Documentación Front-end Proyecto Comparador de Celulares

Autos: Deiner Mateo Castro Useche

Fecha: 14/01/2025

Cuidad: Bogota DC– Colombia

URL: <https://front-cel.vercel.app/>

URL REPO FRONT-END: <https://github.com/Mateperry/Front-cel.git>

URL REPO BACK-END: <https://github.com/Mateperry/Back-Cel.git>

**Comparador de Móviles (Celulares)**

**Introducción**

**Propósito del proyecto:**

El objetivo principal de este proyecto es desarrollar una aplicación web innovadora para la comparación de teléfonos móviles. Esta plataforma permitirá a los usuarios seleccionar y comparar una amplia gama de modelos de diferentes marcas y gamas, proporcionando una visión detallada de las características clave de cada dispositivo, como el rendimiento, la cámara, la batería, la pantalla y otros aspectos fundamentales. La finalidad es ayudar a los consumidores a tomar decisiones informadas, ajustadas a sus necesidades y preferencias personales. El comparador estará diseñado para ofrecer una experiencia intuitiva, accesible y eficiente, asegurando que los usuarios puedan encontrar el móvil ideal de manera rápida y sencilla, priorizando lo que más les importa al momento de realizar su compra.

**Arquitectura**

La arquitectura del proyecto se basará en un enfoque simple y minimalista, garantizando una experiencia de usuario limpia y sin distracciones. El sistema estará dividido en tres secciones principales:

1. **Inicio**: Página principal donde los usuarios podrán acceder a la funcionalidad de comparación.
2. **Comparador**: Sección central de la aplicación, donde los usuarios podrán seleccionar y comparar hasta 4 modelos de teléfonos, ver sus especificaciones y contrastar sus características de manera directa.
3. **Temas Legales**: Sección dedicada a los aspectos legales, que incluirá la política de privacidad, términos de uso y condiciones del servicio, asegurando que los usuarios comprendan el marco legal de la plataforma.

La arquitectura estará diseñada para ser fácil de navegar y completamente responsiva, garantizando que los usuarios puedan acceder a la plataforma desde diferentes dispositivos sin inconvenientes.

**Tecnologías utilizadas**

En este proyecto se emplearán diversas tecnologías modernas para garantizar una experiencia de usuario eficiente y fluida. Las principales herramientas y tecnologías utilizadas serán:

**Front-End**

* **HTML5**: Para la estructuración del contenido y la creación de la base de la página web.
* **CSS3**: Para la estilización visual y la implementación de diseño responsivo, asegurando que la página se adapte correctamente a diferentes tamaños de pantalla.
* **JavaScript**: Para la implementación de la lógica y la interactividad en la aplicación.
* **React**: Como la herramienta principal para construir la interfaz de usuario (UI). React permitirá crear una experiencia dinámica y eficiente, gestionando el estado de la aplicación de manera óptima y mejorando el rendimiento de la misma.
* **Tailwind CSS**: Un framework de CSS que facilitará la creación de un diseño moderno y limpio de forma rápida y flexible, permitiendo personalización sin necesidad de escribir un CSS complejo desde cero.

**Back-End**

* **Next.js API Routes**: Para la creación de endpoints backend dentro del mismo proyecto, permitiendo manejar peticiones sin un servidor adicional.
* **Node.js**: Entorno de ejecución de JavaScript en el servidor, utilizado para manejar peticiones y procesos del backend.
* **MySQL**: Sistema de gestión de bases de datos relacional que permite almacenar y gestionar la información de manera eficiente.
* **Axios**: Librería utilizada para realizar solicitudes HTTP desde el front-end hacia el backend, facilitando la comunicación con la API.

Estas tecnologías serán utilizadas de manera conjunta para crear una página web que no solo sea visualmente atractiva, sino también fácil de navegar y altamente interactiva, brindando una experiencia de usuario excepcional.

**Instalación y configuración**

Para configurar y ejecutar este proyecto de manera local, sigue los pasos a continuación. Esto te permitirá tener una copia del proyecto en tu máquina y empezar a trabajar con él.

**1. Requisitos previos**

Asegúrate de tener instaladas las siguientes herramientas:

* **Node.js**: Es necesario tener Node.js instalado para ejecutar el proyecto. Puedes descargar la última versión estable desde [Node.js](https://nodejs.org).
* **Git**: Para gestionar el control de versiones y trabajar con el repositorio de código. Descárgalo desde [Git](https://git-scm.com/downloads).
* **Editor de código**: Se recomienda **Visual Studio Code** para un entorno de desarrollo óptimo. Puedes descargarlo desde [VS Code](https://code.visualstudio.com).

**2. Despliegue en GitHub Pages**

Aunque el proyecto ya está desplegado y disponible en línea, puedes realizar cambios y actualizarlo localmente. Para hacerlo, sigue estos pasos:

1. **Construir la versión optimizada para producción**:

**Terminal:** npm run build

1. **Subir los archivos generados a la rama** gh-pages:

**Terminal:** git push origin gh-pages

Así, la aplicación estará disponible públicamente en la URL de GitHub Pages correspondiente a tu repositorio.

**3. Clonar el repositorio del proyecto**

Para trabajar con el proyecto en tu máquina local, clona el repositorio desde GitHub. Abre la terminal (o el terminal integrado de Visual Studio Code) y ejecuta:

git clone https://github.com/Mateperry/comparacionmovil.git

Esto descargará el proyecto en tu máquina local.

**4. Instalar las dependencias del proyecto**

Navega al directorio del proyecto y ejecuta el siguiente comando para instalar todas las dependencias necesarias:

**Terminal:** cd comparador-moviles

**Terminal:** npm install

Este comando instalará todos los paquetes necesarios como **React**, **Tailwind CSS**, **Vite**, entre otros.

**5. Configurar el entorno de desarrollo**

1. Abre el proyecto en **Visual Studio Code**:  
   Usa **File > Open Folder...** y selecciona la carpeta del proyecto clonado.
2. Si lo deseas, instala extensiones recomendadas para un desarrollo más eficiente, como **ESLint**, **Prettier**, y **Bracket Pair Colorizer**.

**6. Ejecutar el servidor de desarrollo**

Para ver la aplicación funcionando localmente, ejecuta:

npm run dev

Esto iniciará el servidor de desarrollo usando **Vite**. El proyecto se compilará y se servirá en tiempo real. Una vez que el servidor esté en funcionamiento, podrás acceder a la aplicación en <http://localhost:3000>.

**7. Desarrollar y realizar cambios**

Durante el desarrollo, puedes hacer cambios en los archivos del proyecto (por ejemplo, en los componentes de React o en las hojas de estilo de Tailwind CSS). **Vite** detectará automáticamente los cambios y actualizará la página en el navegador en tiempo real.

**8. Control de versiones con Git**

Usa **Git** para gestionar el código y mantener el proyecto actualizado. Aquí tienes algunos comandos útiles:

* Para añadir cambios al área de preparación:

**Terminal:** git add .

* Para hacer un commit con un mensaje descriptivo:

**Terminal:** git commit -m "Descripción de los cambios realizados"

* Para subir los cambios al repositorio en GitHub:

**Terminal:** git push origin main

**Funcionalidades clave**

A continuación, se describen las principales características implementadas en el proyecto **Comparador de Celulares**:

**1. Comparación entre múltiples dispositivos**

El sistema permite comparar hasta **4 celulares** simultáneamente. El usuario puede seleccionar diferentes modelos de celulares desde una lista de opciones y visualizar sus especificaciones de manera organizada.

* **Comparación de características**: El proyecto presenta especificaciones detalladas de cada celular, como la pantalla, procesador, cámara, batería, etc.
* **Interfaz interactiva**: Las especificaciones se muestran de manera clara y accesible, permitiendo una comparación visual entre los dispositivos seleccionados.

**2. Búsqueda avanzada de celulares**

La aplicación cuenta con un sistema de búsqueda que permite al usuario filtrar y buscar celulares según diversos parámetros como marca, características específicas (como cámara o tamaño de pantalla), entre otros.

* **Filtros personalizables**: El usuario puede filtrar las opciones de celulares según diferentes criterios, como el sistema operativo, el rango de precio, o el tipo de pantalla.
* **Búsqueda rápida**: Al escribir el nombre de un celular en el campo de búsqueda, la aplicación muestra resultados en tiempo real que coinciden con el texto ingresado.

**3. Interfaz de usuario responsiva**

El diseño de la interfaz es completamente **responsivo**, lo que permite que el sitio web se vea correctamente en dispositivos de diferentes tamaños, desde computadoras de escritorio hasta móviles y tabletas.

* **Diseño adaptativo**: La disposición de los elementos se ajusta de manera fluida según el tamaño de la pantalla del dispositivo utilizado.
* **Optimización móvil**: Las funcionalidades clave están adaptadas para una experiencia óptima en dispositivos móviles, garantizando una navegación cómoda.

**4. Enlaces a tiendas para compra**

Cada celular tiene enlaces directos a tiendas en línea donde el usuario puede comprar el dispositivo seleccionado. Esta funcionalidad permite a los usuarios hacer compras rápidamente sin salir de la página.

* **Enlaces a tiendas asociadas**: Los usuarios pueden acceder a diferentes opciones de compra, como Amazon, eBay, o tiendas locales, según la disponibilidad del celular.
* **Actualización de disponibilidad**: La información de disponibilidad y precios se actualiza en tiempo real cuando el usuario selecciona un celular.

**5. Descripción detallada y evaluación de celulares**

Cada celular tiene una sección de descripción con detalles completos sobre sus características técnicas y de rendimiento, además de opiniones de usuarios.

* **Valoraciones de usuarios**: Los usuarios pueden ver las valoraciones promedio de otros compradores para cada modelo, lo que ayuda en la toma de decisiones.
* **Comparativa visual de características**: La interfaz permite comparar de forma visual características como la calidad de la cámara, rendimiento del procesador y duración de la batería.

**6. Implementación con tecnologías modernas**

El proyecto ha sido desarrollado utilizando tecnologías modernas para asegurar un rendimiento rápido y una experiencia de usuario fluida.

* **React.js**: Utilizado para gestionar el estado de la aplicación y construir una interfaz de usuario interactiva y dinámica.
* **Tailwind CSS**: Framework de CSS para un diseño limpio y responsivo con un enfoque en la personalización.
* **Vite**: Herramienta de construcción rápida que optimiza la experiencia de desarrollo y garantiza tiempos de carga rápidos en producción.

