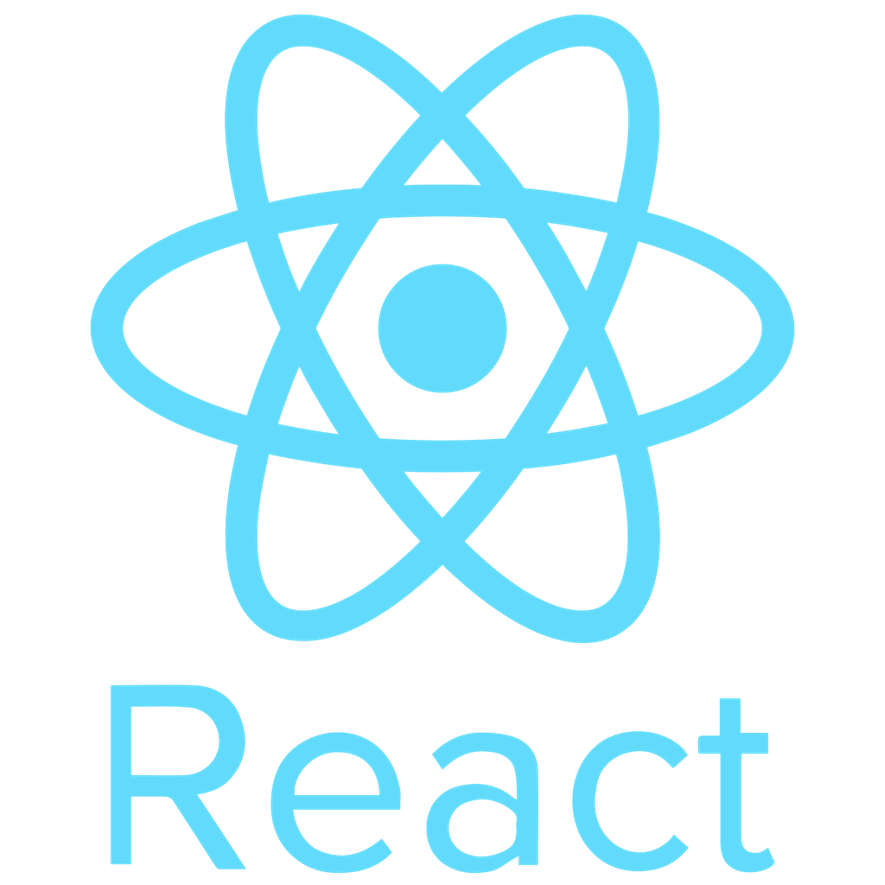
****

**JEROME GAMBOA Y MARCO BATISTA**

**[C.F.G.S - 2º DESARROLLO DE APLICACIONES WEB]**

**[ASIGNATURA: DESARROLLO DE APLICACIONES WEB EN ENTORNO CLIENTE]**

# ÍNDICE

[**ÍNDICE 2**](#_heading=h.gjdgxs)

[**¿Qué es React? 3**](#_heading=h.30j0zll)

[Características principales de React: 3](#_heading=h.1fob9te)

[¿Qué necesito instalar para trabajar con React? 4](#_heading=h.3znysh7)

[**Proyecto Inicial: Manual Conceptos Básicos. 5**](#_heading=h.mk3kdx3fj595)

[1. Cómo implementar React + Vite 5](#_heading=h.8lr0tyu0ywsm)

[2. Estructura 5](#_heading=h.8mqcdrj94s6n)

[Explicación de los Archivos y Carpetas 5](#_heading=h.vxo4rsvkoe1k)

[3. Desarrollo del Proyecto (Explicación de conceptos) 6](#_heading=h.md3uedxdzow7)

[3.1. Crear y Anidar Componentes 6](#_heading=h.egtwjsgujxrd)

[3.3. Escribir marcado con JSX 6](#_heading=h.kiv2t5tlaf25)

[3.4. Añadir estilos 6](#_heading=h.pger4n5mt6jt)

[3.5. Mostrar datos 6](#_heading=h.oq08wbe8arxb)

[3.6. Renderizado condicional 7](#_heading=h.gpr7nuaxo7sj)

[3.7. Renderizado de listas 7](#_heading=h.sbobtol8zt4q)

