 18

Despliegue de la aplicación en internet



*Nota:* Imagen elaborada por el autor del documento



## 3.7 Documentación de la aplicación

### 3.7.1 *Manual de Usuario*

Se desarrolló un manual de usuario completo de 20 páginas que abarca todos los aspectos operacionales de la aplicación. El manual está estructurado en 13 secciones principales que incluyen desde la introducción y arquitectura hasta las especificaciones técnicas y solución de problemas.

### 3.7.2 *Estructura de la Documentación*

El manual incluye:

- **Guías de navegación:** Para usuarios finales y administradores.
- **Instrucciones de configuración:** Setup local y requerimientos del sistema
- **Funcionalidades detalladas:** Solución de problemas comunes.
- **Troubleshooting:** Solución de problemas comunes.
- **Especificaciones técnicas:** APIs internas y arquitectura de componentes.

### 3.7.3 *Documentación Técnica*

La documentación técnica incluye diagramas de flujos de datos, especificaciones de las interfaces TypeScript, comandos de diagnóstico, y guías de mantenimiento. Se proporcionan ejemplos de código y comandos de consola para verificación y debugging del sistema.

El manual sirve como recurso completo tanto para usuarios finales que desean utilizar la aplicación como para desarrolladores que requieran entender, mantener o extender el sistema en futuras iteraciones.

El manual de usuario se encuentra en docs/ManualUsuario

## 4. Conclusiones

De manera inicial, el desarrollo de la aplicación web para la empresa Kibu, cumplió satisfactoriamente con todos los objetivos planteados inicialmente. Se logró crear una plataforma integral que permite la visualización y gestión de servicios tecnológicos, implementando una interfaz web, reflejando la identidad corporativa de Kibu y brindando una experiencia de usuario intuitiva, además de un sistema administrativo funcional. La aplicación demuestra la capacidad de construir soluciones web modernas utilizando tecnologías actuales del frontend.

La implementación exitosa de React con TypeScript proporcionó una base sólida para el desarrollo de componentes reutilizables y mantenibles. La integración de Tailwind CSS permitió crear una interfaz visual coherente y responsiva que se adapta efectivamente a diferentes dispositivos. El sistema de persistencia basado en localStorage demostró ser una solución para simular operaciones de base de datos sin requerir infraestructura backend. El despliegue en Vercel estableció un flujo de integración continua que facilita las actualizaciones y mantenimiento del sistema. La automatización del proceso de deploy desde GitHub garantiza que los cambios en el código se reflejen inmediatamente en la aplicación en línea.

Finalmente, el desarrollo de la aplicación para Kibu permitió demostrar la aplicación práctica de conceptos teóricos de desarrollo web en un contexto empresarial real. La metodología empleada, desde el análisis de requerimientos hasta el despliegue en producción, proporciona un marco de referencia para futuros proyectos de desarrollo frontend. El proyecto establece una base sólida para futuras expansiones que podrían incluir la implementación de un backend real con base de datos relacional, integración de sistemas de pago, desarrollo de nuevas funcionalidades móviles.

## 5. Referencias

- Álvarez, M. A., Alvarez, S., Nadie, J., Rousset, D., Vega, J. V., Tresancos, J. P., & Lurita, J. R. C. (2017). Manual de CSS 3. *Desarrollo. web [en línea]*, 2-3.
- Eguíluz Pérez, J. (2012). Introducción a JavaScript.
- Galarza, P. C. (2016). Usabilidad web y la experiencia de usuario. *Tecnología & Diseño*, (6).
- Nikiforova, O., Babris, K., & Mahmoudifar, F. (2024). Automated generation of Web application front-end components from user interface Mockups. In *Proceedings of International Conference on Software Technologies* (Vol. 1, pp. 100-111).
- Tabarés Gutiérrez, R. (2012). El inicio de la Web: historia y cronología del hipertexto hasta HTML 4.0 (1990-99). *ArtefaCToS: revista del Instituto de Estudios de la Ciencia y la Tecnología*: 5, 1, 2012, 57-82.