**7. Estructura de Proyecto FRONT-END :** En el proyecto manejaremos una estructura que será de la siguiente forma para guiarnos

**Cara\_Cel\_Front**

├── **dist**

├── **node\_modules**

├── **public**

│ └── vite.svg

├── **src**

│ ├── assets

│ ├── App.css

│ ├── App.jsx

│ ├── Contenido.jsx

│ ├── Footer.jsx

│ ├── Header.jsx

│ ├── index.css

│ ├── Inicio.jsx

│ ├── main.jsx

│ ├── TemasLegales.jsx

**├── .gitignore**

**├── eslint.config.js**

**├── index.html**

**├── package-lock.json**

**├── package.json**

**├── postcss.config.js**

**├── README.md**

**├── tailwind.config.js**

**├── vite.config.js**

**└── Documentación**

└── documento

**8. Explicación Código Hoja por hoja Front-End**

Es importante resaltar que lo que haremos en este punto será explicar las hojas que tenemos en el src, algunas modificaciones en el package.json, las cuales son las más importantes. Antes de comenzar es relevante decir que tenemos que tener un index.html, que lo podemos ver en la estructura de antes, explicaremos que modificaciones tiene este.

* Index.html:

Como podemos observar, este código es una html5 normal, hemos puesto un href para unos iconos con Google.

El titulo lo predefinimos con el nombre: “Comparación Movil”.

En el body que será lo más importante que haremos acá en el HTML tiene un div con un id “root”, este nos funcionará para que react haga los cambios en este sector y utilice este div como referencia y será el padre de toda la página.

En el script, lo hacemos para la conexión de HTML con el js asi podemos hacer que la pagina tenga interactividad.

* <!doctype html>
* <html lang="en">
* <head>
* <meta charset="UTF-8" />
* <link rel="icon" type="image/svg+xml" href="/vite.svg" />
* <meta name="viewport" content="width=device-width, initial-scale=1.0" />
* <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap" rel="stylesheet">
* <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet" />
* <title>Comparación Movil</title>
* </head>
* <body>
* <div id="root"></div>
* <script type="module" src="/src/main.jsx"></script>
* </body>
* </html>

Nos dirigimos a la carpeta src, en la cual vamos a poder tener nuestros componentes. Explicare brevemente aquí y seguiremos con una explicación más a profundidad, en esta carpeta tenemos varios archivos, en este caso los importantes o más importantes son: Main.jsx, App.jsx, index.css. Estos serán los más importantes ya que tenemos en el main la conexión con el html de antes, la App.jsx es el que manejará los componentes que traeremos después y el index.css será para tener nuestros estilos con el framework tailwind. Pasemos a la explicación a profundidad.

* **main.jsx:** El archivo main.jsx es el punto de entrada principal de la aplicación. Aquí se realizan las configuraciones iniciales para montar la aplicación en el DOM. A continuación, se explican las importaciones y el código:

1. **import { StrictMode } from 'react'**La importación de StrictMode desde React permite envolver los componentes de la aplicación para habilitar herramientas que ayudan a detectar problemas potenciales, como ciclos de vida obsoletos y advertencias adicionales en desarrollo. Es útil para asegurar que el código siga buenas prácticas.
2. **import { createRoot } from 'react-dom/client'**  
   La función createRoot permite crear un punto de entrada para la aplicación React en el DOM. Reemplaza la función ReactDOM.render usada en versiones anteriores de React.
3. **import './index.css'**  
   Aquí se importa el archivo de estilos CSS globales, que se aplicarán a toda la aplicación.
4. **import App from './App.jsx'**  
   La importación del componente App nos permite incluir el componente raíz de la aplicación, que contiene el resto de los componentes.
5. **createRoot(document.getElementById('root')):** Obtiene el elemento DOM con el ID root para montar la aplicación. Este elemento normalmente está definido en el archivo index.html.
6. **<StrictMode>:** Envuelve el componente App para habilitar el modo estricto.
7. **<App />:** Representa el componente raíz de la aplicación.

* import { StrictMode } from 'react'
* import { createRoot } from 'react-dom/client'
* import './index.css'
* import App from './App.jsx'
* createRoot(document.getElementById('root')).render(
* <StrictMode>
* <App/>
* </StrictMode>,
* );

Ahora bien, vamos a explicar el App.jsx:

* **App.jsx:**

1. **useState:** Hook de React que permite manejar estados dentro de un componente funcional.
2. **useEffect:** Hook de React que permite realizar efectos secundarios, como suscripciones o manipulaciones del DOM.
3. **BrowserRouter (alias Router):** Componente de React Router que permite configurar rutas en aplicaciones SPA.
4. **Routes y Route:** Utilizados para definir las diferentes rutas de la aplicación.
5. **Navigate:** Componente que permite redirigir al usuario a una ruta específica.
6. **Header, Footer, Inicio, Contenido, TemasLegales:** Importaciones de otros componentes del proyecto, que representan el encabezado, pie de página y las páginas principales de la aplicación.

import React, { useState, useEffect } from 'react';

import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';

import Header from './Header';

import Footer from './Footer';

import Inicio from './Inicio';

import Contenido from './Contenido';

import TemasLegales from './TemasLegales';

1. **selectedButton:** Estado que almacena qué botón está seleccionado en el componente Header. Por defecto, se inicializa con 'inicio'.
2. **setSelectedButton:** Función que actualiza el estado selectedButton.

const App = () => {

const [selectedButton, setSelectedButton] = useState('inicio');

1. Esta función actualiza el estado **selectedButton** al botón seleccionado. Es útil para mantener sincronizados el estado y los elementos interactivos en la UI.

const App = () => {

const [selectedButton, setSelectedButton] = useState('inicio');

const handleSelectButton = (button) => {

    setSelectedButton(button);

  };

1. Este código se ejecuta una vez al cargar el componente.
2. Objetivo: Redirigir a la página de inicio (/) cuando la aplicación se recarga y la URL no coincide con la raíz.
3. **window.location.replace('/'):** Reemplaza la ruta actual por la raíz para garantizar que el usuario sea llevado al inicio.

  useEffect(() => {

    // *Cuando la página se recarga, forzamos la navegación a la página de inicio*

    if (window.location.pathname !== '/') {

      window.location.replace('/');

    }

  }, []);

1. **Estructura General**
2. **<Router>:** Envuelve toda la aplicación para habilitar la navegación entre rutas.
3. **<Header>:** Se le pasan las props selectedButton y handleSelectButton para manejar la interacción de los botones en el encabezado.
4. **<Routes>:** Define las rutas de la aplicación.

* <Route path="/" element={<Inicio />} />: Carga el componente Inicio al navegar a la raíz.
* <Route path="/comparar" element={<Contenido />} />: Carga el componente Contenido para la página de comparación.
* <Route path="/temas-legales" element={<TemasLegales />} />: Carga el componente TemasLegales para mostrar la información legal.
* <Route path="\*" element={<Navigate to="/" />} />: Redirige a la página de inicio si el usuario intenta acceder a una ruta no válida.

1. <Footer>: Renderiza el pie de página de la aplicación.

  return (

    <Router>

      <Header selectedButton={selectedButton} handleSelectButton={handleSelectButton} />

      <Routes>

        {/\* *Redirigimos siempre al inicio si estamos en la raíz o recargamos* \*/}

        <Route path="/" element={<Inicio handleSelectButton={handleSelectButton} />} />

        <Route path="/comparar" element={<Contenido handleSelectButton={handleSelectButton} />} />

        <Route path="/temas-legales" element={<TemasLegales />} />

        {/\* *Si alguna ruta no se encuentra, redirige al inicio* \*/}

        <Route path="\*" element={<Navigate to="/" />} />

      </Routes>

      <Footer />

    </Router>

  );

1. Se exporta el componente App para que pueda ser utilizado en otros archivos del proyecto.

export default App;

* **Header.jsx**

El componente Header es una barra de navegación diseñada para la aplicación **"Mi Comparador de Celulares"**. Utiliza **React**, **React Router** para la navegación y **Tailwind CSS** para los estilos. Además, tiene soporte para menús responsivos, mostrando un **menú hamburgués** en dispositivos móviles.

* 1. **Importaciones**

import React, { useState } from 'react';

import { Link } from 'react-router-dom';

* **useState**: Hook de React que permite manejar el estado dentro del componente.
* **Link**: Componente de **React Router** para la navegación entre páginas sin recargar la aplicación.

**2. Definición del Componente Header**

const Header = ({ selectedButton, handleSelectButton }) => {

Este componente recibe dos **props**:

* **selectedButton**: Indica qué botón está seleccionado actualmente, permitiendo resaltar la opción activa.
* **handleSelectButton**: Función para actualizar el estado cuando un botón de la navegación es presionado.

1. **Estado para el Menú en Dispositivos Móviles**

const [isMenuOpen, setIsMenuOpen] = useState(false);

 **isMenuOpen**: Controla si el menú móvil está abierto (true) o cerrado (false).

* **setIsMenuOpen**: Función que cambia el estado de isMenuOpen.

const toggleMenu = () => setIsMenuOpen(!isMenuOpen);

* Esta función alterna el estado del menú al hacer clic en el botón de hamburguesa.

1. **Estructura del header**

<header className="bg-[#555A54] py-2 flex justify-between items-center px-4">

El header usa **Tailwind CSS** para el diseño:

* **bg-[#555A54]**: Fondo gris oscuro.
* **py-2**: Espaciado vertical.
* **flex justify-between items-center**: Distribuye los elementos horizontalmente con alineación central.
* **px-4**: Espaciado horizontal.

1. **Logo / Nombre del Sitio**

<div className="text-white text-lg font-semibold">

      <Link to="/" onClick={() => handleSelectButton('inicio')}>

      Mi Comparador de Celulares</Link>

</div>

* Usa Link para redirigir al usuario a la página de inicio.
* Al hacer clic, ejecuta handleSelectButton('inicio'), marcando el botón "Inicio" como seleccionado.

1. **Menú en Pantallas Grandes (md:flex)**

<div className="hidden md:flex gap-6 font-sans text-white">

Este menú:

* **Oculto en móviles (hidden)**.
* **Visible en pantallas medianas y grandes (md:flex)**.
* **Distribuye los elementos horizontalmente (gap-6)**.

Cada Link representa un botón de navegación:

<Link

  to="/"

  onClick={() => handleSelectButton('inicio')}

  className={`text-center text-white text-[18px] m-2 p-2 border-2 border-solid border- transparent transition-all duration-200 rounded-[5px] ${selectedButton === 'inicio' ? 'bg-[#4CAF50] text-white' : 'hover:bg-[#4CAF50] hover:text-white'}`}>Inicio

</Link>

* Si el botón está seleccionado (selectedButton === 'inicio'), cambia el color de fondo a verde.
* Si no está seleccionado, cambia el color al pasar el cursor (hover:bg-[#4CAF50]).

Este patrón se repite para las otras opciones:

* **Comparar Móviles (/comparar)**.
* **Temas Legales (/temas-legales)**.

1. **Botón de Menú Hamburguesa (Móviles)**

<button

className="md:hidden text-white p-2" onClick={toggleMenu}>

<i className={`fas ${isMenuOpen ? 'fa-times' : 'fa-bars'} text-2xl`}></i>

</button>

* **Visible solo en móviles (md:hidden)**.
* **Al hacer clic, alterna isMenuOpen**.
* **Muestra fa-bars (menú cerrado) o fa-times (menú abierto)**.