[3.8. Responder a eventos 7](#_heading=h.po79oq23l8gf)

[3.9. Actualizar la pantalla 7](#_heading=h.7og6g7vyfd34)

[3.10. Uso de los Hooks 8](#_heading=h.59l7f6q2nfx6)

[3.11. Compartir datos entre componentes 8](#_heading=h.33wn8ug7hzi1)

[**Proyecto API y filtros 9**](#_heading=h.qosqkkanrlfq)

[1. Introducción 9](#_heading=h.smmj8kiz1saq)

[2. Instalación y Configuración 9](#_heading=h.nrvfesafq5jc)

[2.1 Creación del Proyecto con Vite 9](#_heading=h.474kte8xz9n)

[2.2 Estructura del Proyecto 10](#_heading=h.y14cke2gman9)

[3. Desarrollo del Proyecto 10](#_heading=h.g299y2f5hp7u)

[3.1 Configurar App.jsx 10](#_heading=h.i9l40gjn9mkm)

[3.2 Obtener Datos en Home.jsx 11](#_heading=h.3fyiorxgqow2)

[3.3 Crear CharacterCard.jsx 12](#_heading=h.az9b57ptzduq)

[3.4 Crear Filter.jsx 13](#_heading=h.o2bcsujedfs)

[3.5 Agregar Estilos en styles.css 13](#_heading=h.1ntz32otd6de)

[3.5 Resultado 14](#_heading=h.dyvr1jxv8vsr)

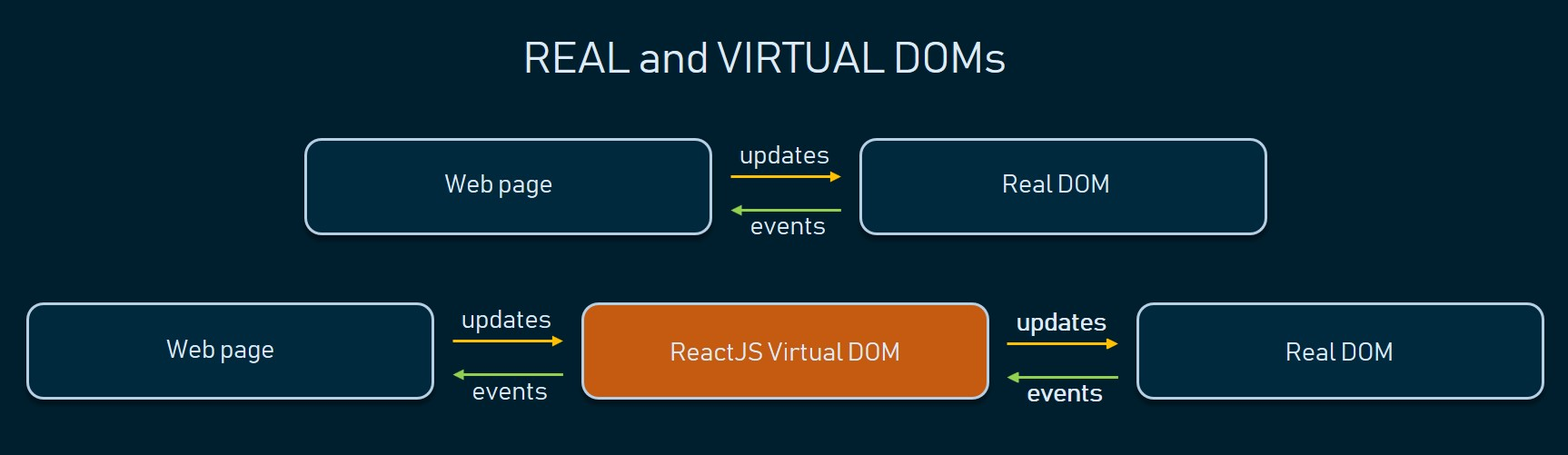
# ¿Qué es React?

React es una **biblioteca** (es decir, un conjunto de herramientas) de JavaScript que se utiliza para construir **interfaz de usuario (UI)** en aplicaciones web. Fue creada por **Facebook** y se usa para hacer que las aplicaciones web sean rápidas, interactivas y fáciles de manejar. La principal ventaja de React es que te permite crear aplicaciones donde los cambios en la interfaz se actualizan automáticamente sin tener que recargar toda la página.