1. **Menú Desplegable en Móviles**

 <div className={`md:hidden ${isMenuOpen ? 'block' : 'hidden'} absolute top-0 left-0 w-full bg-[#555A54] text-white p-4`}>

* **md:hidden**: Visible solo en móviles.
* **isMenuOpen ? 'block' : 'hidden'**: Se muestra o se oculta según isMenuOpen.
* **absolute top-0 left-0 w-full**: Ocupa toda la pantalla en la parte superior.

Cada opción de menú en móviles:

<Link

  to="/"

  onClick={() => {handleSelectButton('inicio');setIsMenuOpen(false);}}

  className={`block text-center text-white text-[18px] m-2 p-2 border-2 border-solid border- transparent transition-all duration-200 rounded-[5px] ${selectedButton === 'inicio' ? 'bg-[#4CAF50] text-white' : 'hover:bg-[#4CAF50] hover:text-white'}`}>Inicio

</Link>

* Similar al menú en escritorio, pero usa block para ocupar toda la línea.
* **Cierra el menú (setIsMenuOpen(false)) después de seleccionar una opción**.

1. **Exportación del Componente**

export default Header;

Permite usar Header en otros archivos del proyecto.

* **Inicio.jsx**

Este componente Inicio en **React** representa la pantalla de bienvenida de un comparador de celulares. Su propósito es mostrar un mensaje introductorio y un botón que dirige a la sección de comparación de móviles.

* 1. **Importaciones**

import React from 'react';

import { Link } from 'react-router-dom';

* React: Se importa para definir el componente funcional Inicio.
* Link (de react-router-dom): Se usa para la navegación sin recargar la página

**2. Definición del componente**

const Inicio = ({ handleSelectButton }) => {

* Inicio es un **componente funcional** que recibe handleSelectButton como **prop**.
* handleSelectButton es una función que actualiza el estado del botón seleccionado en el Header.

**3. Manejo del clic en el botón**

  const handleButtonClick = () => {

    handleSelectButton('comparar');  // *Cambia la selección a "Comparar Móviles"*

  };

* handleButtonClick se ejecuta cuando el usuario hace clic en el botón "Comienza Ahora".
* Llama a handleSelectButton con el argumento 'comparar', lo que indica que el usuario ha seleccionado la sección de comparación.

**4. Estructura y diseño del componente**

  return (

    <div className="min-h-screen flex justify-center items-start bg-[#BFBBAF] p-4 sm:p-6">

* min-h-screen: Hace que el contenedor tenga al menos la altura de la pantalla.
* flex justify-center items-start: Usa **Flexbox** para centrar el contenido horizontalmente y alinearlo en la parte superior.
* bg-[#BFBBAF]: Define un color de fondo personalizado.
* p-4 sm:p-6: Aplica **padding** con valores ajustados para diferentes tamaños de pantalla (sm para pantallas pequeñas).

<div className="bg-[#555A54] p-8 sm:p-10 md:p-12 rounded-xl shadow-xl w-full max-w-md flex flex-col gap-6 mt-20">

* bg-[#555A54]: Color de fondo oscuro para la tarjeta de bienvenida.
* p-8 sm:p-10 md:p-12: Espaciado interno que cambia según el tamaño de la pantalla.
* rounded-xl: Bordes redondeados para un diseño moderno.
* shadow-xl: Sombra para dar efecto de elevación.
* w-full max-w-md: Ajusta el ancho máximo del contenedor para que sea responsivo.
* flex flex-col gap-6: Organiza los elementos en una columna con separación (gap-6).
* mt-20: Margen superior para separarlo del borde de la pantalla.

<div className="text-white text-center">

* text-white: Texto en color blanco.
* text-center: Centra el contenido del texto.

5. **Título y descripción**

<h1 className="text-3xl sm:text-4xl font-bold mb-4">Bienvenido al Comparador de Celulares</h1>

* text-3xl sm:text-4xl: Tamaño del texto, ajustado según el tamaño de la pantalla.
* font-bold: Texto en **negrita**.
* mb-4: Margen inferior para separar del siguiente elemento.

<p className="mb-6 text-sm sm:text-base">Compara las mejores marcas y modelos de celulares.</p>

* text-sm sm:text-base: Define el tamaño del texto, aumentando en pantallas más grandes.
* mb-6: Margen inferior para separarlo del botón.

<Link

 to="/comparar"

 onClick={handleButtonClick}

 className="bg-[#4CAF50] text-white py-3 px-6 rounded-full hover:bg-[#45a049] transition-all duration-300">Comienza Ahora

</Link>

* to="/comparar": Usa **React Router** para navegar a la página de comparación sin recargar.
* onClick={handleButtonClick}: Ejecuta la función handleButtonClick al hacer clic.
* bg-[#4CAF50]: Color de fondo verde.
* text-white: Texto en blanco.
* py-3 px-6: Padding para hacer el botón más grande.
* rounded-full: Bordes completamente redondeados.
* hover:bg-[#45a049]: Cambia el color al pasar el mouse.
* transition-all duration-300: Suaviza la animación en **300ms**.

**6. Exportación del componente**

export default Inicio;

* Permite que Inicio se importe en otros archivos.
* **Footer.jsx**

Este código define un componente Footer en React, que incluye información de derechos de autor, enlaces a redes sociales y un botón que abre un modal con información sobre el proyecto.

**1. Importaciones y Estado**

import { useState } from 'react';

Solo se importa useState, ya que el componente necesita manejar estados para la selección de botones y la apertura/cierre del modal.

const Footer = () => {

  const [selectedButton, setSelectedButton] = useState('acerca');

  const [isModalOpen, setIsModalOpen] = useState(false)

* selectedButton: Estado que almacena el botón seleccionado (inicialmente 'acerca').
* isModalOpen: Estado que controla si el modal está abierto (true) o cerrado (false).

**2. Función handleSelectButton**

const handleSelectButton = (button) => {

   setSelectedButton(button); // *Actualizamos el estado con el botón seleccionado*

   if (button === 'acerca') {

    setIsModalOpen(true); // *Abre el modal cuando el botón "Acerca del sitio" es presionado*

    }

  };

* Actualiza el estado selectedButton con el botón seleccionado.
* Si el usuario hace clic en el botón "Acerca del sitio", abre el modal (setIsModalOpen(true)).

**3. Función closeModal**

  const closeModal = () => {

    setIsModalOpen(false); // *Cierra el modal*

  };

* Cierra el modal al cambiar el estado isModalOpen a false.

**4. Función handleOutsideClick**

  const handleOutsideClick = (e) => {

    if (e.target.id === "modal-background") {

      closeModal();  // *Cierra el modal si se hace clic fuera del popup*

    }

  };

* Si el usuario hace clic fuera del contenido del modal (modal-background), se cierra el modal.

**5. Estructura del return**

<footer className="bg-[#555A54] py-4 text-white ">

* footer: Pie de página con fondo oscuro y texto en blanco.

**5.1. Sección de derechos de autor**

      <div className="text-left text-sm mb-4 sm:mb-0">

        <p>&copy; 2024 Mateo Castro Useche</p>

      </div>

* Muestra el nombre del autor con derechos de autor.

**5.2. Enlaces a redes sociales**

<div className="flex gap-6 items-center mb-4 sm:mb-0">

* Cada ícono tiene un enlace a LinkedIn, portafolio y CV, con estilos y efectos hover.

**5.3. Botón "Acerca del sitio"**

<button

 className={`text-center px-4 py-2 border-2 border-solid border-transparent transition-all duration-200 rounded-full ${selectedButton === 'acerca' ? 'bg-[#4CAF50] text-white' : 'hover:bg-[#4CAF50] hover:text-white'}`}

  onClick={() => handleSelectButton('acerca')}

  aria-label="Acerca del sitio">Acerca del sitio</button>

* Aplica estilos condicionales si el botón está seleccionado (selectedButton === 'acerca').
* Llama a handleSelectButton('acerca'), que abre el modal.

**5.4. Modal (si está abierto)**

{isModalOpen && (

<div id="modal-background"

 className="fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50"

 onClick={handleOutsideClick}>

 <div className="bg-[#555A54] p-10 rounded-xl shadow-xl w-96 text-white" onClick={(e) => e.stopPropagation()}>

 <h2 className="text-2xl font-bold mb-4">Acerca de este Proyecto</h2>

 <p className="mb-4">Este proyecto fue creado por Mateo Castro Useche como parte de su portafolio. Aprendí sobre el uso de React, el manejo de estados, el diseño de interfaces y la implementación de funcionalidades interactivas como comparadores y modales.</p>

<button onClick={closeModal}

className="bg-[#4CAF50] text-white py-2 px-6 rounded-full hover:bg-[#45a049] transition-all duration-300">Cerrar

          </button>

        </div>

      </div>

* Si isModalOpen === true, se muestra un div con fondo oscuro semitransparente (bg-opacity-50).
* El contenido del modal tiene un fondo gris oscuro (bg-[#555A54]).
* Un botón "Cerrar" que llama a closeModal() para cerrar el modal.
* **Contenido.jsx**

Este código es un componente de React que permite seleccionar y comparar hasta cuatro celulares al mismo tiempo. Utiliza **React Hooks** (useState, useEffect) para manejar el estado y **Axios** para hacer una petición al backend y obtener datos de celulares. Ahora lo desglosaré paso a paso:

**1. Importación de Módulos**

import { useState, useEffect } from 'react';

import axios from 'axios';

* useState: Maneja estados dentro del componente funcional.
* useEffect: Maneja efectos secundarios, como solicitudes HTTP.
* axios: Se usa para hacer peticiones HTTP al backend.

**2. Definición del Componente Contenido**

const Contenido = () => {

Este es un **componente funcional** que representa la interfaz donde los usuarios pueden buscar, seleccionar y comparar celulares.

**3. Estados Principales**

const [celularesSeleccionados, setCelularesSeleccionados] = useState([null, null, null, null]);

const [searchTerm, setSearchTerm] = useState('');

const [celulares, setCelulares] = useState([]);

const [filteredCelulares, setFilteredCelulares] = useState([]);

const [celularIndexSeleccionado, setCelularIndexSeleccionado] = useState(null);

* celularesSeleccionados: Guarda hasta 4 celulares seleccionados (null significa que no hay celular en esa posición).
* searchTerm: Almacena el texto ingresado en la barra de búsqueda.
* celulares: Guarda la lista completa de celulares obtenidos del backend.
* filteredCelulares: Almacena los celulares filtrados según el término de búsqueda.
* celularIndexSeleccionado: Indica qué posición de comparación está activa para agregar un celular.

**4. Obtener la Lista de Celulares (useEffect)**

useEffect(() => {

axios.get('https://back-cel.onrender.com/api/celulares')

.then((response) => {

 setCelulares(response.data);

 setFilteredCelulares(response.data);

 })

 .catch((error) => {

 console.error('Hubo un error al obtener los celulares:', error);

      });

  }, []);

* Se ejecuta **una sola vez** al montar el componente ([] como segundo argumento).
* Obtiene datos de celulares desde http://localhost:3001/api/celulares.
* Guarda los datos en celulares y filteredCelulares.

**5. Filtrar Celulares al Escribir en el Input**

  const handleSearch = (e) => {

    const term = e.target.value;

    setSearchTerm(term);

    const filtered = celulares.filter((celular) =>

      celular.marca.toLowerCase().includes(term.toLowerCase()) ||

      celular.modelo.toLowerCase().includes(term.toLowerCase())

    );

    setFilteredCelulares(filtered);

  };

* handleSearch se activa al escribir en la barra de búsqueda.
* Filtra celulares por **marca** o **modelo**.
* Actualiza filteredCelulares para mostrar solo los resultados coincidentes.

**6. Seleccionar un Celular para Comparar**

const handleSelectCelular = (index, celularId) => {

const updatedCelulares = [...celularesSeleccionados];

  updatedCelulares[index] = celularId;

  setCelularesSeleccionados(updatedCelulares);

  setSearchTerm('');

  };

* index: Posición del celular a reemplazar en la comparación.
* celularId: Identificador del celular seleccionado.
* Se actualiza celularesSeleccionados con el nuevo celular en la posición indicada.
* Se **borra** el searchTerm para limpiar el input.

**7. Eliminar un Celular de la Comparación**

const handleEliminarCelular = (index) => {

const updatedCelulares = [...celularesSeleccionados];

  updatedCelulares[index] = null;

  setCelularesSeleccionados(updatedCelulares);

  };

* **Convierte en null** el celular en la posición index.
* Elimina el celular sin afectar los demás seleccionados.

**8. Renderización de la Interfaz**

<div className="min-h-screen flex justify-center items-start bg-[#BFBBAF] p-5 sm:p-6 md:p-20 mb-0">

* Ocupa toda la pantalla (min-h-screen).
* Está centrado horizontalmente (justify-center) y alineado arriba (items-start).
* Fondo color #BFBBAF y p-20 para padding en pantallas medianas.

**Botones para Elegir Celular a Comparar**

<div className="flex justify-center gap-6 mb-6 flex-wrap">

 {[0, 1, 2 ].map((index) => (

  <button

  key={index}

   onClick={() => setCelularIndexSeleccionado(index)}

   className={`py-3 px-6 rounded-full text-lg font-semibold transition-all duration-300 ${

    celularIndexSeleccionado === index ? 'bg-[#4CAF50] text-white'

    : 'bg-white text-[#555A54] border-2 border-[#4CAF50]'

    }`}>Celular {index + 1}

</button>))}

</div>

* Renderiza **tres botones** (Celular 1, 2 y 3) que permiten seleccionar dónde agregar un celular.
* Cambian de color si están seleccionados (bg-[#4CAF50] para el activo).
* setCelularIndexSeleccionado(index): Guarda cuál se seleccionó.

**Barra de Búsqueda**

<input

  type="text"

  value={searchTerm}

  onChange={handleSearch}

  placeholder="Buscar celular..."

className="py-3 px-4 rounded-md shadow-md bg-white text-black border-2 border-[#4CAF50] w-64"

/>

* Permite escribir el modelo o marca del celular.
* Al escribir, filtra los celulares con handleSearch.

**Lista de Resultados de Búsqueda**

{searchTerm && (

<div className="bg-white text-black rounded-lg shadow-lg mt-2 w-80 h-48 max-h-60 overflow-y- auto flex flex-col-reverse mx-auto">

 {filteredCelulares.map((celular) => (

 <div

key={celular.id}

  className="py-2 px-4 hover:bg-[#4CAF50] hover:text-white cursor-pointer"

  onClick={() => celularIndexSeleccionado !== null && handleSelectCelular(celularIndexSeleccionado, celular.id)}>

                  {celular.marca} {celular.modelo} {/\* *Mostrar marca y modelo* \*/}

</div>))}

</div>)}

* **Si hay un searchTerm**, se muestra la lista de celulares filtrados.
* Cada celular es un div **clickeable** para seleccionarlo.
* Usa hover:bg-[#4CAF50] para resaltar al pasar el mouse.

**Mostrar Celulares Seleccionados- Tarjetas en pantalla**

<div className="mt-6 flex gap-6 justify-center flex-wrap">

 {celularesSeleccionados.map((celularId, index) =>

  celularId ? (

<div key={index} className="flex justify-center mb-5">

 {celulares

  .filter((cel) => cel.id === celularId)

  .map((celular) => (

<div

className="flex flex-col bg-white p-4 rounded-lg shadow-lg w-full sm:w-72 text-black"

 key={celular.id}>

<h3 className="text-lg font-bold mb-2">{celular.marca} {celular.modelo}</h3> {/\* *Nombre* \*/}

<img

 src={celular.imagen\_url}

 alt={celular.marca}

className="w-64 h-64 object-contain mb-4 rounded-lg"

/>

<ul className="text-sm leading-6">

<li><strong>Marca:</strong> {celular.marca}</li>

<li><strong>Modelo:</strong> {celular.modelo}</li>

<li><strong>Línea:</strong> {celular.linea}</li>

<li><strong>Fecha de lanzamiento:</strong> {celular.anio\_lanzamiento}</li>

<p className="text-lg font-semibold text-red-700 text-center">Dimensiones y diseño</p>

<li><strong>Dimensiones:</strong> {celular.altura} x {celular.ancho} x {celular.espesor} mm</li>

<li><strong>Peso:</strong> {celular.peso} gramos</li>

<li><strong>Material del cuerpo:</strong> {celular.material\_cuerpo}</li>

<li><strong>Colores disponibles:</strong> {celular.colores\_disponibles}</li>

<p className="text-lg font-semibold text-red-700 text-center">Pantalla</p>

<li><strong>Pantalla:</strong> {celular.tipo\_pantalla} de {celular.tamano\_pantalla}"</li>

<li><strong>Resolución:</strong> {celular.resolucion} ({celular.densidad\_pixeles} ppi)</li>

<li><strong>Tasa de refresco:</strong> {celular.tasa\_refresco} Hz</li>

<li><strong>Protección:</strong> {celular.proteccion}</li>

<p className="text-lg font-semibold text-red-700 text-center">Rendimiento</p>

<li><strong>Procesador:</strong> {celular.procesador} ({celular.nucleos\_procesador} núcleos, {celular.frecuencia\_procesador} GHz)</li>

<li><strong>GPU:</strong> {celular.gpu}</li>

<li><strong>Memoria RAM:</strong> {celular.memoria\_ram} GB</li>

<li><strong>Almacenamiento:</strong> {celular.almacenamiento} GB</li>

<li><strong>Ranura SD:</strong> {celular.ranura\_sd ? 'Sí' : 'No'}</li>

<p className="text-lg font-semibold text-red-700 text-center"> Sistema operativo</p>

<li><strong>Sistema operativo:</strong> {celular.sistema\_operativo} {celular.version\_so}</li>

<li><strong>Actualizaciones garantizadas:</strong> {celular.actualizaciones\_garantizadas} años</li>

<p className="text-lg font-semibold text-red-700 text-center"> Cámaras</p>

<li><strong>Cámaras traseras:</strong> {celular.num\_camaras\_traseras} ({celular.resolucion\_camara\_principal} MP, {celular.apertura\_camara\_principal})</li>

<li><strong>Lentes adicionales:</strong> {celular.lentes\_adicionales}</li>

<li><strong>Zoom:</strong> {celular.zoom\_optico}x óptico, {celular.zoom\_digital}x digital</li>

<li><strong>Estabilización:</strong> {celular.estabilizacion}</li>

<li><strong>Grabación video trasera:</strong> {celular.grabacion\_video\_trasera}</li>

<li><strong>Cámara frontal:</strong> {celular.resolucion\_camara\_frontal} MP ({celular.apertura\_camara\_frontal})</li>

<li><strong>Grabación video frontal:</strong> {celular.grabacion\_video\_frontal}</li>

<p className="text-lg font-semibold text-red-700 text-center"> Batería</p>

<li><strong>Batería:</strong> {celular.capacidad\_bateria} mAh</li>

<li><strong>Carga rápida:</strong> {celular.carga\_rapida} W</li>

<li><strong>Carga inalámbrica:</strong> {celular.carga\_inalambrica} W</li>

<li><strong>Autonomía estimada:</strong> {celular.autonomia\_estimada} horas</li>

<p className="text-lg font-semibold text-red-700 text-center">Conectividad</p>

<li><strong>Red móvil:</strong> {celular.red\_movil}</li>

<li><strong>Wi-Fi:</strong> {celular.wifi}</li>

<li><strong>Bluetooth:</strong> {celular.bluetooth}</li>

<li><strong>NFC:</strong> {celular.nfc ? 'Sí' : 'No'}</li>

<li><strong>USB:</strong> {celular.usb}</li>

<li><strong>GPS:</strong> {celular.gps}</li>

<li><strong>Jack 3.5mm:</strong> {celular.jack\_35 ? 'Sí' : 'No'}</li>

<p className="text-lg font-semibold text-red-700 text-center">Audio</p>

<li><strong>Altavoces estéreo:</strong> {celular.altavoces\_estereo ? 'Sí' : 'No'}</li>

<li><strong>Certificación de sonido:</strong> {celular.certificacion\_sonido}</li>

<li><strong>Microfonos:</strong> {celular.microfonos}</li>

<p className="text-lg font-semibold text-red-700 text-center">Sensores</p>

<li><strong>Sensor de huellas:</strong> {celular.sensor\_huellas}</li>

<li><strong>Reconocimiento facial:</strong> {celular.reconocimiento\_facial ? 'Sí' : 'No'}</li>

<li><strong>Sensores:</strong> {celular.acelerometro ? 'Acelerómetro, ' : ''}{celular.giroscopio ? 'Giroscopio, ' : ''}{celular.brujula ? 'Brújula, ' : ''}{celular.barometro ? 'Barómetro' : ''}</li>

<p className="text-lg font-semibold text-red-700 text-center">Resistencia</p>

<li><strong>Certificación IP:</strong> {celular.certificacion\_ip}</li>

<li><strong>Resistencia a caídas:</strong> {celular.resistencia\_caidas ? 'Sí' : 'No'}</li>

<p className="text-lg font-semibold text-red-700 text-center">Precio</p>

<li><strong>Precio de lanzamiento:</strong> ${celular.precio\_lanzamiento}</li>

<li><strong>Precio actual:</strong> ${celular.precio\_actual}</li>

</ul><button

onClick={() => handleEliminarCelular(index)}

className="bg-red-500 text-white py-2 px-4 rounded-md mt-4 hover:bg-red-700">Eliminar

</button>

 </div>))}

</div>

* Muestra **hasta 3 celulares** en tarjetas con imágenes y detalles técnicos.
* Muestra **un botón de eliminar** en cada tarjeta de información.
* export default Contenido;
* **Exportamos Con el nombre Contenido**
* **TemasLegales.jsx**

**1. Importación de React**

import React from 'react';

* Importa React, que es necesario para escribir componentes en JSX.
* Aunque en versiones modernas de React (>=17), no es obligatorio importar React en cada archivo, sigue siendo una buena práctica incluirlo.