Por ejemplo, si estás en una red social y alguien te da "me gusta" en una publicación, React se encarga de actualizar solo esa parte de la página para mostrar el nuevo "me gusta", sin necesidad de recargar toda la página. Esto hace que la experiencia sea mucho más fluida y rápida.

## Características principales de React:

1. **Componentes**: React usa componentes para crear partes de la interfaz de usuario. Un componente es como una "pieza" de la página que puede ser reutilizada y actualizada. Por ejemplo, puedes tener un componente que muestre un botón, y puedes usarlo en diferentes partes de tu aplicación.
2. **Virtual DOM**: El **DOM** (Document Object Model) es la representación de la estructura de la página web. React usa un **Virtual DOM**, que es una copia ligera del DOM real. Cuando algo cambia en la aplicación, React actualiza primero el Virtual DOM y luego compara con el DOM real para actualizar solo lo necesario. Esto hace que las actualizaciones sean muy rápidas.
3. **Declarativo**: React es declarativo, lo que significa que tú solo tienes que describir cómo debería lucir la interfaz en función de los datos. No necesitas preocuparte de cómo se deben hacer las actualizaciones. React se encarga de eso por ti.
4. **Unidireccionalidad de los datos**: React sigue un flujo de datos de arriba hacia abajo. Esto quiere decir que los datos fluyen de los componentes principales (padres) hacia los componentes más pequeños (hijos), lo que hace que el código sea más fácil de entender y controlar.



## 

## ¿Qué necesito instalar para trabajar con React?

Para empezar a trabajar con React, necesitas tener instalados algunos programas y herramientas:

1. **Node.js**: React necesita **Node.js**, que es un entorno de ejecución para JavaScript fuera del navegador. Node.js te permite usar herramientas como **npm** (Node Package Manager), que es necesario para instalar bibliotecas y dependencias.
   * Puedes descargar Node.js desde [su página oficial](https://nodejs.org/).
2. **Editor de código**: Necesitarás un editor de código donde puedas escribir tu código React. Un editor muy usado es **Visual Studio Code (VSCode)**, ya que tiene muchas herramientas útiles para trabajar con React.
   * Puedes descargar VSCode desde [su página oficial](https://code.visualstudio.com/).
3. **Instalar React**: Una vez tengas Node.js y tu editor de código, puedes crear tu primer proyecto en React. Para hacerlo, debes abrir la terminal (o consola) y usar un comando para crear una aplicación nueva. Usamos el siguiente comando de **Create React App**, que es una herramienta que configura todo automáticamente para que solo te concentres en escribir tu código:

npx create-react-app nombre-de-tu-app

Este comando descargará e instalará React y las dependencias necesarias. Luego, puedes ingresar a la carpeta de tu proyecto con:

cd nombre-de-tu-app

Y para empezar el servidor y ver tu aplicación en el navegador, usas el comando:

npm start

# 

# 

# 

# 

# Proyecto Inicial: Manual Conceptos Básicos.

Este proyecto es un manual interactivo que explica los conceptos fundamentales de React, basado en la documentación oficial. La aplicación ha sido desarrollada utilizando **React con Vite**, proporcionando ejemplos prácticos para cada concepto abordado.

## **1. Cómo implementar React + Vite**

Para ejecutar el proyecto en un entorno local, se deben seguir los siguientes pasos:

1. **Instalar Vite** de forma global si aún no se tiene instalado:  
   npm install -g create-vite
2. **Crear el proyecto con la plantilla de React:**create-vite react-manual --template react
3. **Acceder al directorio del proyecto:**cd react-manual
4. **Instalar las dependencias necesarias:**npm install
5. **Ejecutar el servidor de desarrollo:**npm run dev
6. **Abrir la aplicación en el navegador:** La aplicación estará disponible en <http://localhost:5173/>.

## **2. Estructura**

### Explicación de los Archivos y Carpetas

* **src/components/** → Contiene los componentes reutilizables de la aplicación.
  + **Header.jsx** → Muestra el encabezado de la página.
  + **Sidebar.jsx** → Contiene la barra de navegación con enlaces internos.
  + **Content.jsx** → Muestra la información detallada sobre cada concepto de React con ejemplos interactivos.
  + **Footer.jsx** → Pie de página de la aplicación.
  + **ContadorHijo.jsx** → Compartir datos entre componentes.
* **App.jsx** → Componente principal que organiza la estructura general de la página.
* **main.jsx** → Punto de entrada de la aplicación donde se renderiza el componente App.
* **index.css** → Archivo de estilos globales para la aplicación.
* **package.json** → Archivo de configuración con dependencias del proyecto.
* **vite.config.js** → Configuración del entorno Vite.