**2. Definición del componente TemasLegales**

const TemasLegales = () => {

* Define un **componente funcional** llamado TemasLegales utilizando una función flecha (=>).
* No recibe props en este caso porque todo el contenido es estático.

**3. Contenedor principal**

return (

<div className="min-h-screen bg-[#BFBBAF] p-5 sm:p-6 md:p-8 flex justify-center items-center">

* **<div>:** Es el contenedor principal del componente.
* **min-h-screen**: Asegura que el div ocupe al menos la altura completa de la pantalla.
* **bg-[#BFBBAF]**: Usa un color de fondo personalizado en formato hexadecimal.
* **p-5 sm:p-6 md:p-8**: Añade padding (espaciado interno), que varía según el tamaño de la pantalla (sm, md).
* **flex justify-center items-center**:

1. flex: Activa el modelo de caja flexible.

2. justify-center: Centra el contenido horizontalmente.

3. items-center: Centra el contenido verticalmente.

**4. Contenedor de los términos y condiciones**

<div className="bg-[#555A54] p-6 sm:p-10 rounded-xl shadow-xl w-full max-w-4xl flex flex-col gap-6 mb-20">

* **bg-[#555A54]**: Fondo oscuro para contrastar con el fondo del div anterior.
* **p-6 sm:p-10**: padding adaptable según el tamaño de pantalla.
* **rounded-xl**: Bordes redondeados.
* **shadow-xl**: Agrega una sombra para dar efecto de elevación.
* **w-full max-w-4xl**: Ocupa todo el ancho disponible, pero con un límite de 4xl (aprox. 64rem / 1024px).
* **flex flex-col gap-6**:
  1. flex flex-col: Organiza los elementos en columna.
  2. gap-6: Espacio entre los elementos internos.
* **mb-20**: Margen inferior grande para separar el footer (si lo agregas después).

**5. Encabezado principal**

<div className="text-white text-center">

 <h1 className="text-3xl sm:text-4xl font-bold mb-4">Términos y Condiciones</h1>

 <p className="text-lg sm:text-xl mb-6">Al utilizar este sitio web, aceptas cumplir con los siguientes términos y condiciones.

 </p>

</div>

* **text-white**: Texto en color blanco.
* **text-center**: Alinea el texto al centro.
* **h1**: Título principal con:
* text-3xl sm:text-4xl: Tamaño de fuente dinámico.
* font-bold: Texto en negrita.
* mb-4: Margen inferior.
* **p**: Párrafo con:
* text-lg sm:text-xl: Tamaño de fuente adaptable.
* mb-6: Margen inferior.

**6. Secciones de términos y condiciones**

<h2 className="text-2xl sm:text-3xl font-bold mb-4">1. Introducción</h2>

 <p className="mb-4">Bienvenido al comparador de celulares. Estos términos y condiciones regulan el uso de nuestro sitio web y los servicios que ofrecemos. Por favor, léelos detenidamente.

</p>

* **h2**: Título de sección con:
* text-2xl sm:text-3xl: Tamaño de fuente dinámico.
* font-bold: Negrita.
* mb-4: Margen inferior.
* **p**: Texto explicativo con margen inferior mb-4

export default TemasLegales;

* Exporta el componente para que pueda ser utilizado en otros archivos.

**9**. **Explicación Código Hoja por hoja Back-End**

Esta carpeta contiene los archivos necesarios para la implementación del backend de un sistema de comparación de celulares. Se encuentra separada del front-end para mantener una arquitectura modular y facilitar el mantenimiento del proyecto. Dentro de esta carpeta, se incluye un archivo SQL que define la estructura de la base de datos, permitiendo almacenar información detallada de los dispositivos. Además, los archivos package.json y package-lock.json gestionan las dependencias necesarias para el correcto funcionamiento del servidor, asegurando un entorno de desarrollo estable y reproducible.

El archivo server.js implementa un servidor en Node.js utilizando Express, encargado de manejar las solicitudes y la conexión con la base de datos MySQL. A través de esta API, el backend proporciona los datos requeridos por el front-end para comparar distintos modelos de celulares de manera eficiente. Al mantener este backend en una carpeta independiente, se facilita su escalabilidad, depuración y futura integración con otros servicios sin afectar la interfaz de usuario.

**7. Estructura de Proyecto Back-END:**

Cara\_Cel\_Back

├── node\_modules

│── Base\_de\_datos\_Celulares.sql

├── server.js

├── package-lock.json

├── package.json

└── Documentación

└── documento

* **Server.jsx**

Este código implementa un servidor backend con **Node.js** y **Express** para gestionar la comunicación con una base de datos **MySQL**, permitiendo recuperar información sobre celulares. A continuación, se explica en detalle cada sección del código.

**1️. Importación de módulos**

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const bodyParser = require('body-parser');

* **express**: Framework para crear servidores en Node.js.
* **mysql2**: Módulo para interactuar con bases de datos MySQL.
* **cors**: Middleware que permite que el frontend pueda hacer solicitudes al backend sin problemas de restricciones de origen (CORS).
* **bodyParser**: Middleware que permite procesar datos en formato JSON enviados en las solicitudes.

**2.Configuración del servidor**

const app = express();

const PORT = 3001;

* **app**: Se crea una instancia de Express para configurar el servidor.
* **PORT**: Define el puerto en el que el servidor escuchará las peticiones (3001 por defecto).

**3. Uso de middlewares**

app.use(cors());

app.use(bodyParser.json());

* **app.use(cors())**: Habilita **CORS** para permitir que el frontend (que puede estar en otro dominio) se comunique con el backend.
* **app.use(bodyParser.json())**: Permite que el backend procese **solicitudes en formato JSON** (por ejemplo, cuando el frontend envía datos en una petición POST).

**4. Configuración de la conexión a MySQL**

const db = mysql.createConnection({

host: 'roundhouse.proxy.rlwy.net', // *🔹 Host de Railway (Pública)*

user: 'root',

password: 'qimstoalvWGIiPWrCpbbuzyOyvObdPMb', // *🔹 Clave de Railway*

database: 'railway',

port: 23462 // *🔹 Puerto correcto*

});

* **mysql.createConnection({...})**: Crea una conexión a la base de datos.
* **host: 'roundhouse.proxy.rlwy.net':** Indica que la base de datos está alojada en la misma máquina donde corre el backend.
* **user: 'root'**: Nombre de usuario de MySQL.
* **password: ‘qimstoalvWGIiPWrCpbbuzyOyvObdPMb ':** Contraseña del usuario MySQL.
* **database:** **'railway'**: Nombre de la base de datos utilizada.
* 🔴 **Nota**: **No se recomienda** dejar la contraseña en texto plano en el código. Se debería usar un archivo .env con dotenv.

**5. Conectar a la base de datos**

db.connect(err => {

  if (err) {

    console.error('🔴 Error al conectar a MySQL:', err);

    return;

  }

  console.log('🟢 Conectado a MySQL');

});

* **db.connect(callback)**: Intenta conectar el servidor al **motor de base de datos MySQL**.
* Si hay un **error**, lo imprime en consola (console.error(...)).
* Si la conexión es **exitosa**, imprime un mensaje de confirmación.

**6. Endpoint para obtener celulares**

app.get('/api/celulares', (req, res) => {

  const sql = 'SELECT \* FROM celulares';

  db.query(sql, (err, result) => {

    if (err) {

      console.error('🔴 Error al obtener datos:', err);

      res.status(500).json({ error: 'Error en el servidor' });

      return;

    }

    res.json(result);

  });

});

* **app.get('/api/celulares', (req, res) => {...})**: Define un **endpoint GET** en http://localhost:3001/api/celulares.
* **const sql = 'SELECT \* FROM celulares'**: Consulta SQL para obtener todos los celulares de la base de datos.
* **db.query(sql, callback)**: Ejecuta la consulta en la base de datos:
  + - Si hay un error, se imprime en consola y se envía un mensaje de error (res.status(500).json(...)).
    - Si la consulta es exitosa, se envía el resultado en formato JSON al frontend.

**7. Iniciar el servidor**

app.listen(PORT, () => {

  console.log(`🟢 Servidor backend corriendo en http://localhost:${PORT}`);

});

* **app.listen(PORT, callback)**: Inicia el servidor en el puerto 3001 (o el definido en PORT).
* Cuando el servidor arranca correctamente, imprime el mensaje:  
  🟢 **"Servidor backend corriendo en** [**http://localhost:3001**](http://localhost:3001)**"**.
* **Base de datos celulares.sql**

Este código define la estructura de la base de datos para un **comparador de celulares**, estableciendo una tabla detallada (celulares) con información técnica y comercial de cada dispositivo. Luego, inserta datos iniciales sobre celulares de la marca **Samsung,Apple,Xiaomi, Huawei, Google, etc**.

**1. Creación de la base de datos**

create database comparador\_celulares;

use comparador\_celulares;

* **CREATE DATABASE comparador\_celulares;** → Crea una base de datos llamada comparador\_celulares.
* **USE comparador\_celulares;** → Indica que todas las operaciones siguientes se ejecutarán sobre esta base de datos.

**2. Creación de la tabla celulares**

CREATE TABLE celulares (

id INT AUTO\_INCREMENT PRIMARY KEY, -- ID único para cada celular

imagen\_url VARCHAR(555), -- Imagen almacenada en formato URL

marca VARCHAR(50) NOT NULL, -- Marca del celular

modelo VARCHAR(100) NOT NULL, -- Modelo del celular

linea VARCHAR(100), -- Línea o serie del celular

anio\_lanzamiento YEAR, -- Año de lanzamiento

* **id INT AUTO\_INCREMENT PRIMARY KEY** → Cada celular tiene un **ID único** que se incrementa automáticamente.
* **imagen\_url VARCHAR(555)** → Almacena la URL de una imagen del celular.
* **marca y modelo** → Indican la marca y el modelo del celular. Son **obligatorios** (NOT NULL).
* **linea** → Línea del celular (ej. Galaxy S, iPhone Pro).
* **anio\_lanzamiento** → Año en que el celular fue lanzado.

**Dimensiones y diseño**

altura FLOAT,

ancho FLOAT,

espesor FLOAT,

peso FLOAT,

material\_cuerpo VARCHAR(50),

colores\_disponibles TEXT,

* Se almacenan las dimensiones físicas del celular en **milímetros (mm)** y peso en **gramos (g)**.
* **colores\_disponibles** → Lista de colores en los que está disponible el celular.