## **3. Desarrollo del Proyecto (Explicación de conceptos)**

### 3.1. Crear y Anidar Componentes

El componente principal **App.jsx** estructura la aplicación importando y renderizando los componentes **Header, Sidebar, Content y Footer,** mostrando cómo se pueden anidar diferentes componentes en React.



### 3.3. Escribir marcado con JSX

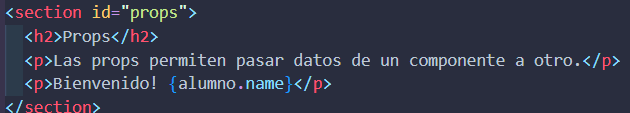
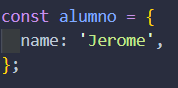
Cada componente utiliza JSX para escribir la estructura de la interfaz de usuario de una manera declarativa y similar a HTML. Por ejemplo, en **Content.jsx,** el contenido está estructurado con etiquetas JSX como **<h2>** y **<p>.**

### 3.4. Añadir estilos

Los estilos están definidos en **index.css** y aplicados a través de clases CSS en los componentes.

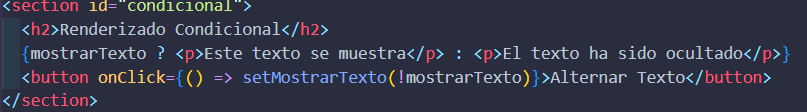
### 3.5. Mostrar datos

Se demuestra en **Content.jsx** con la variable **alumno**, la cual almacena un objeto y es mostrada mediante **{alumno.name}.**

****

### 3.6. Renderizado condicional

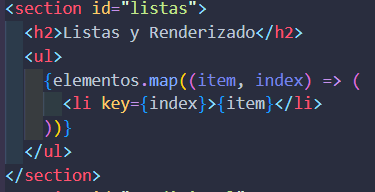
Se implementa en **Content.jsx** con **mostrarTexto**, una variable de estado que cambia al hacer clic en un botón y alterna el contenido mostrado con **{mostrarTexto ? <p>Este texto se muestra</p> : <p>El texto ha sido ocultado</p>}.**

****

### 3.7. Renderizado de listas

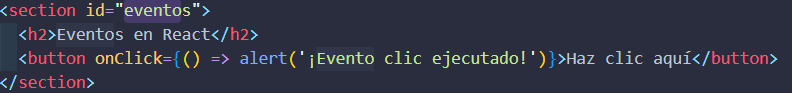
Se crea una lista de elementos y se renderiza en **Content.jsx** con:





### 3.8. Responder a eventos

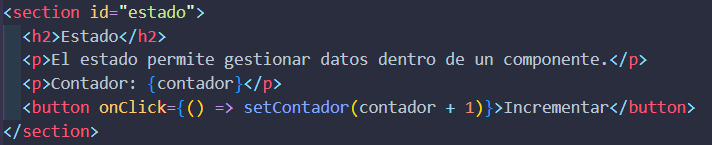
Se maneja un evento de clic en **Content.jsx** con:

Esto muestra una alerta cuando el usuario presiona el botón.

### 3.9. Actualizar la pantalla

Se utiliza el hook **useState** en **Content.jsx** para actualizar la pantalla al incrementar un contador:





Cada vez que se presiona el botón, la pantalla se actualiza con el nuevo valor del contador.

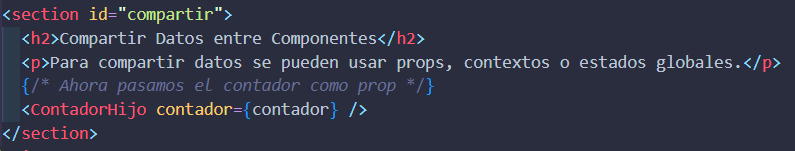
### 3.10. Uso de los Hooks

En React, un **hook** es una función especial que te permite utilizar características de React, como el estado y el ciclo de vida, sin tener que escribir una clase.

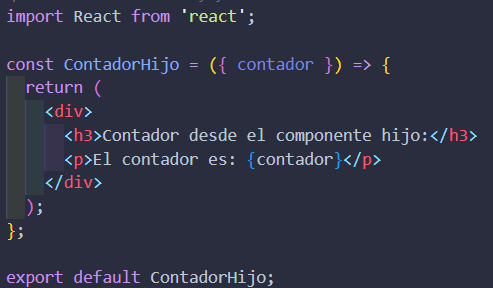
Se usa **useState** para manejar el estado del contador y el texto condicional. Este hook permite actualizar la UI sin necesidad de recargar la página.

### 3.11. Compartir datos entre componentes

El componente **ContadorHijo.jsx** recibe el estado del contador como prop desde **Content.jsx**, demostrando cómo compartir datos entre componentes padre e hijo:



En **ContadorHijo.jsx**, se recibe y muestra el valor del contador:



# 

# Proyecto API y filtros

## 1. Introducción

Esta aplicación en React obtiene datos de la API de Rick and Morty, los muestra en pantalla organizados en tarjetas y permite filtrar los personajes según su estado (Vivo, Muerto o Desconocido). Se ha desarrollado con React+Vite y Fetch API.

## 2. Instalación y Configuración

### 2.1 Creación del Proyecto con Vite

Ejecutar en la terminal:

***npm create vite@latest react-api-app --template react***

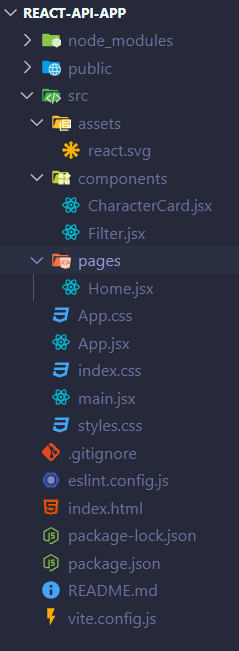
Luego, entrar en la carpeta del proyecto:

**cd react-api-app**

Instalar las dependencias necesarias:

**npm install**

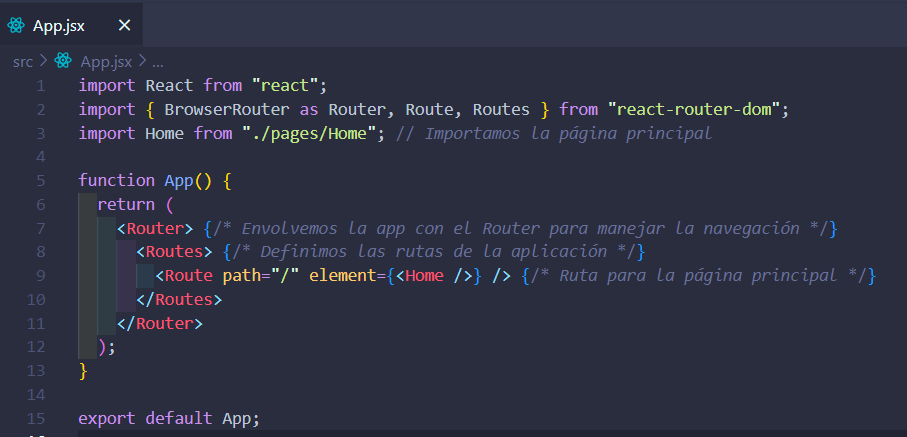
### 2.2 Estructura del Proyecto



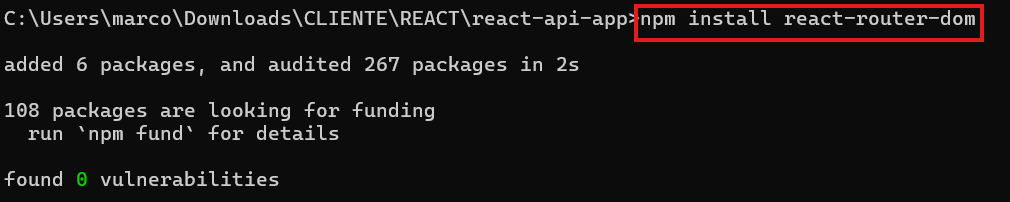
## 3. Desarrollo del Proyecto

### 3.1 Configurar App.jsx

Este archivo configura la navegación de la aplicación utilizando React Router. Define las rutas disponibles y renderiza el componente Home en la ruta principal (/).