**Especificaciones de la pantalla**

tipo\_pantalla VARCHAR(50),

tamano\_pantalla FLOAT,

resolucion VARCHAR(50),

densidad\_pixeles FLOAT,

tasa\_refresco INT,

proteccion VARCHAR(50),

* **tipo\_pantalla** → Tecnología de la pantalla (AMOLED, LCD, etc.).
* **tamano\_pantalla** → Tamaño en pulgadas.
* **resolucion** → Resolución en formato "ancho x alto" (ej. 2400x1080).
* **densidad\_pixeles** → Densidad en píxeles por pulgada (PPI).
* **tasa\_refresco** → Frecuencia en Hz (60Hz, 120Hz, etc.).
* **proteccion** → Protección de pantalla como Gorilla Glass.

**Rendimiento**

procesador VARCHAR(100),

nucleos\_procesador INT,

frecuencia\_procesador FLOAT,

gpu VARCHAR(100),

memoria\_ram INT,

almacenamiento INT,

ranura\_sd BOOLEAN,

* **procesador** → Nombre del procesador (ej. Snapdragon 8 Gen 2).
* **nucleos\_procesador** → Número de núcleos de la CPU.
* **frecuencia\_procesador** → Velocidad en GHz.
* **gpu** → Procesador gráfico (GPU).
* **memoria\_ram y almacenamiento** → Memoria RAM y almacenamiento interno en **GB**.
* **ranura\_sd** → Indica si tiene ranura para tarjeta microSD (BOOLEAN → true/false).

**Sistema operativo y actualizaciones**

sistema\_operativo VARCHAR(50),

version\_so VARCHAR(20),

actualizaciones\_garantizadas INT,

* **sistema\_operativo** → Android o iOS.
* **version\_so** → Versión del sistema operativo (ej. Android 14, iOS 17).
* **actualizaciones\_garantizadas** → Años de actualizaciones garantizadas.

**Cámaras**

num\_camaras\_traseras INT,

resolucion\_camara\_principal INT,

apertura\_camara\_principal VARCHAR(10),

lentes\_adicionales TEXT,

zoom\_optico INT,

zoom\_digital INT,

estabilizacion VARCHAR(50),

grabacion\_video\_trasera VARCHAR(50),

resolucion\_camara\_frontal INT,

apertura\_camara\_frontal VARCHAR(10),

grabacion\_video\_frontal VARCHAR(50),

* **num\_camaras\_traseras** → Número de cámaras traseras.
* **resolucion\_camara\_principal** → Resolución en MP (megapíxeles).
* **apertura\_camara\_principal** → Tamaño de apertura (f/1.8).
* **zoom\_optico y zoom\_digital** → Capacidad de zoom en "x".
* **grabacion\_video\_trasera** → Resolución máxima de video (ej. 4K 60fps).
* **resolucion\_camara\_frontal** → Resolución en MP de la cámara frontal.

**Batería**

capacidad\_bateria INT,

carga\_rapida INT,

carga\_inalambrica INT,

carga\_inversa INT,

autonomia\_estimada FLOAT,

* **capacidad\_bateria** → En mAh.
* **carga\_rapida y carga\_inalambrica** → Potencia de carga en watts.
* **carga\_inversa** → Permite cargar otros dispositivos de manera inalámbrica.
* **autonomia\_estimada** → Horas de uso promedio.

**Conectividad**

red\_movil VARCHAR(20),

wifi VARCHAR(20),

bluetooth VARCHAR(10),

nfc BOOLEAN,

usb VARCHAR(20),

gps VARCHAR(100),

jack\_35 BOOLEAN,

* **red\_movil** → 4G, 5G, etc.
* **wifi** → Versión de Wi-Fi (Wi-Fi 6E, Wi-Fi 7).
* **bluetooth** → Versión del Bluetooth.
* **nfc** → Indica si tiene NFC (true/false).
* **jack\_35** → Indica si tiene puerto de audífonos de 3.5mm.

**Sensores y resistencia**

sensor\_huellas VARCHAR(50),

reconocimiento\_facial BOOLEAN,

acelerometro BOOLEAN,

giroscopio BOOLEAN,

brujula BOOLEAN,

barometro BOOLEAN,

certificacion\_ip VARCHAR(10),

resistencia\_caidas BOOLEAN,

* **sensor\_huellas** → Ubicación del lector de huellas (pantalla, trasero, lateral).
* **certificacion\_ip** → Nivel de resistencia al agua y polvo (ej. IP68).

**Precio**

precio\_lanzamiento FLOAT,

precio\_actual FLOAT

);

* **precio\_lanzamiento** → Precio en USD al momento de salir al mercado.
* **precio\_actual** → Precio promedio actual.

**3. Inserción de datos**

INSERT INTO celulares (

imagen\_url, marca, modelo, linea, anio\_lanzamiento, altura, ancho, espesor, peso,material\_cuerpo, colores\_disponibles,tipo\_pantalla, tamano\_pantalla, resolucion, densidad\_pixeles, tasa\_refresco, proteccion, procesador,nucleos\_procesador, frecuencia\_procesador, gpu, memoria\_ram, almacenamiento, ranura\_sd, sistema\_operativo,

version\_so, actualizaciones\_garantizadas, num\_camaras\_traseras, resolucion\_camara\_principal,

apertura\_camara\_principal, lentes\_adicionales, zoom\_optico, zoom\_digital, estabilizacion,

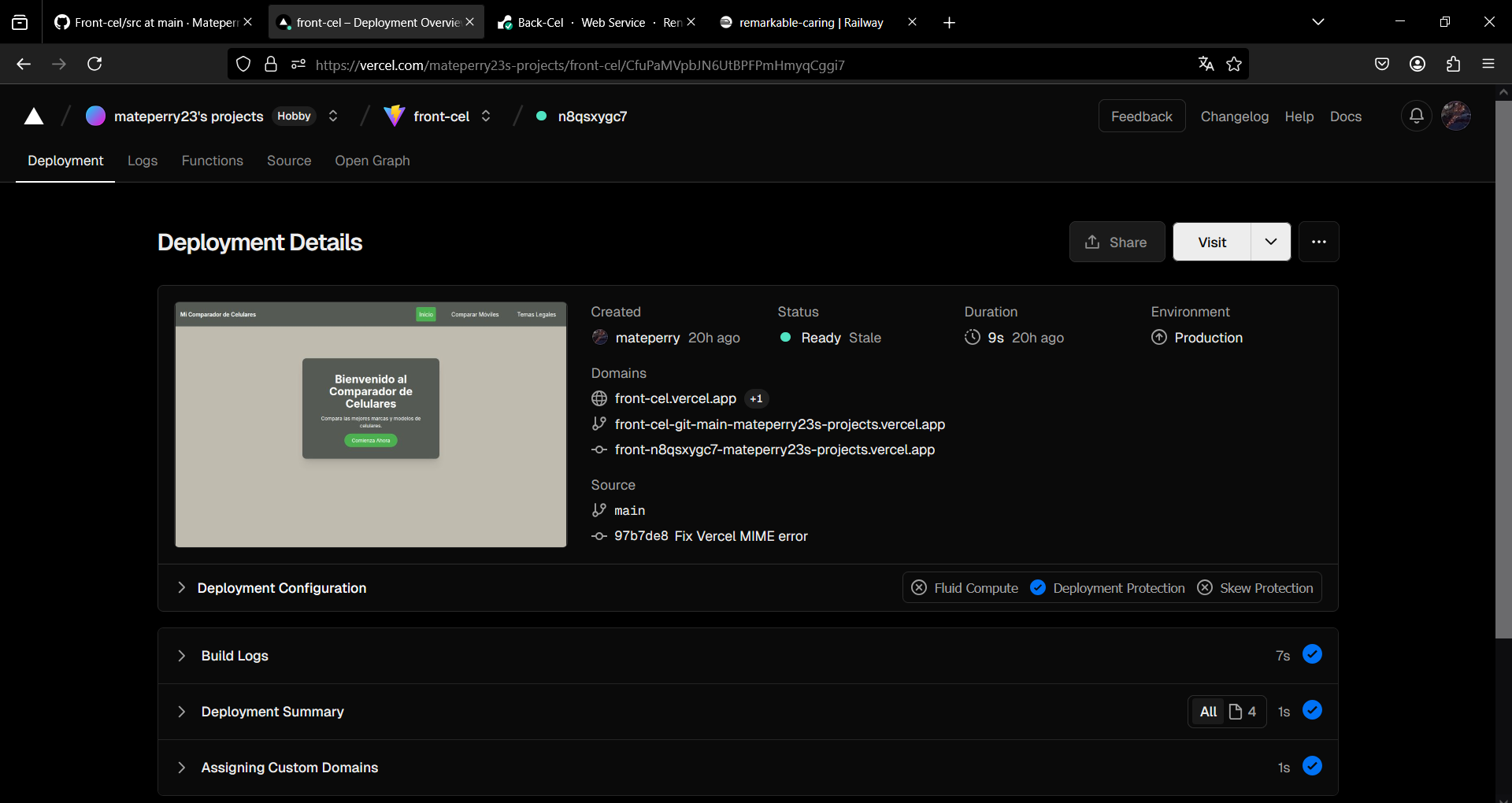
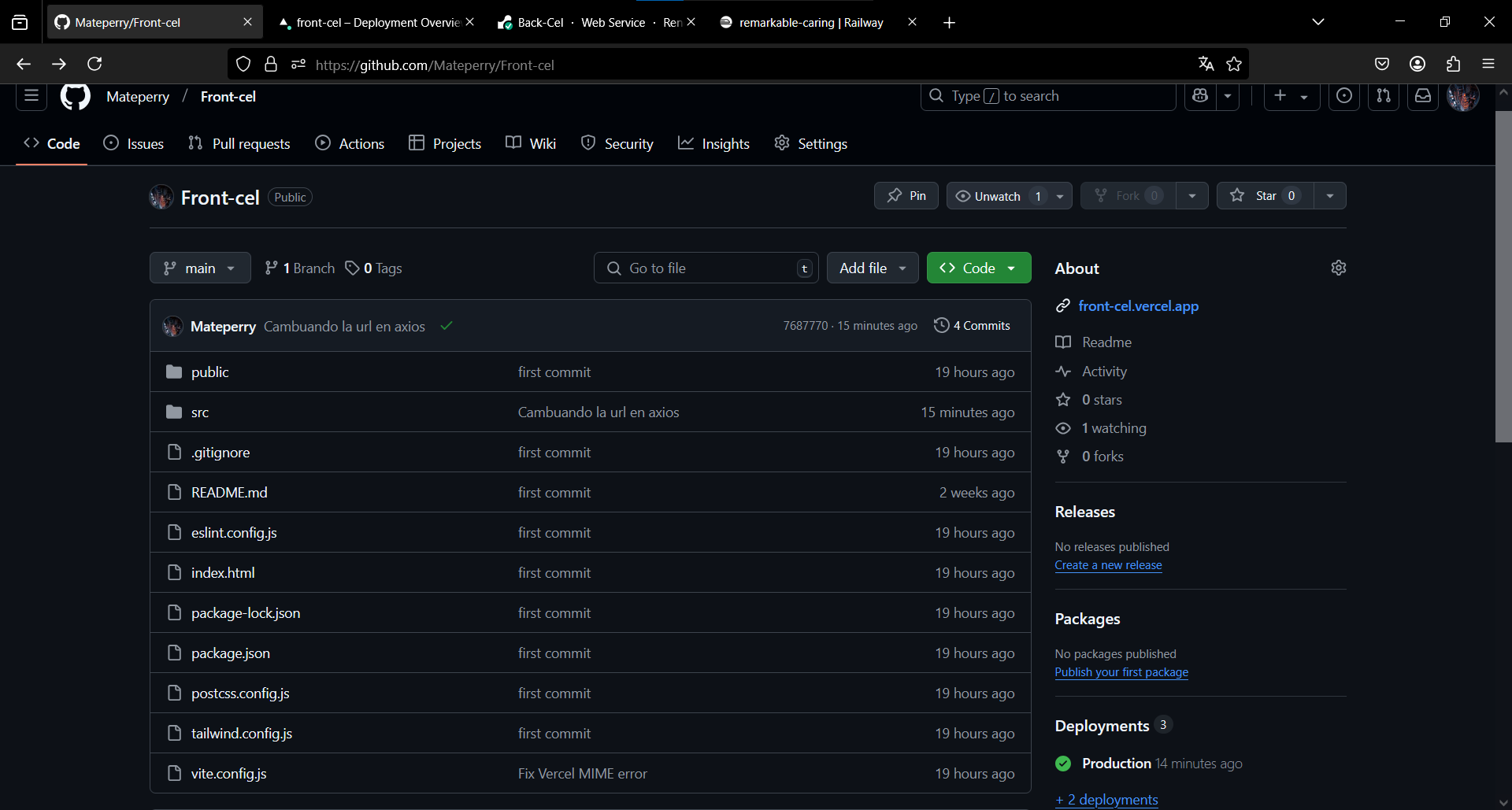
grabacion\_video\_trasera, resolucion\_camara\_frontal, apertura\_camara\_frontal, grabacion\_video\_frontal, capacidad\_bateria, carga\_rapida, carga\_inalambrica, carga\_inversa, autonomia\_estimada, red\_movil, wifi, bluetooth, nfc, usb, gps, jack\_35, altavoces\_estereo, certificacion\_sonido, microfonos, sensor\_huellas, reconocimiento\_facial, acelerometro, giroscopio, brujula, barometro, certificacion\_ip, resistencia\_caidas,precio\_lanzamiento, precio\_actual