Para que los usuarios puedan navegar entre las diferentes secciones del manual, se necesita usar React Router para crear rutas: ***npm install react-router-dom***



### 

### 

### 

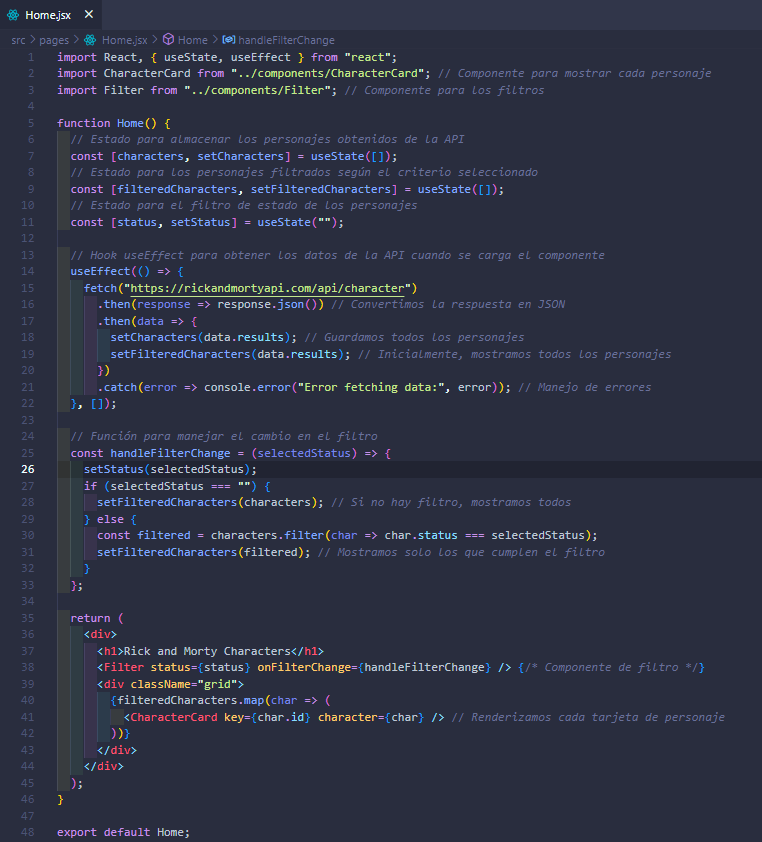
### 

### 

### 3.2 Obtener Datos en Home.jsx

Es la página principal de la aplicación. Se encarga de:

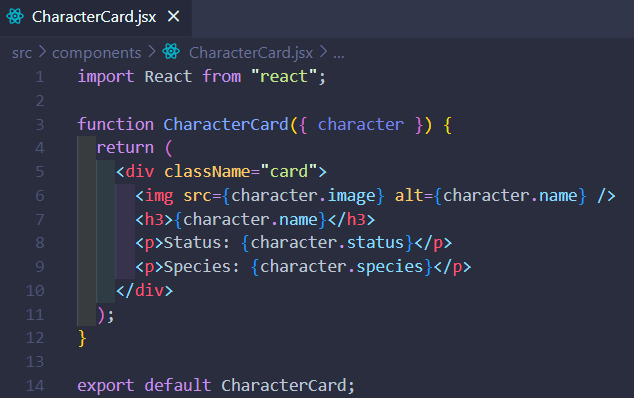
* Obtener los datos de la API de Rick and Morty mediante fetch.
* Almacenar los personajes en un estado.
* Aplicar filtros según el estado del personaje.
* Renderizar el componente CharacterCard para cada personaje.



### 3.3 Crear CharacterCard.jsx

Componente que muestra la información de un personaje en una tarjeta. Incluye:

* Imagen del personaje.
* Nombre.
* Estado (Vivo, Muerto, Desconocido).
* Especie.



### 3.4 Crear Filter.jsx

Este componente contiene un select para filtrar los personajes según su estado.

* Llama a una función cuando el usuario selecciona un filtro, actualizando los personajes mostrados.