)

* Esta consulta **inserta registros** en la tabla celulares, con datos específicos de un modelo de **Celular**
* Se asignan valores a **cada campo de la tabla**, incluyendo características técnicas y precios.
* **Deploy Front-End**

En el deploy del front-end utilizaremos github como repositorio del proyecto y vercel para que tengamos un link y vista de nuestro proyecto.

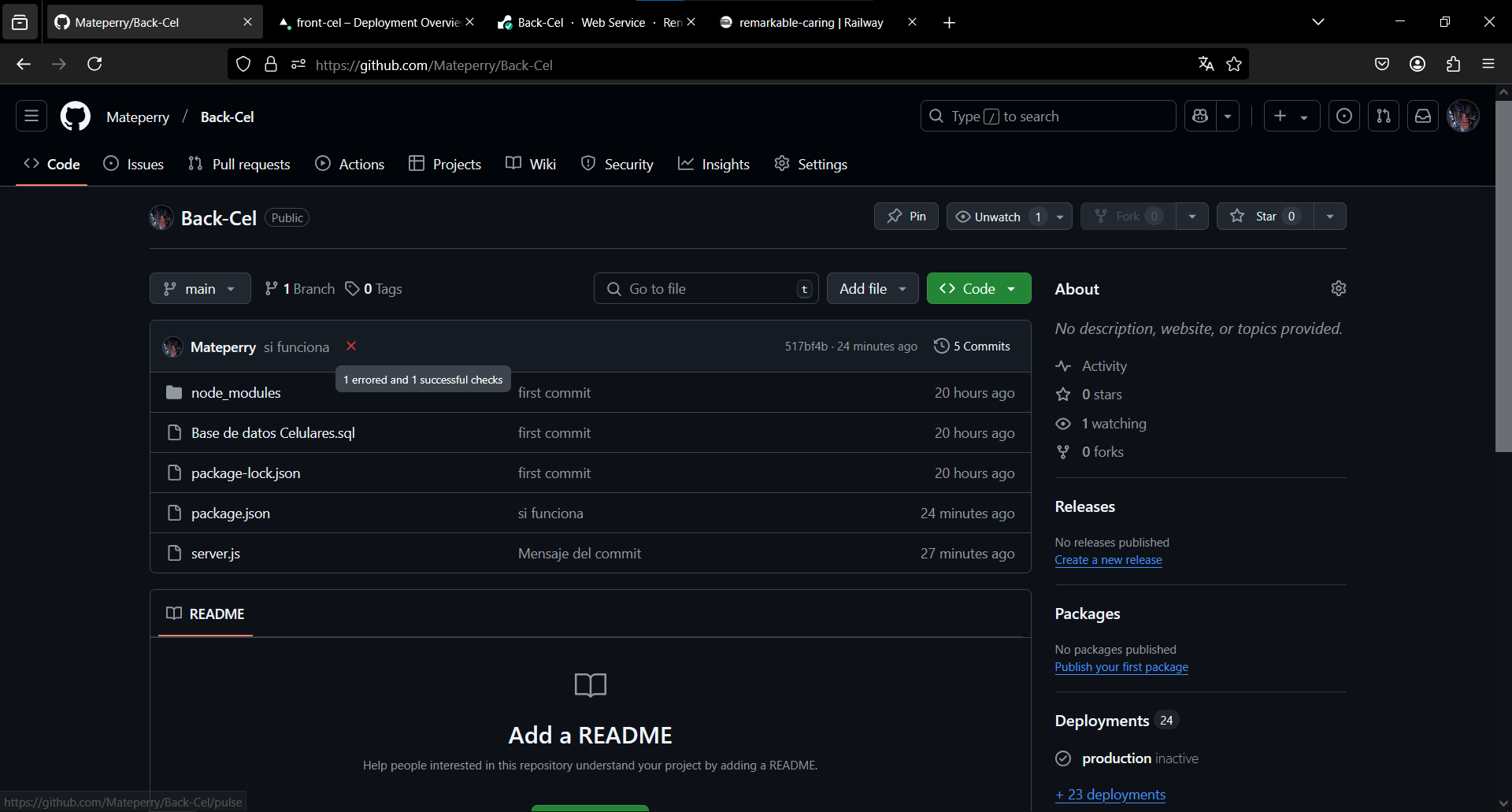
Imagen descriptiva:

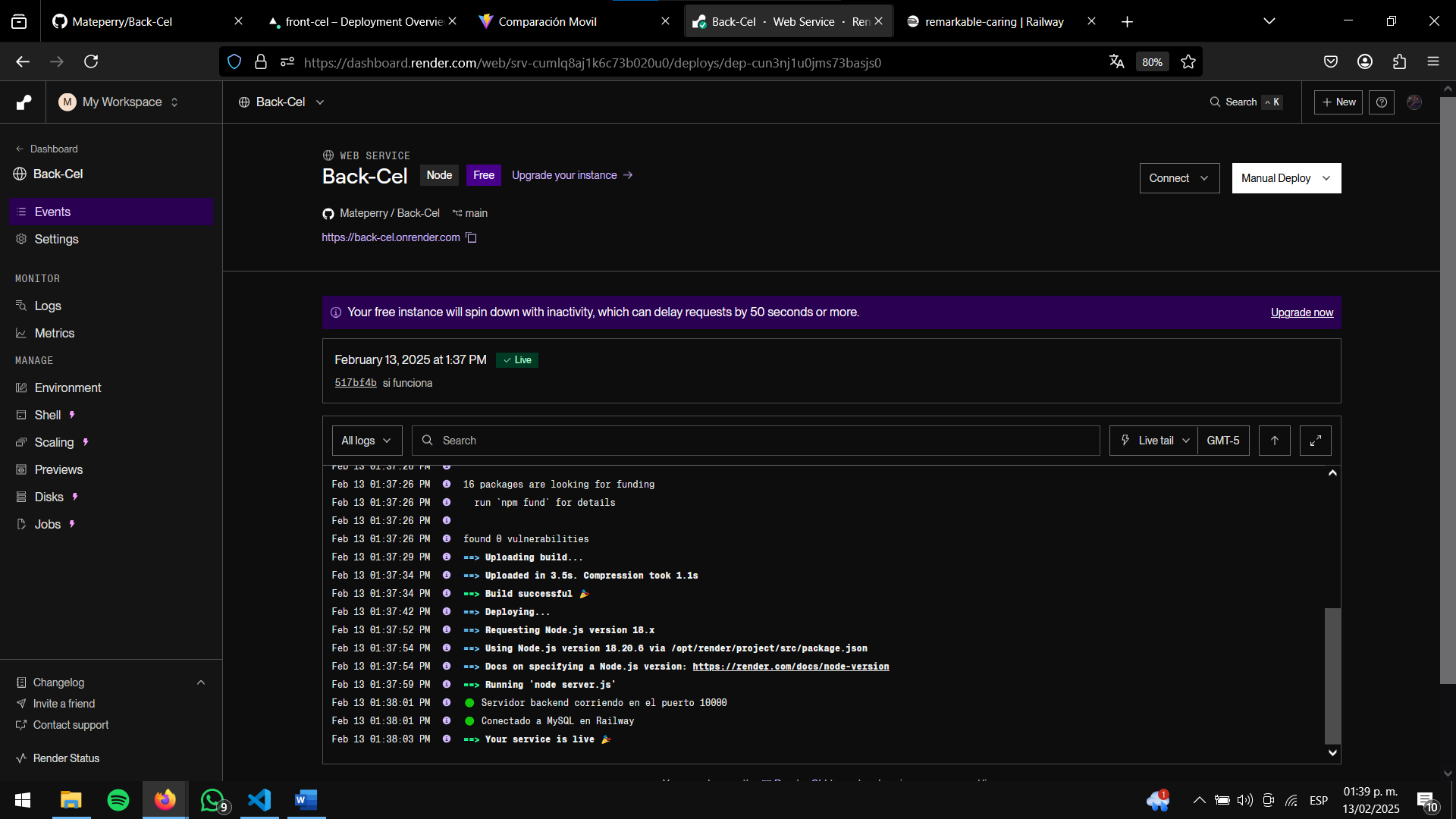


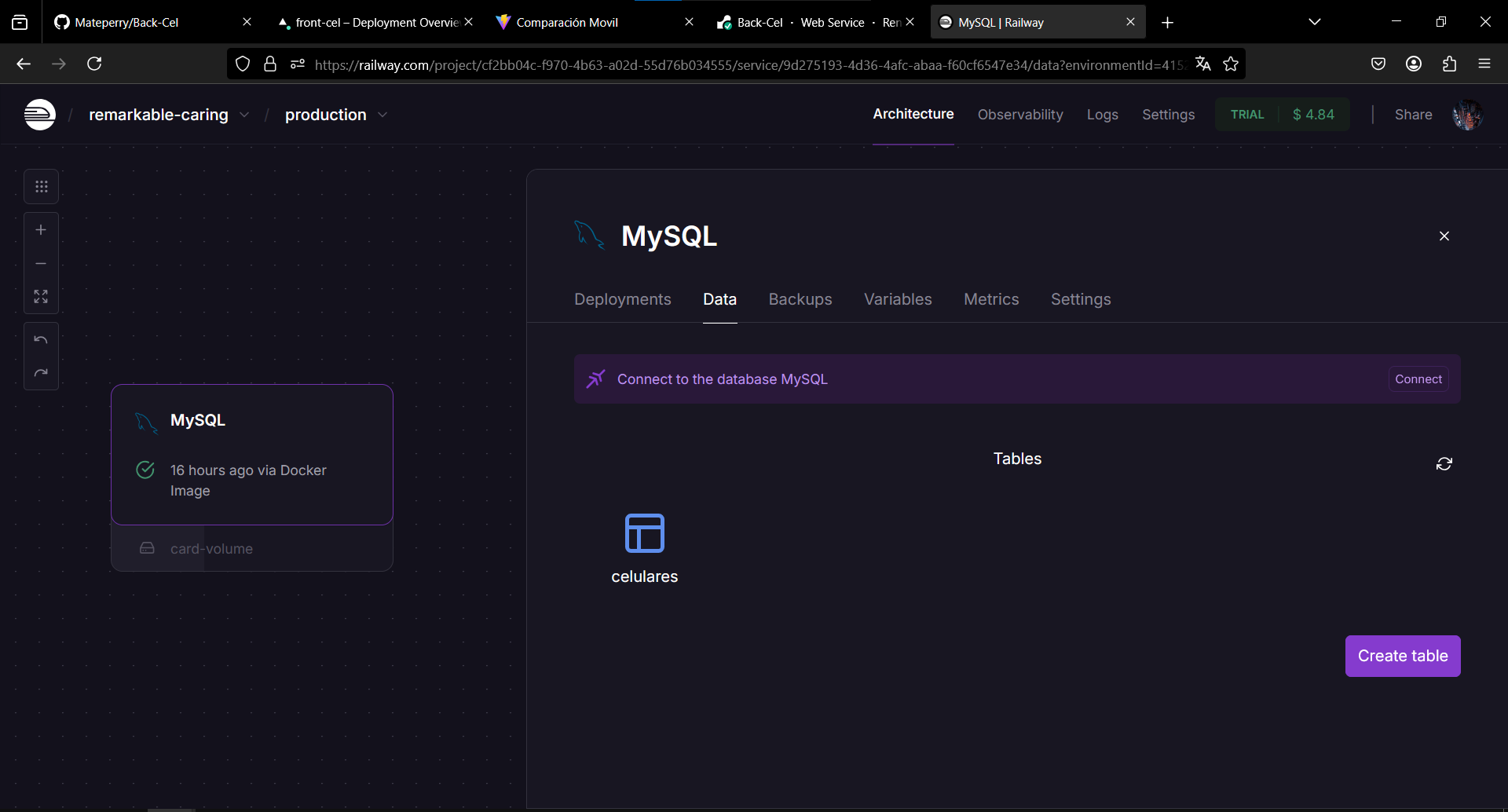
* **Deploy Back-End**

En el deploy del Back-end utilizaremos github como repositorio del proyecto, render para subirlo con git-hub , la base de datos la subiremos a railway y conectaremos render y railway, para que tengamos un link y asi proporcionarlo en nuestro front-end.

Imagen descriptiva:

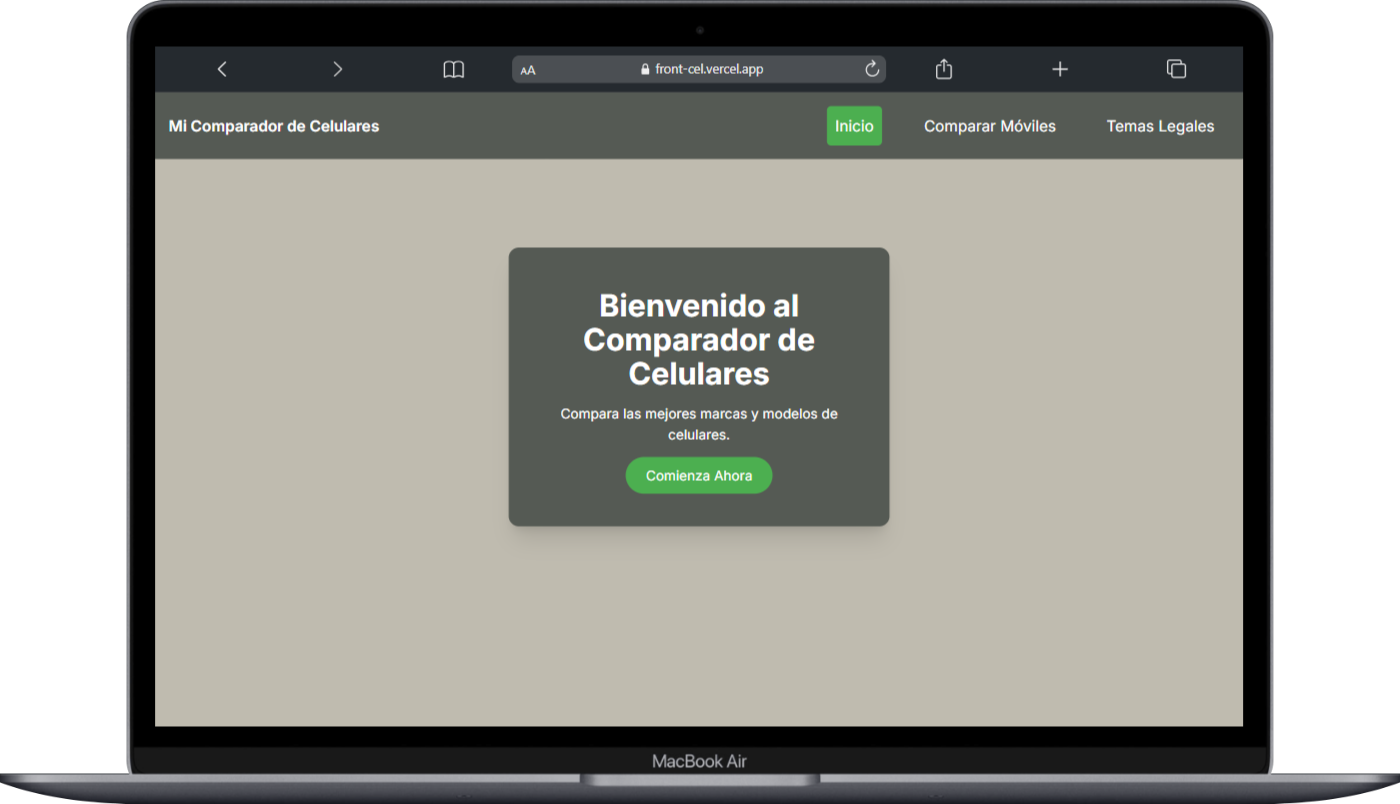


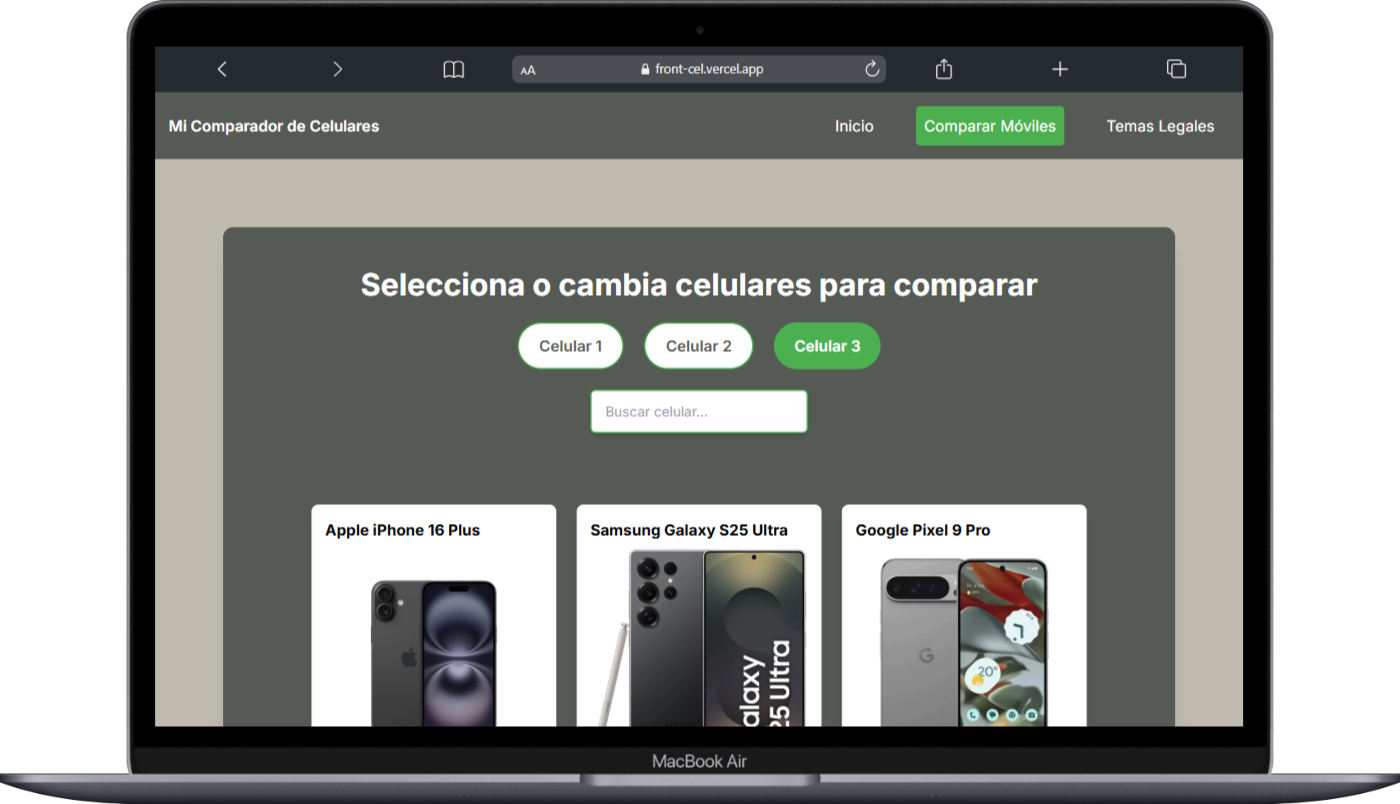




Vista previa: URL: <https://front-cel.vercel.app/>

**En pc:**

****

****

**Móvil:**

**** ****