### 3.5 Agregar Estilos en styles.css

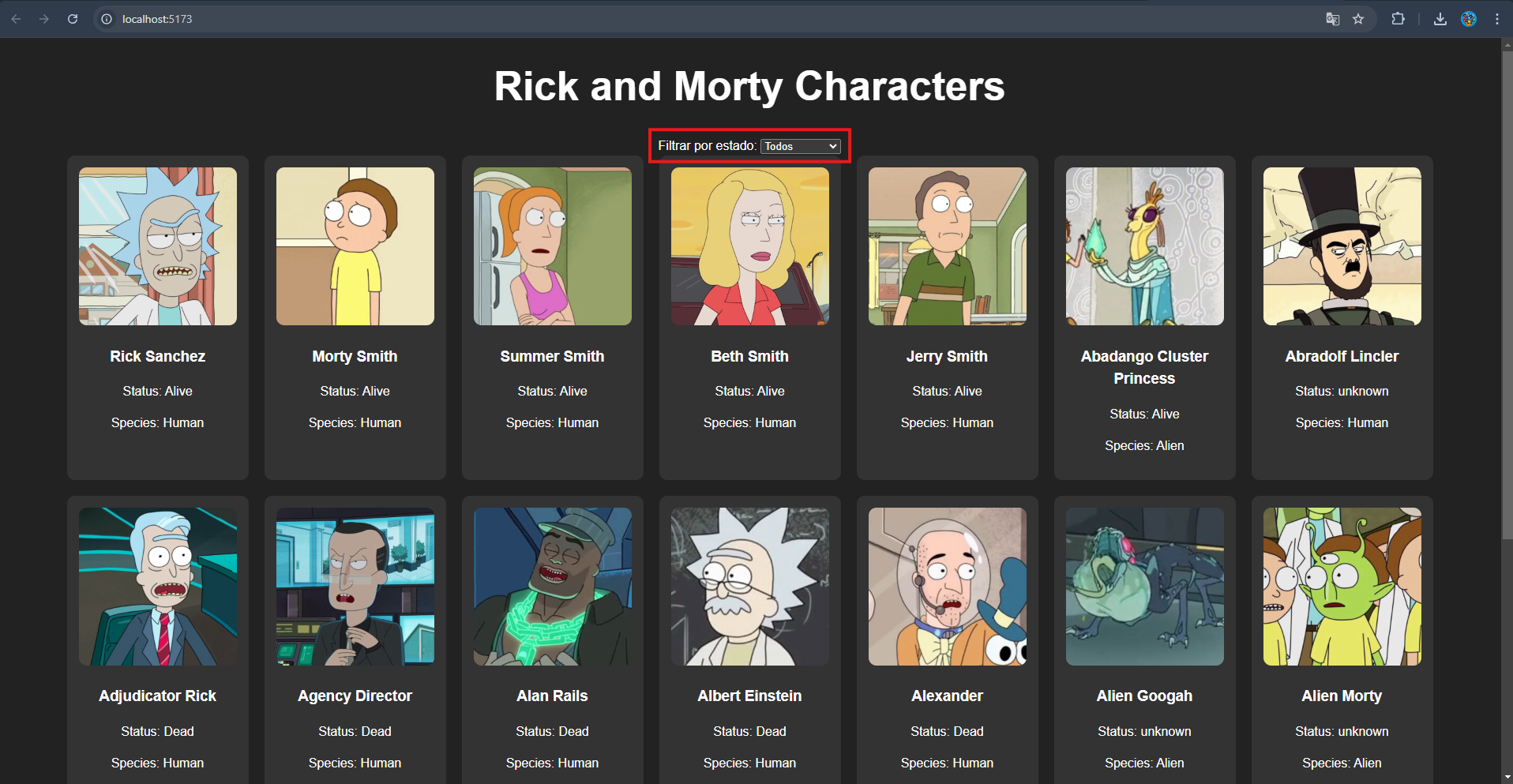
Archivo de estilos CSS que define:

* Estilos generales para la aplicación.
* La disposición en grid de los personajes.
* El diseño de las tarjetas de personaje.

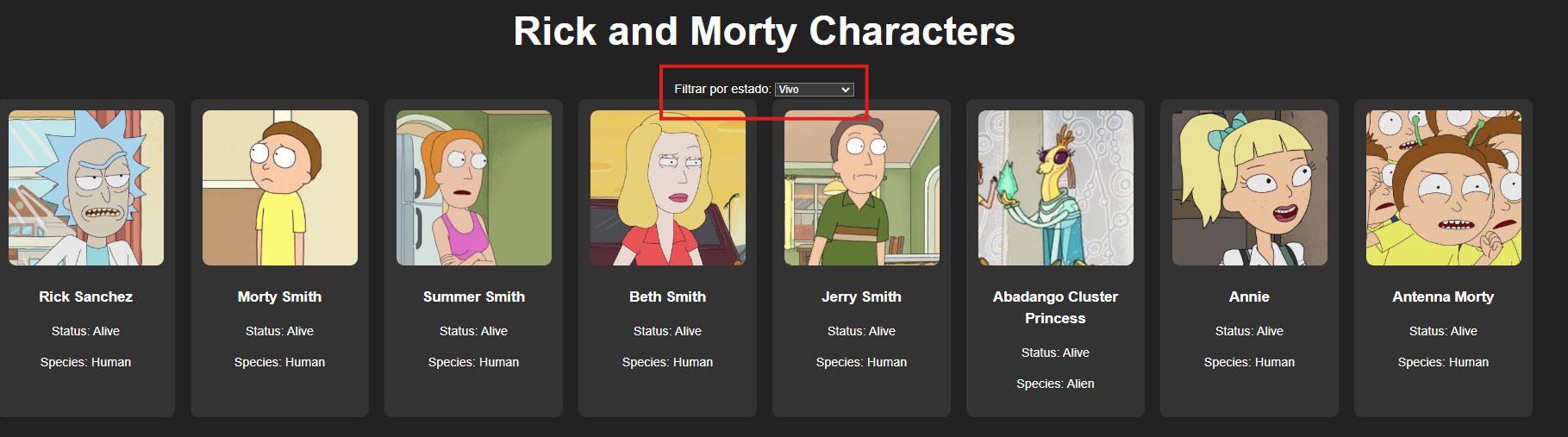
Además hay que importarlo en main.jsx, ya que es un style propio.

### 3.5 Resultado

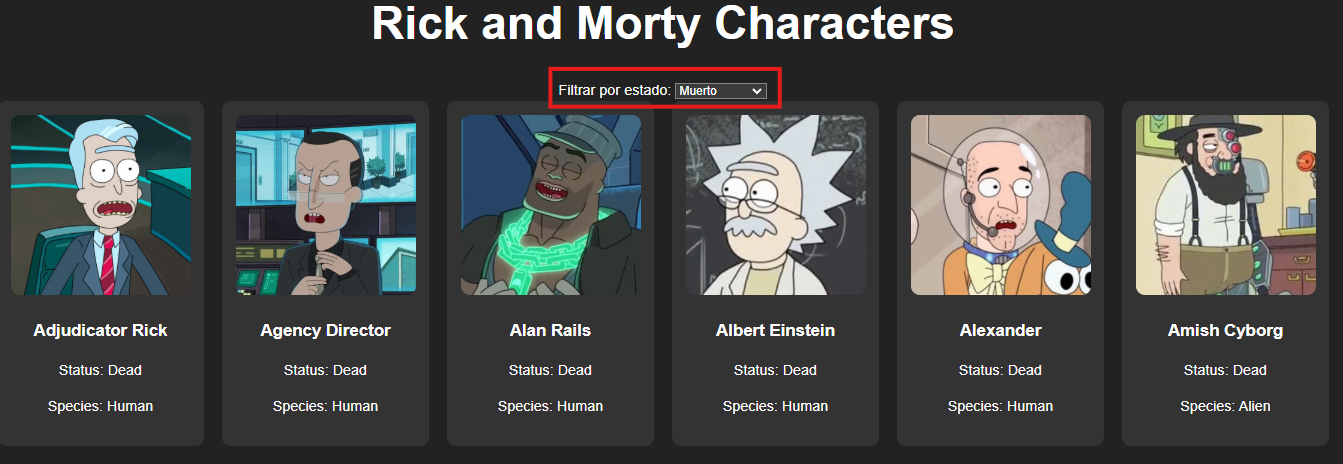
Todos:



Vivos:



Muertos:



Desconocido: