****

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN**

**Desarrollo de Aplicaciones Web NRC: 10522**

**Tema: Informe de Front-End Web Development**

**GRUPO 8**

**INTEGRANTES**

**Tacoaman Karen**

**Jiménez Brandon**

**Llumiquinga Luis**

**Lema Diego**

**Morales Daniela**

**Tutor: Msc. Pillajo Bolagay Carlos Andrés**

**Fecha: 22 de agosto de 2023**

[**1. INTRODUCCIÓN 3**](#_pzlyork7tzn2)

**2**[**. OBJETIVO 3**](#_pzlyork7tzn2)

**3**[**. DESARROLLO DE PROTOTIPO**](#_pzlyork7tzn2) 4

**Consumo de API Pública**

[1.1.](#_7lxi2bx3v095) **Programa 1: Consumo de API Pública**4

1.1.1. **Introducción al proyecto**4

1.1.2. **Objetivo**4

1.1.3. **Roles y Responsabilidades**5

1.1.4. **Diseño de Interfaces**6

1.1.5. **Implementación React y ventajas**12

1.1.6. **Consumo de API pública PLOS**14

1.1.7 **Validación y prueba del proyecto** 17

**Sesiones y Diseño** 18

2.1. **Programa 2: Desarrollo de Interfaces de Usuario para el Proyecto del Equipo** 21

2.2.**Introducción al proyecto** 21

2.2.**Objetivo** 21

2.3.**Diseño de la Interfaz** 21

2.4.**Consumo de URI** 25

2.5.**Integración de componentes**29

**4. CONCLUSIONES** 29

**5. CONCLUSIONES** 30

**6. BIBLIOGRAFÍA**30

# INTRODUCCIÓN

En el mundo actual de desarrollo web, la combinación de tecnologías avanzadas permite crear experiencias interactivas y dinámicas que cautivan a los usuarios. En este contexto, el presente proyecto se adentra en el Desarrollo Web de Front-End, enfocado en la utilización de la biblioteca React y su integración con API públicas. Este enfoque representa un hito en la evolución del diseño y funcionalidad de las aplicaciones web, al permitir la creación de interfaces atractivas y altamente responsivas.

Uno de los componentes esenciales de esta iniciativa fue la implementación de bibliotecas en React, que son herramientas predefinidas que facilitan la creación de componentes reutilizables y la gestión eficiente del estado de la aplicación. Estas bibliotecas no solo aceleran el desarrollo, sino que también fomentan la creación de aplicaciones más estructuradas y mantenibles.

Otro aspecto central de este proyecto fue el consumo de APIs públicas, en este caso, la API proporcionada por la Public Library of Science (PLOS). Esta API ofrece un flujo constante de datos en formato JSON, relacionados con publicaciones científicas. La integración de esta API en las aplicaciones web desarrolladas amplía las posibilidades de información y enriquece la experiencia del usuario, al permitir acceder a contenidos actualizados y relevantes.

El enfoque en el front-end utilizando React agrega una capa adicional de interactividad y usabilidad. La biblioteca React se destaca por su capacidad para construir interfaces de usuario altamente dinámicas, lo que brinda a los usuarios una sensación de fluidez y respuesta en tiempo real. La implementación de interfaces de usuario en React permite una experiencia más atractiva y moderna en comparación con los métodos tradicionales de desarrollo web.

Detallamos cómo se llevaron a cabo estos aspectos clave del proyecto, desde la planificación y diseño hasta la implementación y validación de las soluciones. Se exploran los desafíos enfrentados y las lecciones aprendidas en el camino, además de resaltar los resultados logrados al integrar bibliotecas en React, consumir APIs públicas y crear interfaces de usuario de front-end cautivadoras y funcionales.

# OBJETIVO

**General:**

Documentar y analizar en detalle el proceso de desarrollo, implementación y validación de un proyecto de Desarrollo Web de Front-End utilizando la biblioteca React y la integración de una API pública. Se busca proporcionar una visión completa de las etapas de diseño, programación, colaboración en equipo y validación de las soluciones implementadas, demostrando la capacidad para crear aplicaciones web funcionales y atractivas.

**Específico:**

* Detallar la configuración inicial del proyecto, incluyendo la creación del entorno de desarrollo, instalación de bibliotecas y configuración del repositorio de control de versiones.
* Diseñar interfaces de usuario que permitan a los usuarios ingresar términos de búsqueda y seleccionar opciones de visualización.

# DESARROLLO DE PROTOTIPO

**Repositorio Git**

<https://github.com/Brandon-A-Jimenez/Team8Helados>

# 

**Grabación:**

<https://youtu.be/P4jC0F4NXW0>

**Enlace de Código:**

**Program 1**

<https://drive.google.com/file/d/1V90bTVsDbBM3rToUQgZQx9lKAfukNisY/view?usp=drive_link>

**Programa 2**

<https://drive.google.com/file/d/1NlAVv40gYqkXvcwvqpd78jlWzdV-Eaal/view?usp=drive_link>

**Programa 1: Consumo de API Pública**

**1. Introducción al proyecto**

El presente programa detalla el proceso de desarrollo e implementación de un proyecto de Desarrollo Web de Front-End utilizando la biblioteca React y consumiendo una API pública. Este proyecto se llevó a cabo en equipos multidisciplinarios. Crearemos un sitio web funcional y visualmente atractivo que cumpla con los requisitos establecidos, aprovechando las capacidades de React y otras bibliotecas de Front-End.

Se describe en detalle la implementación de la funcionalidad de consumo de una API pública, específicamente la API de PLOS (Public Library of Science), para recuperar y presentar información sobre publicaciones científicas. Se exploran las etapas de diseño de interfaz, interacción con la API y visualización de datos, siguiendo patrones de diseño de interfaces de usuario.

**Objetivo Principal:**

Desarrollar una página web interactiva que permita a los usuarios realizar búsquedas y visualizar datos recuperados de una API pública de manera dinámica y amigable, brindando información detallada de artículos y documentos, así como datos de la computadora cliente que realiza las solicitudes.

**Objetivos Específicos:**

* Diseñar e implementar cinco páginas web distintas que se crean dinámicamente utilizando los datos obtenidos de la API pública, incluyendo un formulario de búsqueda que permita a los usuarios ingresar un término de búsqueda y seleccionar el modo de visualización de los datos.
* Mostrar los documentos tabulados de los artículos devueltos por la URI 1.
* Desarrollar una página adicional que muestre información sobre la computadora cliente, obtenida a través de una URI dada.

**Roles** **y responsabilidades**

*Tabla 1: Rol 1*

| Nombre | Luis Llumiquinga |
| --- | --- |
| Rol | Diseñador e Investigador |
| Categoría Profesional | Estudiante |
| Responsabilidad | Programación |
| Información de Contacto | lmllumiquinga3@espe.edu.ec |

*Tabla 2: Rol 2*

| Nombre | Karen Tacoaman |
| --- | --- |
| Rol | Diseñador e Investigador |
| Categoría Profesional | Estudiante |
| Responsabilidad | Programación |
| Información de Contacto | kitacoaman@espe.edu.ec |

*Tabla 3: Rol 3*

| Nombre | Daniela Morales |
| --- | --- |
| Rol | Diseñador e Investigador |
| Categoría Profesional | Estudiante |
| Responsabilidad | Programación |
| Información de Contacto | jdmorales@espe.edu.ec |

*Tabla 4: Rol 4*

| Nombre | Diego Lema |
| --- | --- |
| Rol | Diseñador e Investigador |
| Categoría Profesional | Estudiante |
| Responsabilidad | Programación |
| Información de Contacto | dflema3@espe.edu.ec |

*Tabla 5: Rol 5*

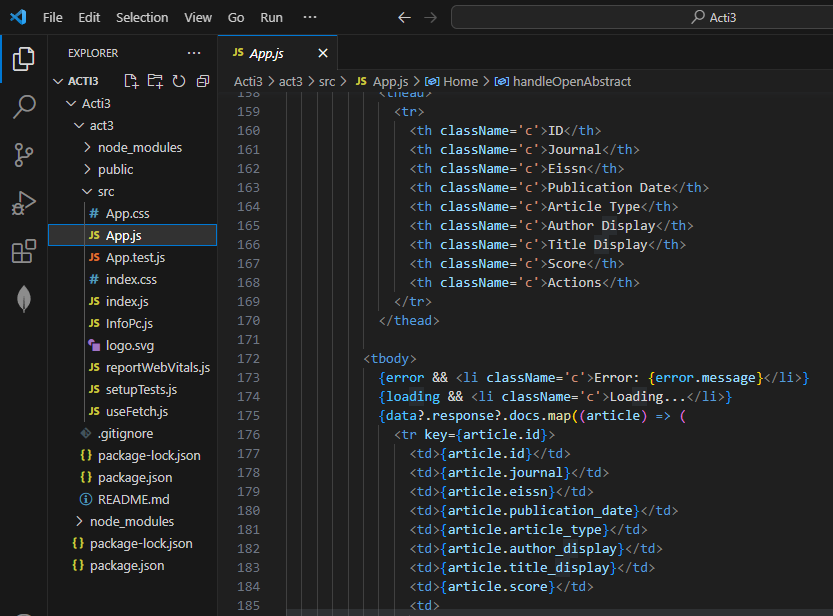
| Nombre | Brandon Jimenez |
| --- | --- |
| Rol | Diseñador e Investigador |
| Categoría Profesional | Estudiante |
| Responsabilidad | Programación |
| Información de Contacto | bajimenez6@espe.edu.ec |

2. **Diseño de interfaces**

El propósito del programa será el consumo de la API pública proporcionada, la cual, será consumida por nuestro sitio con un buscador y una tabla que representa la API con los siguientes datos:

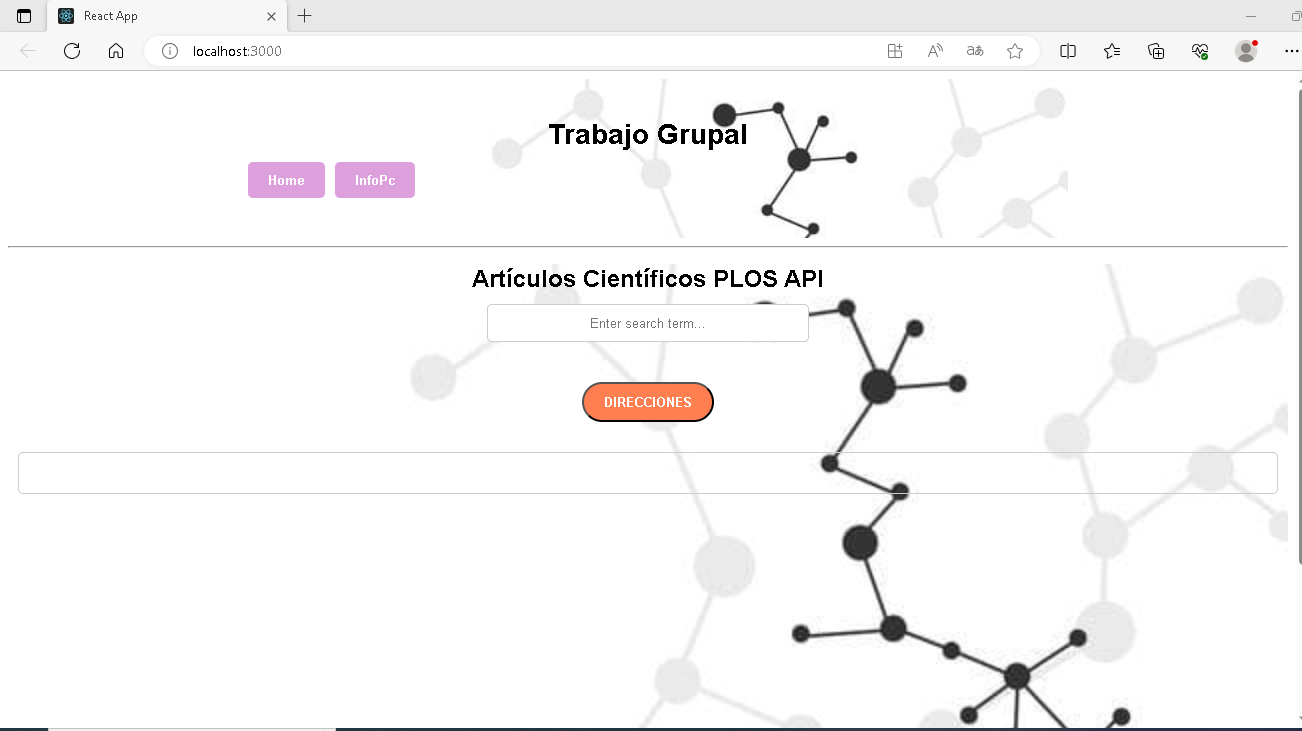
* id
* journal
* eissn
* publication\_date
* article\_type
* author\_display
* title\_display
* score

Se programara el sitio utilizando React y aprovechando su Doom podremos representar etiquetas HTML mediante su JSX, el cual, permitira trabajar mediante jerarquia o raíz, como podemos ver en el siguiente fragmento JS, el cual, retornara al Index.js la tabla de la página.

****

**Figura 1: Código de Tabla**

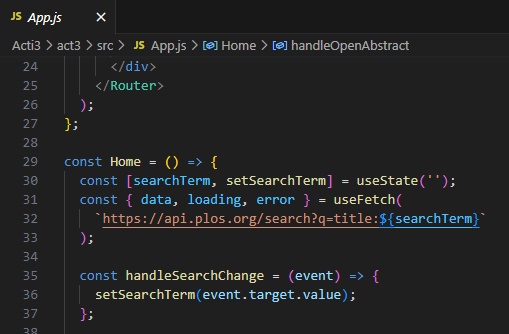
El sitio se verá de la siguiente forma



**Figura 2: Vista de Página inicial**

Al inicio no notaremos la información de la URI pero necesitará ingresar la información en el buscador.

Necesitaremos que la información JSON de la API sea utilizada por en nuestro sitio, por ello, necesitaremos recuperar en base a la descripción que se busca. Por ello, usaremos la siguiente línea para las búsquedas de los artículos.



**Figura 3: Obtención de API de búsqueda**

El código utiliza data, loading y error, los cuales han sido previamente especificados en el useFetch.js y aquí podemos ver claramente como se especifica un modelo para trabajar con las APIs del tipo json.



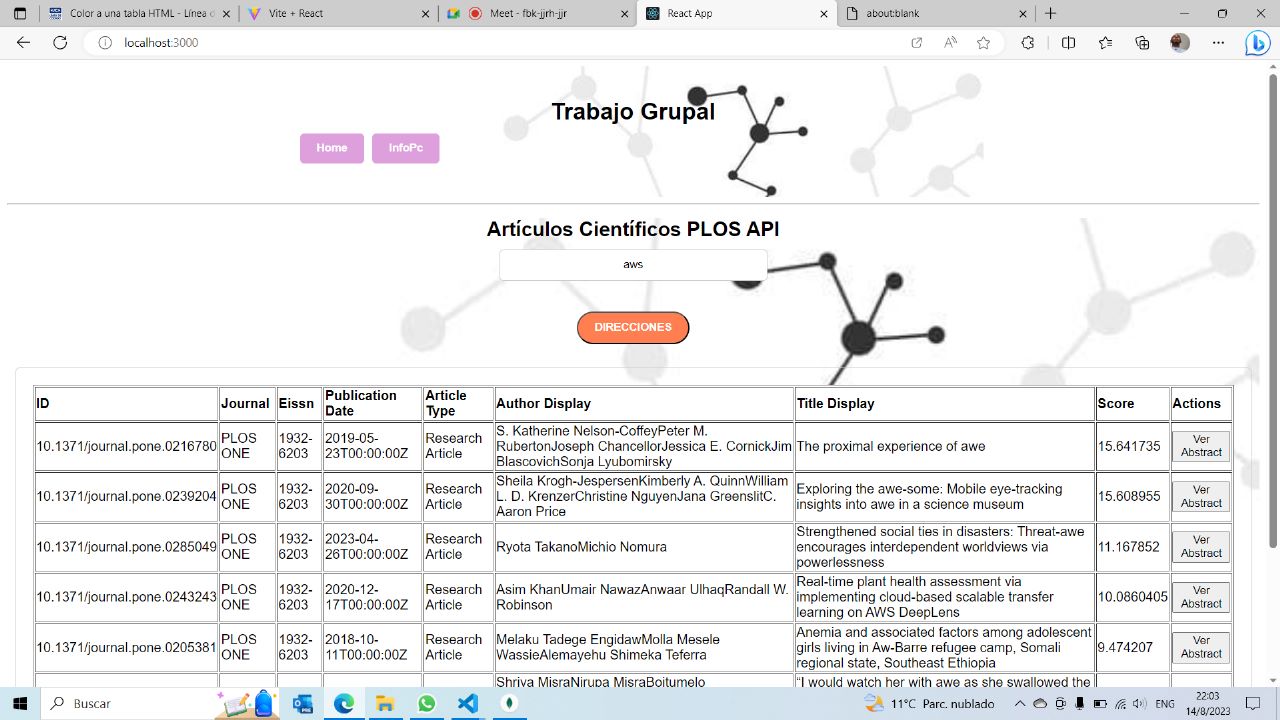
**Figura 4: Modelo para uso de API json**

Estos datos serán recuperados en la app.js debido al uso de Doom, lo cual, permite recuperar en base a la variable article y recuperando los datos para la tabla.



**Figura 5: Recuperación de datos de la API pública**

Así podremos ver el siguiente resultado de nuestro código al realizar búsquedas.

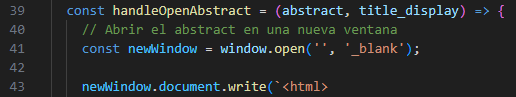


**Figura 6: Búsqueda de artículo**

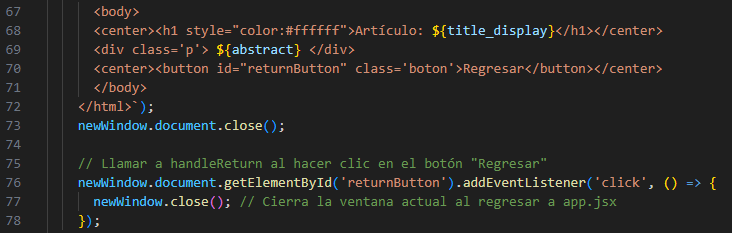
Cómo observamos la información respecto al buscar por spectral nos devuelve la siguiente información, la cual, tendrá los datos de la API y su página extraída.

Ahora probaremos el Abstract o resumen de los artículos

Podremos determinar el resultado de presionar el botón **Ver abstract**, lo cual son las siguientes líneas.

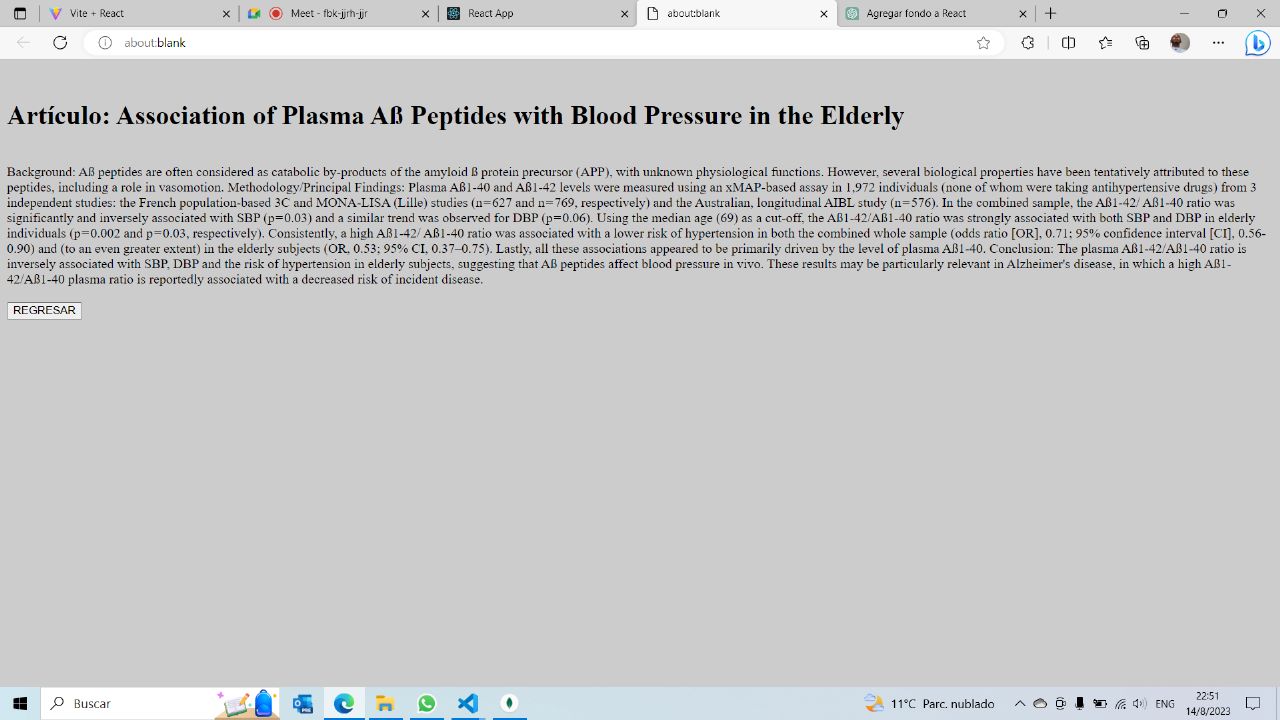


**Figura 7: Especificación de Abract Inicio**



**Figura 8: Recuperación de la información del articulo**

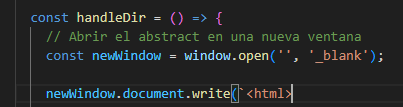
Podremos comprobar el resultado de la página al entrar al abstract.f



**Figura 9: Abstract del Artículo**

Como observamos nos da la información del resumen del artículo escogido. Al regresar, podremos ir a la página inicial y podremos ir a **direcciones,** las cuales, en base a la búsqueda nos dará los respectivos enlaces de búsqueda donde se extrajo.

Para ello, es similar al abstract solo con cierta diferencia que usamos las tablas para ordenar la información dada y enlistandose.

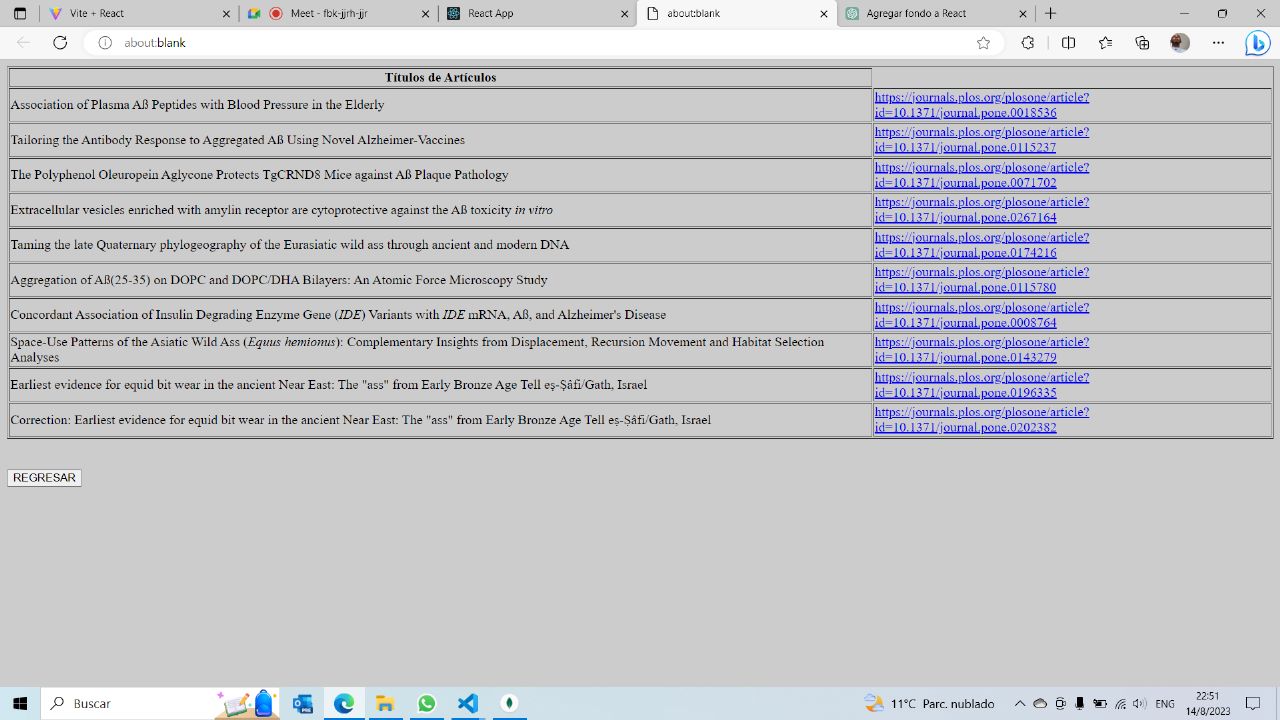


**Figura 10: Especificación de Dirección**

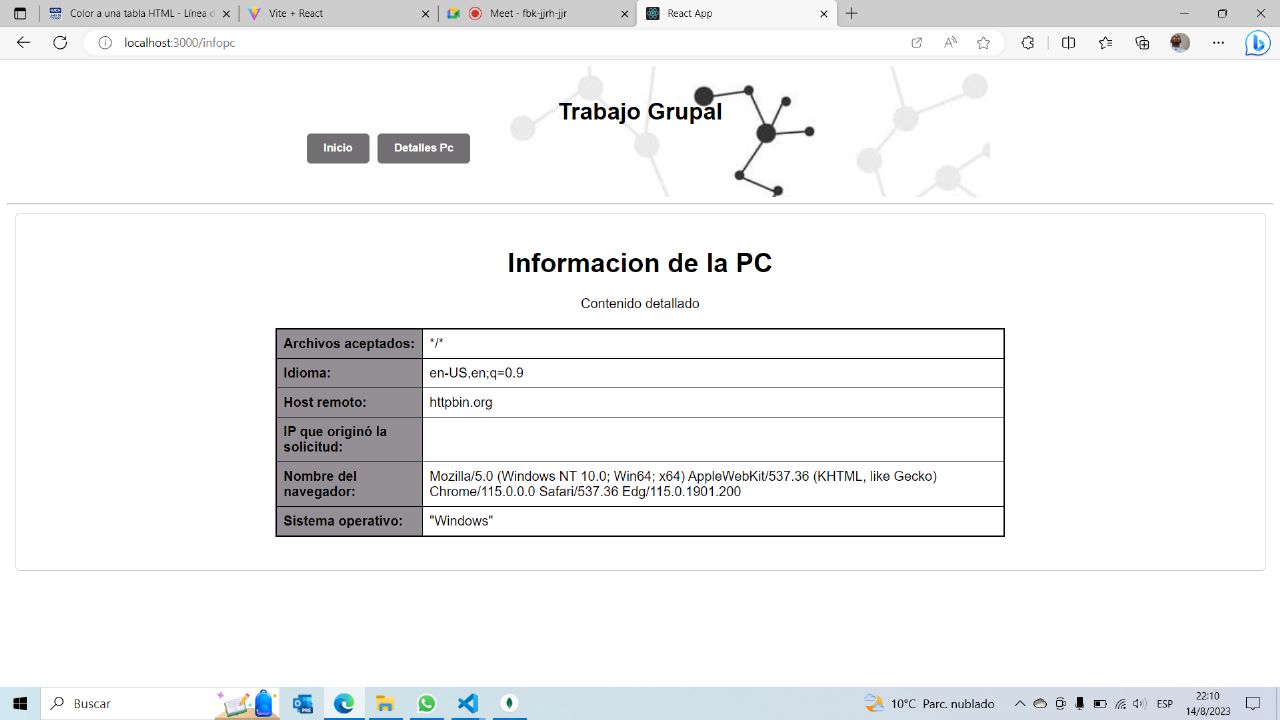


**Figura 11: Obtención de datos para la tabla**

Nos dará una tabla de los artículos y su enlace respectivo como observamos



**Figura 12: Visualización de los resultados de las búsquedas**



**Figura 13. InforPC.js**

3. **Implementación de React y Ventajas**

**React**

Hemos ocupado los siguientes comandos de instalación para nuestro código React

npm install react react-dom

npm install react react-dom react-router-dom

Con estos comandos nos permitirá trabajar con nuestro código y ayudarnos con las dependencias para solo necesitar correr con:

npm start

Pero ¿Por qué usar React?

React es uno de los Frameworks principales para el diseño de interfaces de una forma sencilla, debido a su reutilización de códigos y acelera la ejecución de código. Ocupa un sistema de arquitectura del tipo árbol de componentes, con lo cual, hablamos que trabaja con DOM que lo hace de forma dinámica y se basa en el uso de programas orientados a componentes.

Incorpora un DOM virtual que aplica los cambios de estados para el DOM permitiendo buenos niveles de rendimientos.

Su código ejecuta un modelo cliente-servidor permitiendo la optimización del sitio.

Al ser DOM hablamos de una jerarquía, por ello, si hay una actualización de un hijo deberá también validar el padre la actualización.

**Ventajas**

* Es rentable por ser multiplataforma
* Al ser reutilizable su código y entendible, acelera el proceso para el uso de componentes.
* Al usar JSX, su similitud a JavaScript es muy sencilla de comprender
* Uso de módulos para creación e integración para el uso de las APIs

**Patrones de diseño**

React por defecto incorpora una implementación MVVM y para nuestro código presente hemos usado los siguientes patrones que nos ofrece la librería React para trabajar de mejor manera nuestra página web.

**Composición**

Nos permitirá dividir componentes del código en partes del conocido “divide y vencerás”, al final cada parte se unirá para evitar ser complejo ya sea por funciones doom especificadas en el mismo archivo o en uno externo que se importará. Nos permitirá evitar el código complejo sobre todo los componentes y permite la reutilización por su fácil entendimiento y en el mantenimiento del código.

**Hooks**

Nos permitirá administrar los estados de los componentes y su ciclo de vida permitiendo la conexión con React algunos de los que usa son:

**useState**

Nos devolverá el estado actual y una función que lo actualiza permitiendo recordar el estado para su uso a futuro o modificación en un estado local, podríamos asemejar a una función o variable estática y más al primero pero como un componente de función.

**useEffect**

Nos permitirá manejar los ciclos de vida de los componentes, manejara las acciones del programa pero se centra más en llamada a API para obtener los datos y actualizar el estado de los componentes.

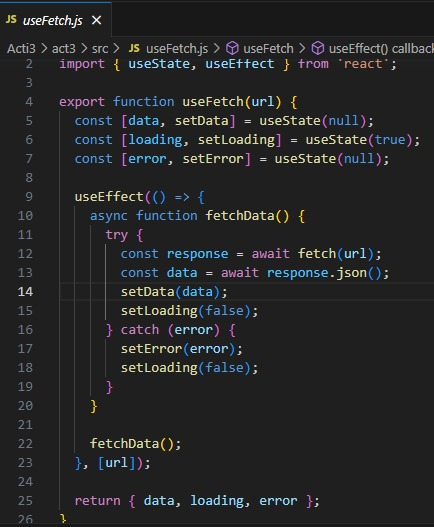
**Compound Components**

Permitirá llevar el control del comportamiento y renderizado de los componentes hijos por parte del padre y permitirá trabajar entre ellos, en nuestro caso para crear una tabla de artículos de la fuente de APIs públicas utilizadas para nuestra página.

**4. Consumo de la API pública de PLOS.**

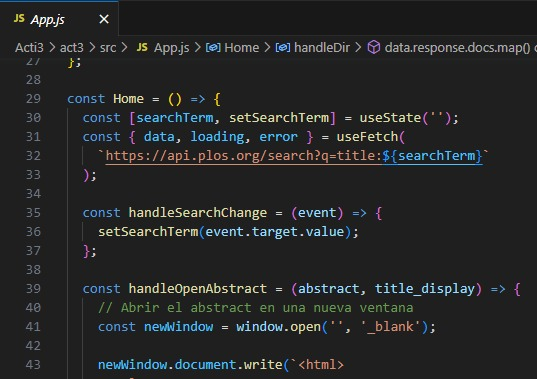
PLOS (Public Library of Science) es una organización que proporciona acceso gratuito a una gran cantidad de artículos científicos en línea por ende para acceder a su API pública y obtener datos de artículos científicos, debemos seguir algunos de estos pasos generales:

Dentro del código implementado primero deberemos establecer el React para establecer un modelo en UseFetch para las APIs restableciendo el useEffect por defecto figura 1.



*Figura.-1 useFetch.js*

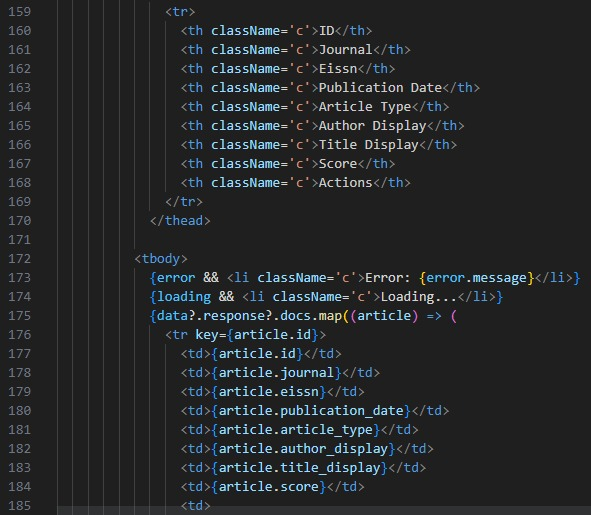
Por siguiente una vez estando en la file de App.js podemos observar en las siguientes líneas de codificación (figura.2) que toma la API y hace una búsqueda del navegador que proporciona cabe recalcar que el abstract se abre en una ventana nueva.



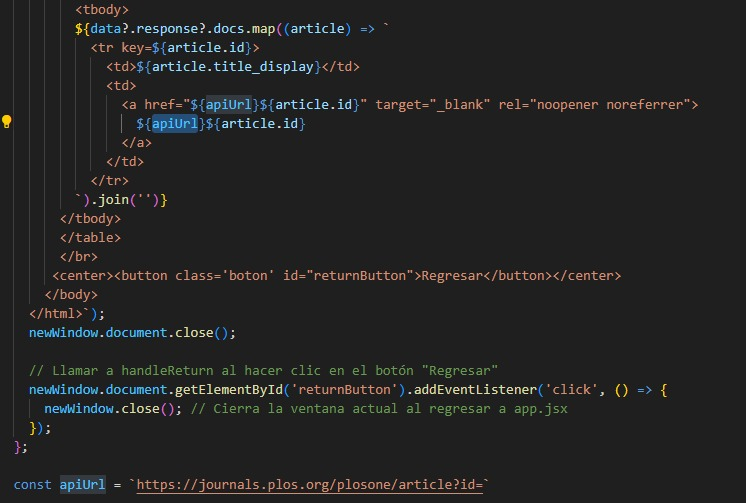
*Figura.-2 App.js*



*Figura.-2.1 App.js (API)*



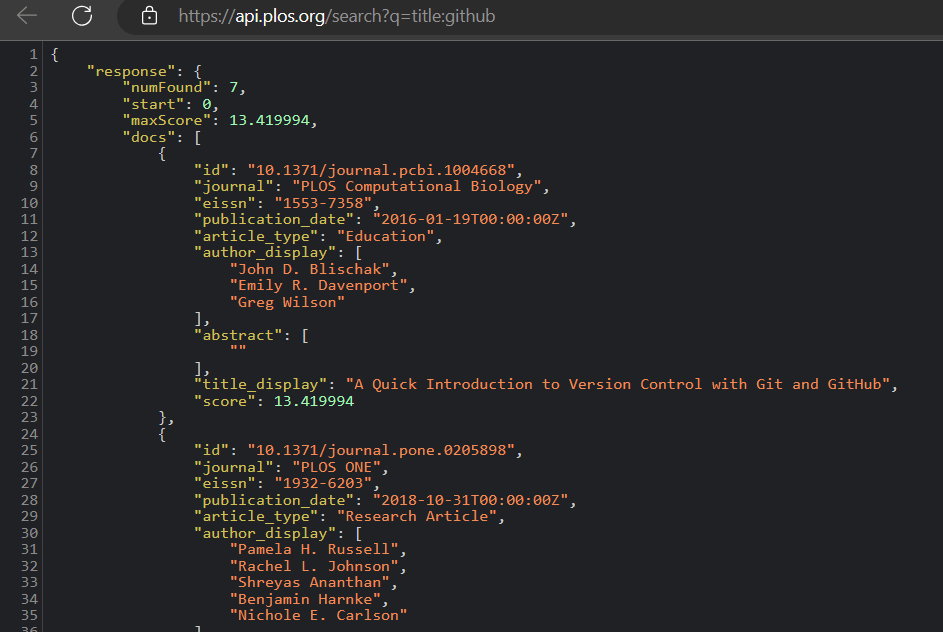
*Figura.-2.2 App.js (API - BODDY)*



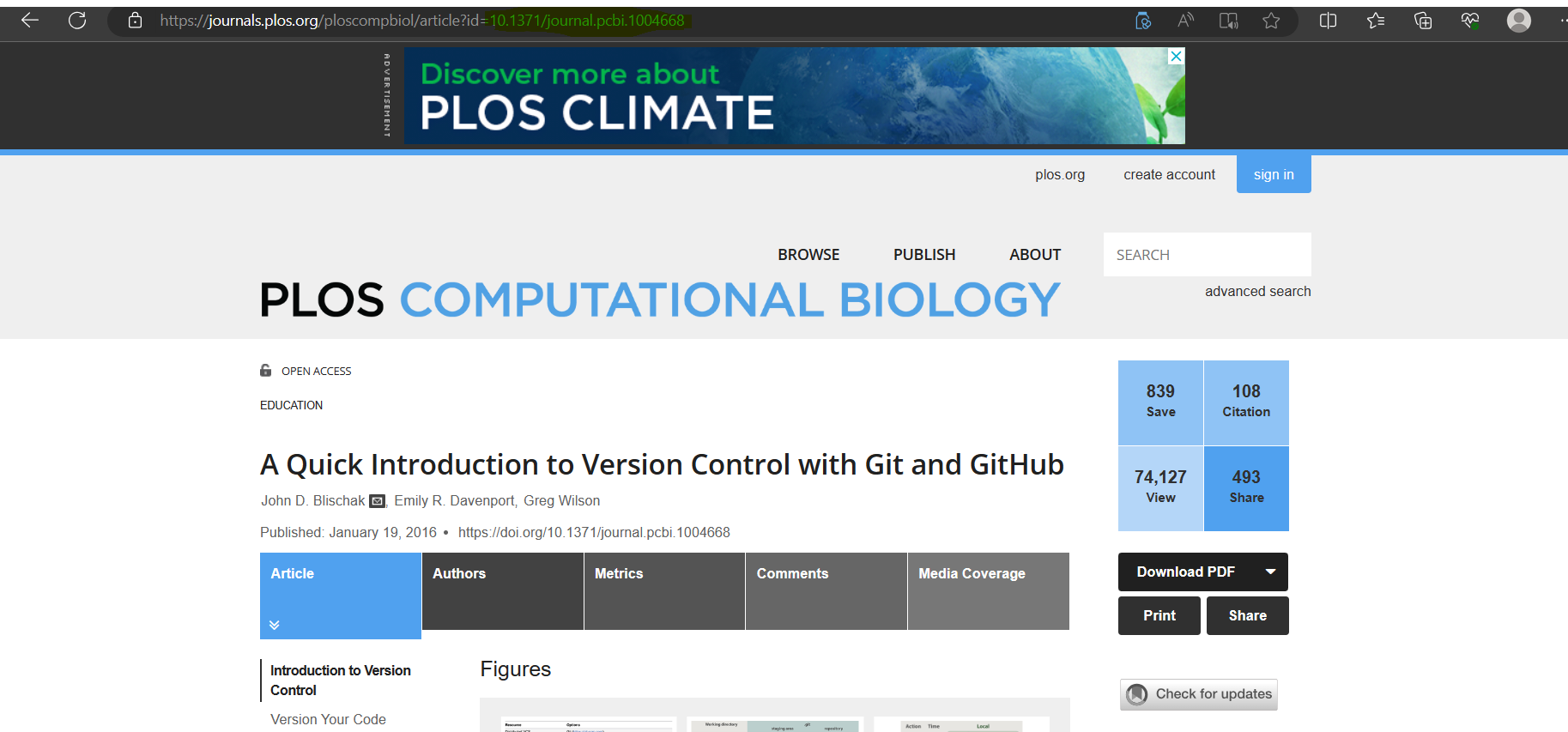
*Figura.-2.3 App.js (API - Direccionamiento)*

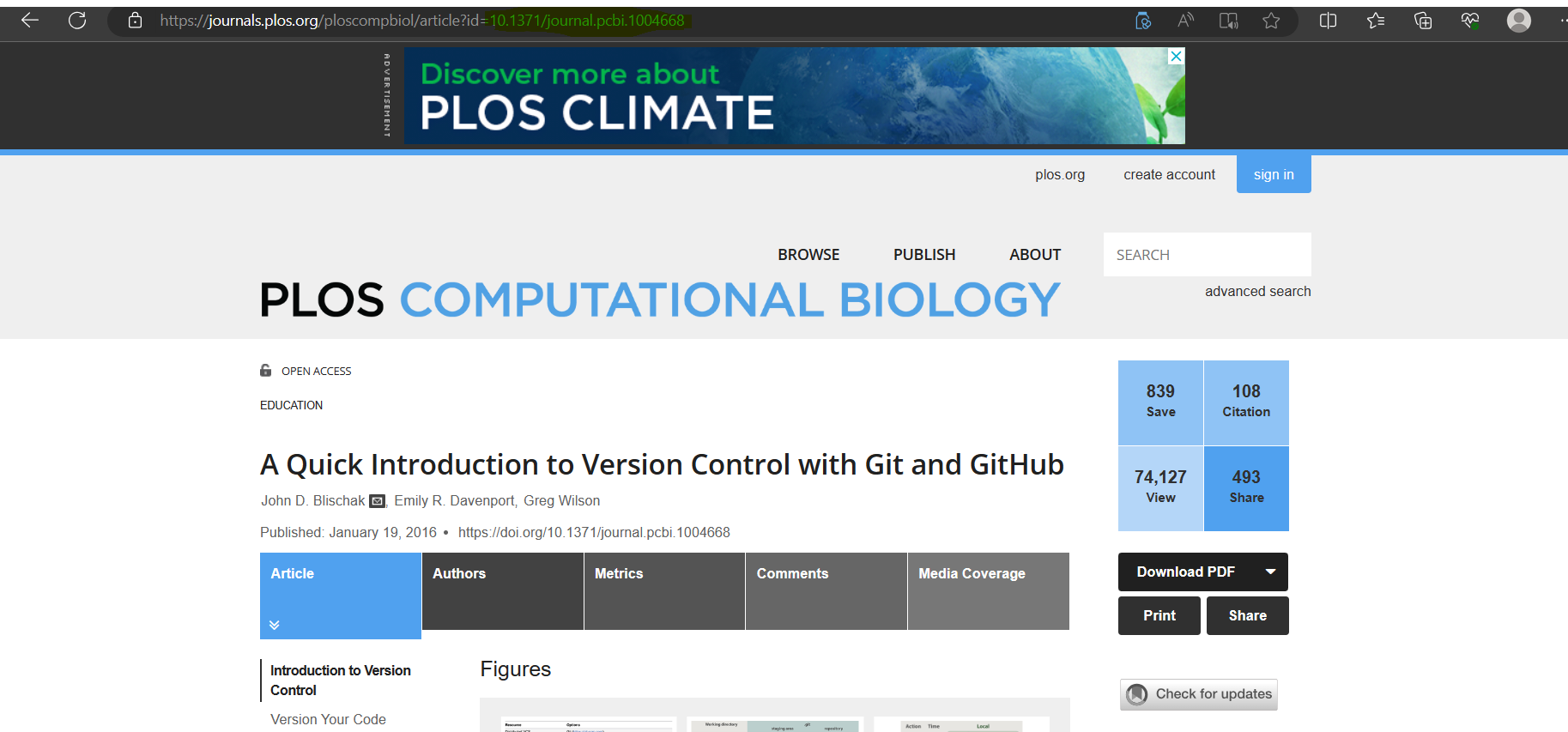
Siguiendo con el proceso con el siguiente link adjunto podemos visualizar que cada tabla hace uso y referencia de los resultados de las tablas, mediante la búsqueda, por ende, los artículos usan la api por id= "página del artículo" por ejemplo “id”:( id de la pagina del articulo a buscar) en este caso seria “id”:( “10.1371/journal.pcbi.1004668”) y de esta manera sucesivamente entre las id’s de las tablas como veremos a continuación en la figura 3.

<https://api.plos.org/search?q=title:github>



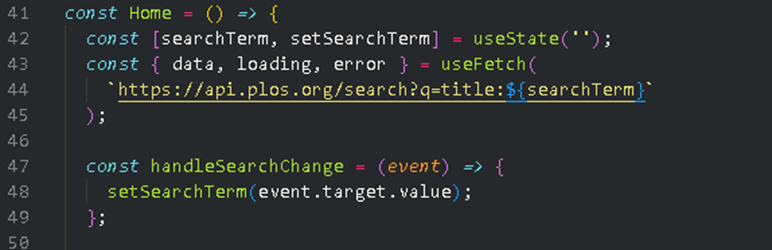
*Figura.-3 Tablas con sus id correspondientes a cada página de investigación.*

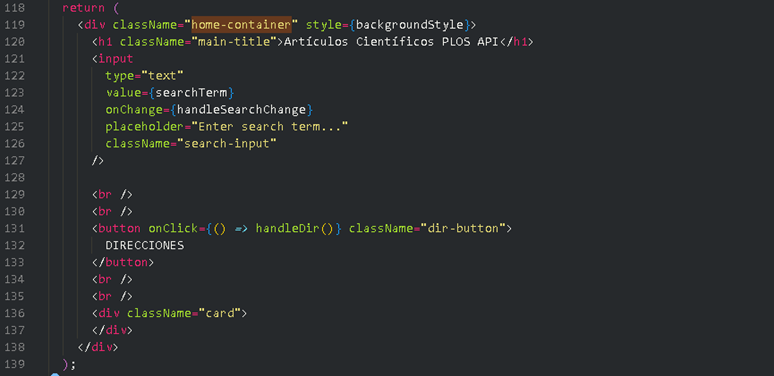
**

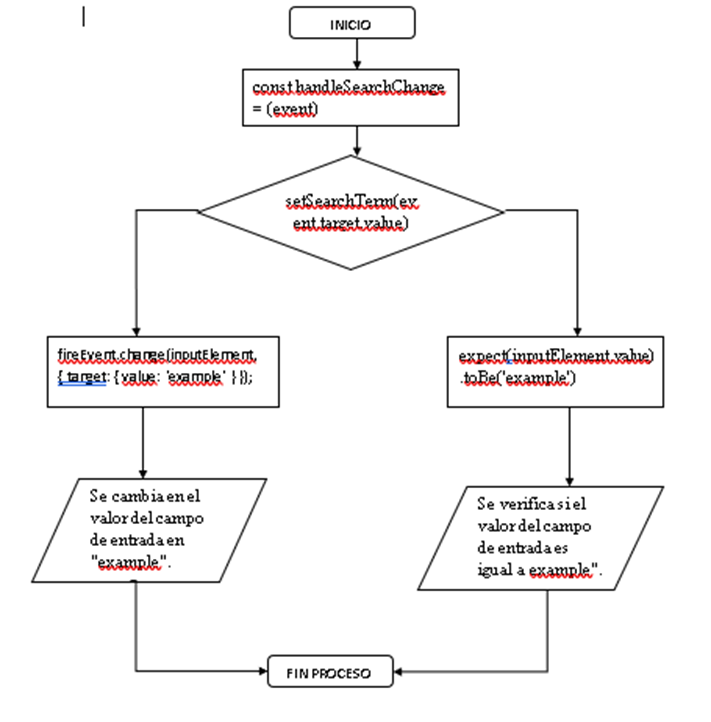
*Figura.-3 página de investigación redireccionada.*

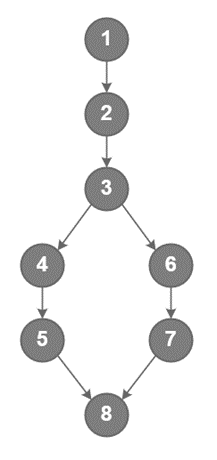
**5. Validación y prueba del proyecto.**

**Pruebas:**

**Caja Blanca**







# RUTAS:

R1: 1,2,3,6,7,8

R2: 1,2,3,4,5,8

**Prueba Caja Negra**

| **VARIABLE** | **CLASE DE EQUIVALENCIA** | **ESTADO** | **REPRESENTANTE** |
| --- | --- | --- | --- |
| **InfoPc** | const { data, loading, error } | **Válido** | **Loading…** |
| const { data, loading, error } | **No válido** | **—-----------------------** |
| **Error** | const { data, loading, error } | **Válido** | **Mensaje de error** |
| const { data, loading, error } | **No válido** | **—----------------------** |
| **Data** | const { data, loading, error } | **Válido** | **Muestra la tabla de datos** |
| const { data, loading, error } | **No válido** | **—-----------------------** |

**Programa 2: Desarrollo de Interfaces de Usuario para el Proyecto del Equipo**

**1. Introducción al proyecto**

El presente programa de nuestro proyecto se detalla de un funcionamiento de un Login que se va a desarrollar en diferentes carpetas vamos a crear un Frontend ya que esta es la parte más importante, vamos a utilizar la biblioteca React, con una base de datos Mongo DB y consumiendo una API pública. Este proyecto es un Login que el usuario debe registrarse con un nombre de usuario, correo y una contraseña, al crear su usuario puede realizar un Administrador de tareas que se puede crear, editar, eliminar tareas, es visualmente atractivo y muy formal.

Se describe en detalle una creación de una base de datos Mongodb ya que aprovecharemos esta base de datos para guardar los datos, también un CRUD de tareas para crear, añadir, recuperar y eliminar las tareas establecidas, por ultimo tenemos rutas protegidas para el Login no acceda a otras páginas sin preguntar.

**Objetivo Principal:**

Desarrollar un Login que el usuario pueda registrarse, al registrarse el usuario debe ingresar con un correo y una contraseña, los usuarios puedan acceder al Administrador de tareas en el cual puede crear, editar y eliminar tareas, también registrar con sus respectivas fechas las tareas realizadas.

**Objetivos Específicos:**

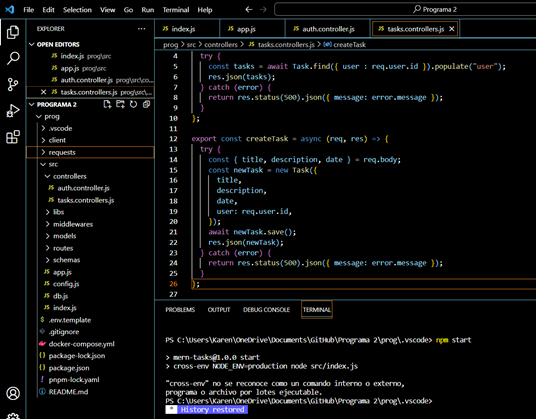
* Diseñar un Login para que el usuario pueda registrarse con un nombre, correo y contraseña.
* Al ingrese a su cuenta el usuario y pueda crear tareas con un título, fecha y una descripción.
* Mostrar las tareas realizadas que se han guardado y si el usuario desea editar, eliminar lo podrá hacer.
* Mostrar en pantalla todas las tareas que se han realizado y también que tenga un control de búsqueda si en el caso quiere buscar su tarea por un título.

**2. Diseño de las interfaces**

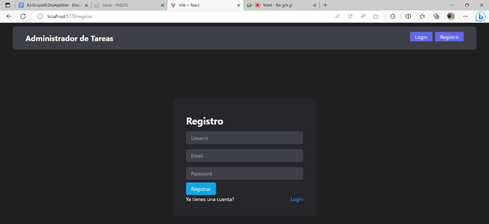
El programa realizado será el consumo de la API que se puede visualizar en nuestro programa es:

* El nombre del usuario que se registra.
* Añadir tarea
* Salir.

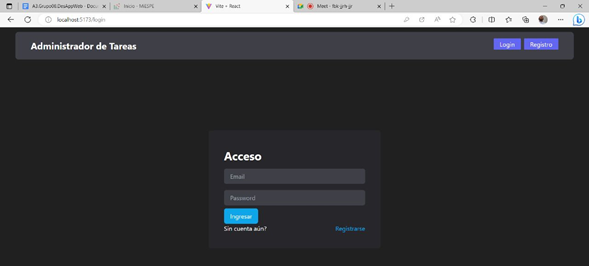
El administrador de tareas se representa por un index.js y su respectivo HTML que mediante su JSX se podrá ingresar con su correo y contraseña al Administrador de tareas se podrá visualizar el nombre del usuario, el botón para crear tarear y salir.

*Ilustración 1 Código del Administrador de tareas. Autor: Propio.*

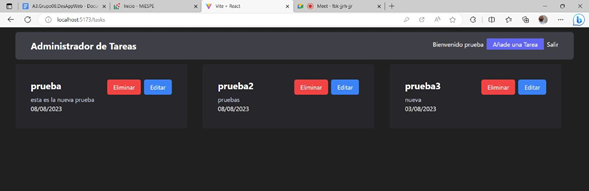
El Registro de usuario de visualizar de la siguiente manera:

*Ilustración 2 Registro de usuario. Autor: Propio.*

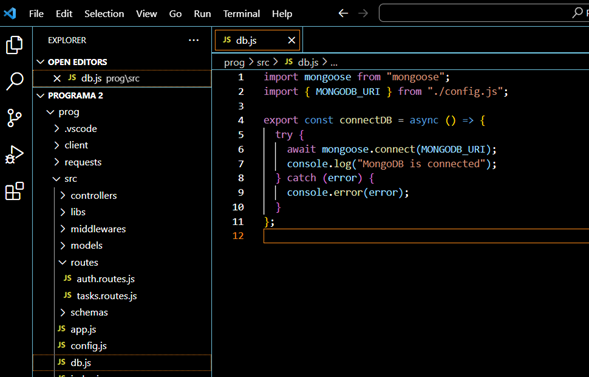
Login del usuario, el acceso para la Administración de tareas es:

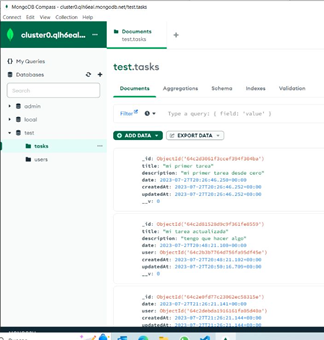
*Ilustración 3. Login del usuario. Autor: Propio*

El administrador de tarea se visualización de la siguiente manera con tareas registradas:

*Ilustración 4. Visualización de Administración de tareas. Autor: Propio*

Para el registro de la Administración de tareas se podrá crear las tareas y se guardará en una base de datos que es mongodb.

*Ilustración 5. Base de datos. Autor: Propio.*

**

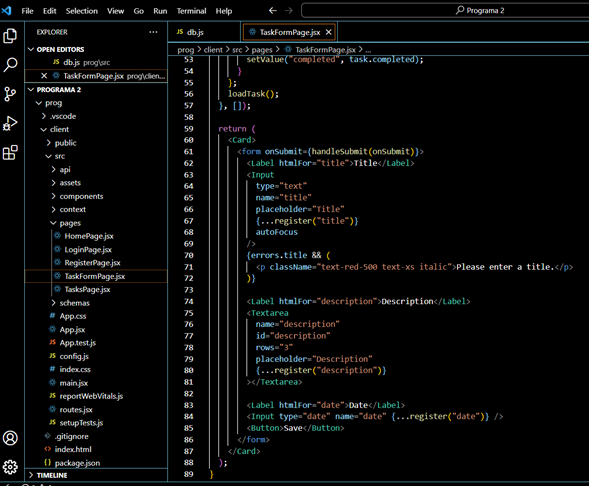
*Ilustración 6. MongoDB. Autor: Propio*

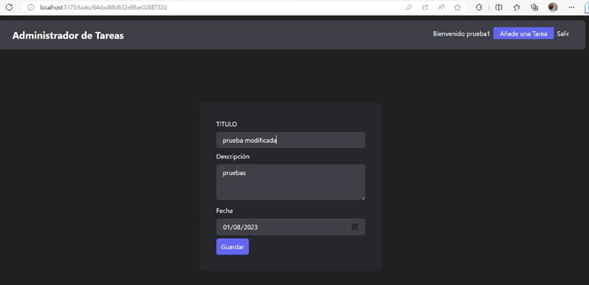
**3. Consumo de información de URI**

Para el consumo de la URI es la abreviatura de Uniform Resource Identifier, en español identificador uniforme de recursos, se utiliza para definir la identidad de un objeto, independientemente del método utilizado.

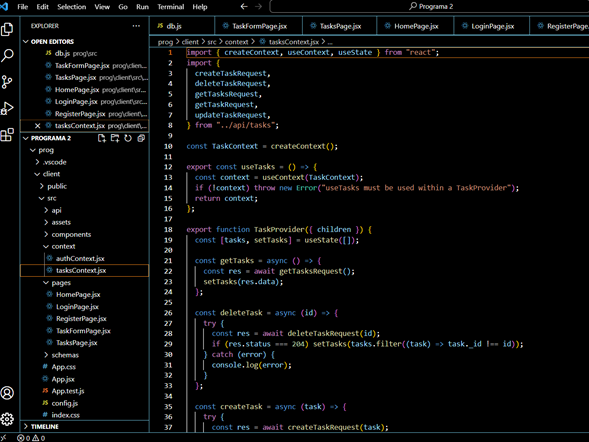
En nuestro proyecto implementamos el gestor de tareas, dentro del código implementado podemos notar las siguientes tareas:

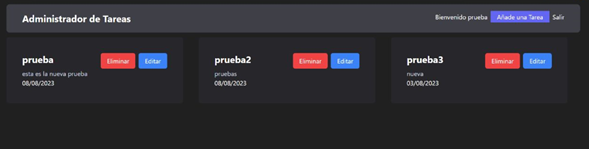
* Crear tareas.

*Ilustración 7. Código Crear tarea. Autor: Propio*

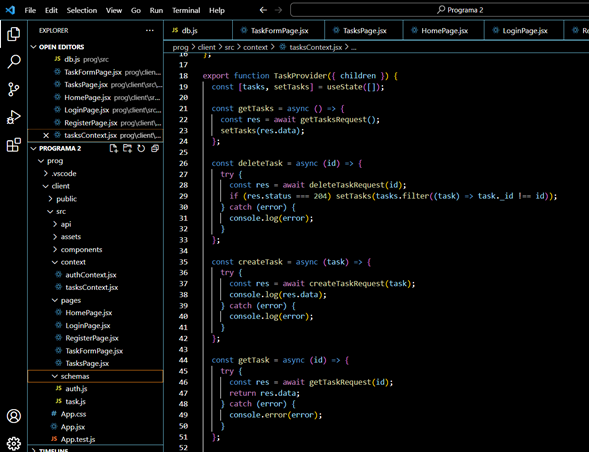
*Ilustración 8. URI Crear tareas. Autor: Propio*

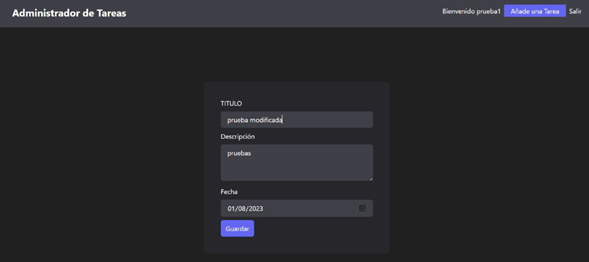
* Visualizar tareas.

*Ilustración 9. Código de la administración de tareas. Autor: Propio*

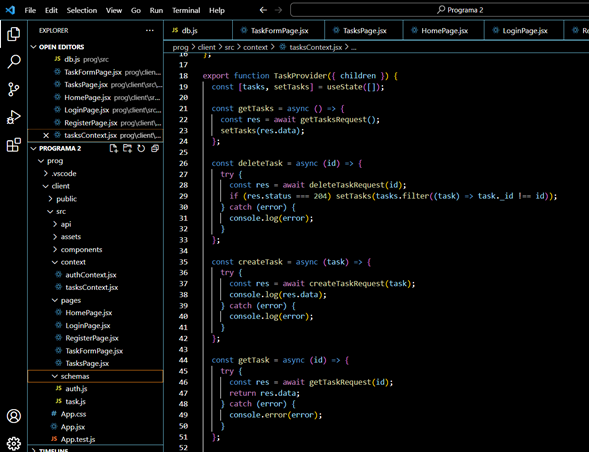
*Ilustración 10. URI Visualizar tareas. Autor: Propio*

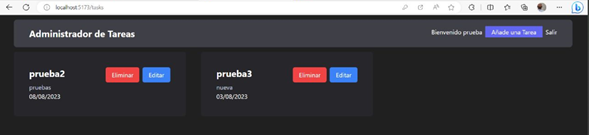
* Editar tarea

*Ilustración 11. Código de editar tarea. Autor: Propio*

*Ilustración 12 URI Editar tarea. Autor: Propio*

* Eliminar tarea.

*Ilustración 13. Código de eliminar tarea. Autor: Propio*

*Ilustración 14. URI Eliminar tarea. Autor: Propio.*

**4. Integración de componentes**

Dentro del diseño de interfaces de usuario, la integración de componentes se refiere al proceso de combinar y organizar diferentes elementos de la interfaz para crear una experiencia de usuario coherente y funcional. Estos componentes pueden incluir elementos visuales, como botones, campos de entrada, iconos, imágenes y elementos interactivos, así como también componentes funcionales, como menús desplegables, paneles de navegación y áreas de contenido.

Dentro de nuestro proyecto se manejaron algunos componentes de los cuáles se mencionan a continuación:

**Coherencia visual:** Nuestra aplicación muestra un estilo coherente en toda la interfaz para una experiencia de usuario unificada.

**Facilitar la interacción:** Nuestra aplicación está diseñada de una manera accesible y fácil para que los usuarios puedan ingresar a nuestro aplicativo y puedan interactuar de una manera fácil y entendible.

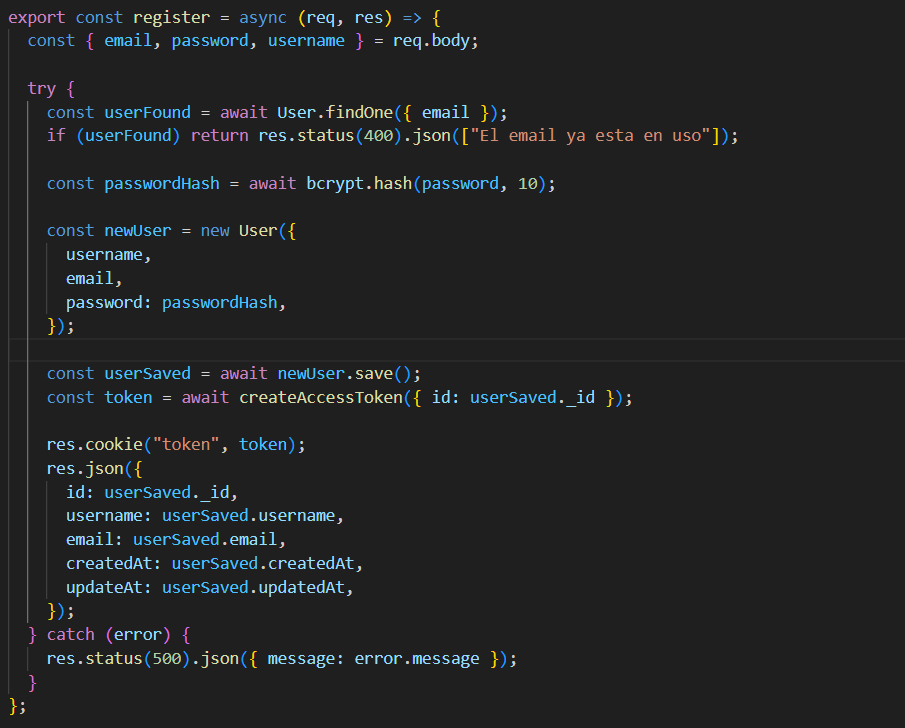
**Optimización del rendimiento:** Dentro de nuestro aplicativo la carga rápida y la eficiencia en la interacción de la misma ayuda a un mejor rendimiento de nuestra interfaz de usuario.

**5. Validación y prueba del proyecto.**

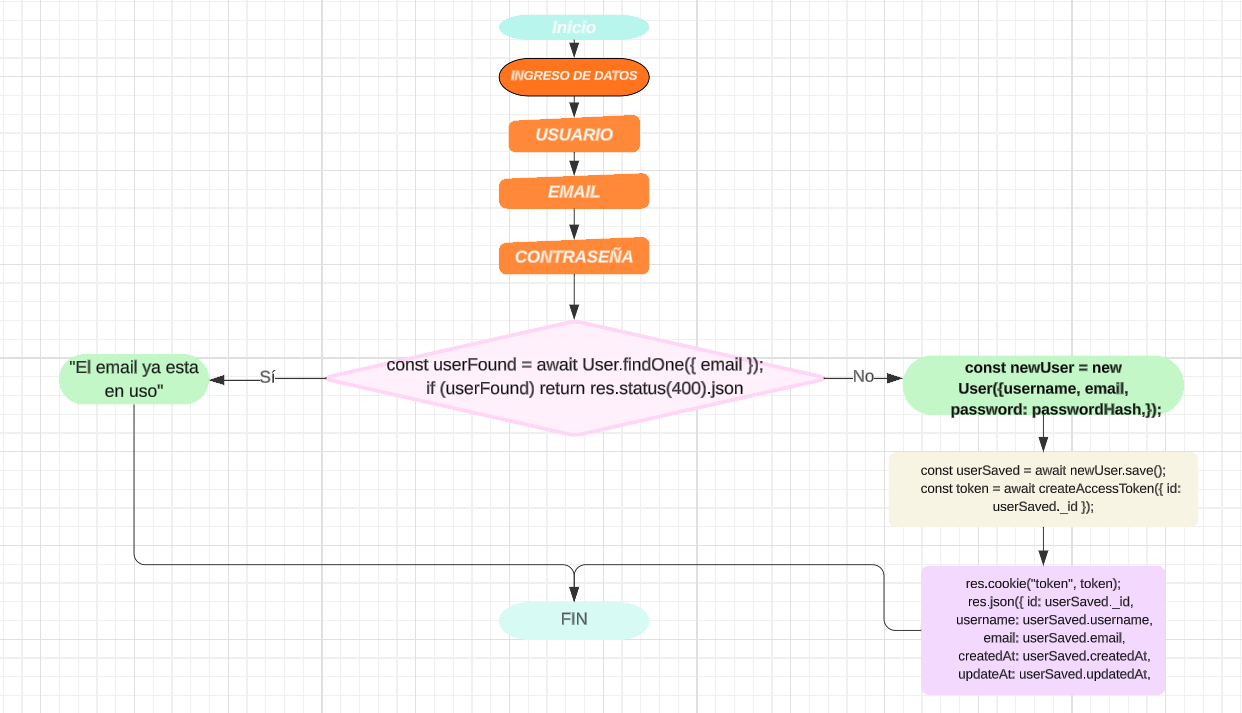
**Pruebas:**

**Caja Blanca**

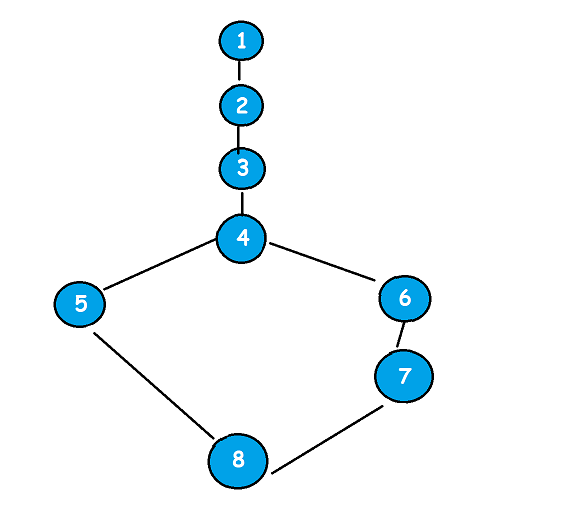
Prueba caja blanca con la respectiva validación de Registro con la variable principal para este ejemplo “email”

****

*Figura 1.- validación de Registro email.*

****

*Figura 2.- Algoritmo validación de Registro con email.*

**

*Figura 3.- Diagrama de flujo validación de Registro con email.*

# RUTAS:

R1: 1,2,3,4,5,8

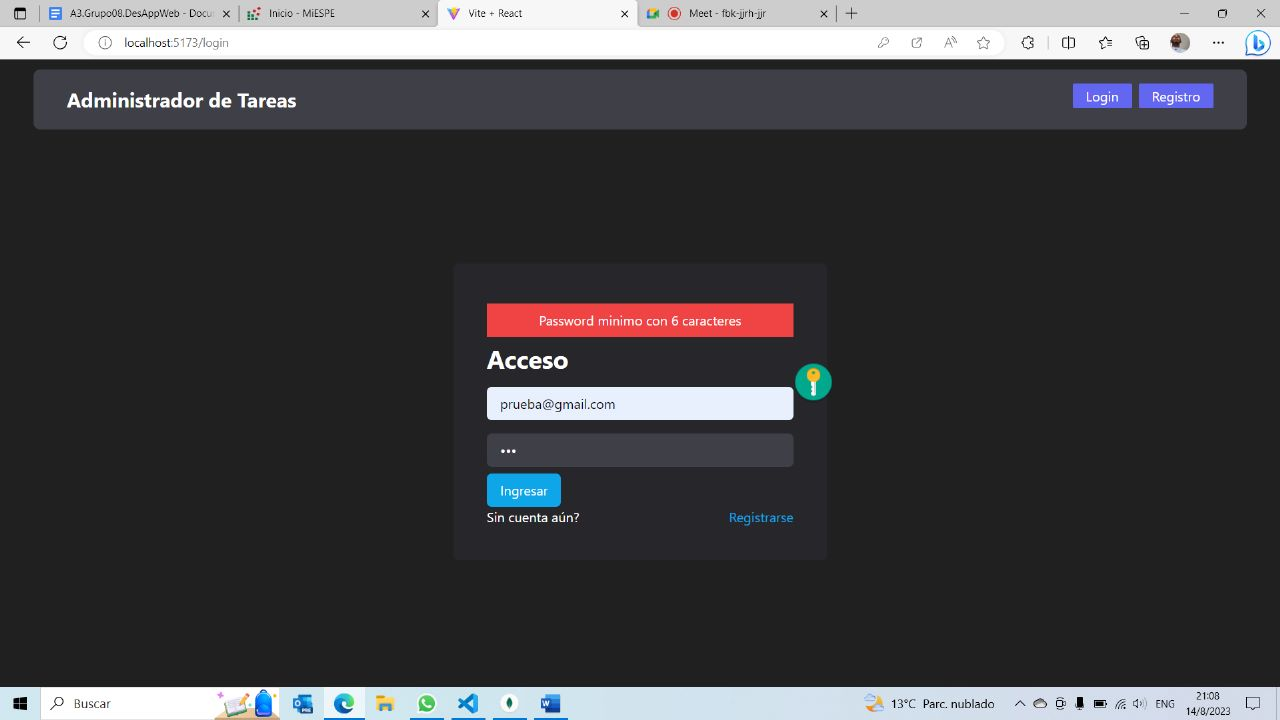
R2: 1,2,3,4,6,7,8

**Caja Negra**

| **VARIABLE** | **CLASE DE EQUIVALENCIA** | **ESTADO** | **REPRESENTANTE** |
| --- | --- | --- | --- |
| **Password** | **password: z**  **.string({**  **required\_error: "Password es requerido",**  **})** | **Válido** | **LuisLL** |
| **password: z**  **.string({**  **required\_error: "Password es requerido",**  **})**  **.min(6, {**  **message: 'Password minimo con 6 caracteres',**  **}),**  **})** | **No válido** | **Luis1234** |
| **Registro** | const { data, loading } | **Válido** | **Usuario: Test12**  **Email:** [**tes@gmail.com**](mailto:tes@gmail.com)  **Password: 123456** |
| **export const loginSchema = z.object({**  **email: z**  **.string({**  **required\_error: 'Email es requerido',**  **})**  **.email({**  **message: 'Email invalido'**  **}),**  **password: z**  **.string({**  **required\_error: 'Password es requerido'**  **})**  **.min(6, {**  **message: 'Password minimo con 6 caracteres'**  **}),**  **{errors.password && (**  **<p className="text-red-500">Password es requerido</p>**  **)}** | **No válido** | **Usuario: Test123**  **Email:** [**tes@gmail.co**](mailto:tes@gmail.com)**n**  **Password: 12345667** |

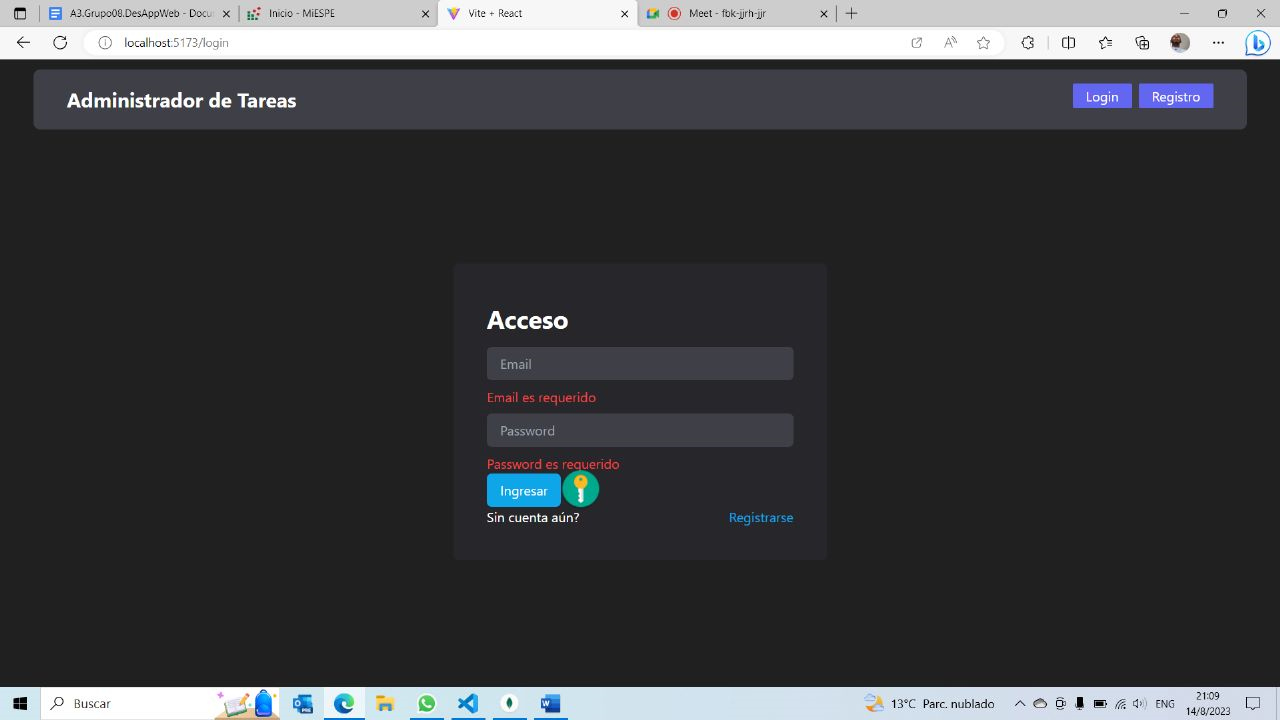
**Validación de Campos Incorrectos**

En la figura 2 podemos observar la validación de campo incorrecto por medio de nuestra contraseña ya que al no contener la misma un de mínimo de 6 caracteres nos validara un error en la cual nos saltará el siguiente mensaje “ Password mínimo con 6 caracteres”



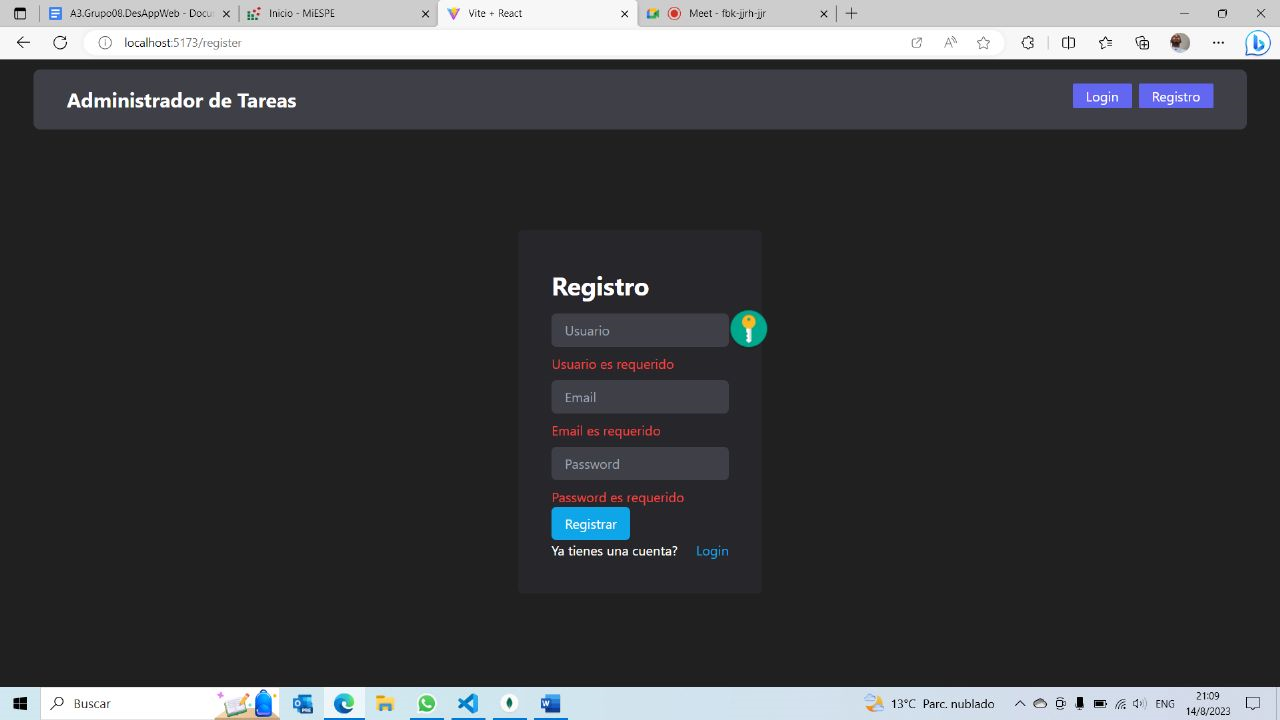
*Figura 2.- Validación de Password mínimo 6 caracteres.*

Figura 3 y 4 en las siguientes figuras podemos observar una validación de campos incorrectos al no digitalizar ningun caracter ya que es necesario colocar tanto el correo electrónico como la contraseña al no digitalizar ningun caracter nos validara de campo incorrecto con los siguientes mensajes “Email es requerido” , “Password es requerido”



*Figura 3.- Validación de Acceso (login) .*

Lo mismo sucede en esta figura al no contener los caracteres digitalizados nos validara como campos erróneos con los siguientes mensajes “Usuario es requerido” , “Email es requerido” y “Password es requerido” como lo veremos a continuación.



*Figura 4.- Figura 3.- Validación de Registro (login) .*

# CONCLUSIONES

* Se realizan diferentes integraciones de componentes para conocer que el aplicativo que estamos realizando cumpla con cada uno de los requerimientos y que la experiencia del usuario sea única, accesible y fácil de utilizar.
* Las pruebas de caja blanca y caja negra son una práctica fundamental en el desarrollo de software. Ayudan a aumentar la confiabilidad y calidad del código, lo que a su vez contribuye a una mejor experiencia del usuario y a un mantenimiento más eficiente del software a lo largo del tiempo.

1. **Recomendaciones**

* Se debe tomar en cuenta que al realizar cambios en el código, nos debemos asegurar de ajustar y agregar pruebas según sea necesario. Las pruebas son una parte importante del proceso de refactorización.
* Se recomienda mantener la interfaz simple y fácil de entender y utilizar un diseño limpio, una tipografía legible y colores coherentes para asegurarte de que los usuarios puedan navegar sin esfuerzo.

# 

# BIBLIOGRAFÍA

CloudAPPi. (2022, julio 4). Patrones de diseño: Composición en React. CloudAPPi. https://cloudappi.net/patrones-de-diseno-composicion-en-react/

¿Conoces React y sus beneficios? Si no es así, ¡te lo explicamos! (2022, mayo 10). Qindel: Consultoría IT; QINDEL. https://www.qindel.com/conoces-react-y-sus-beneficios-si-no-es-asi-te-lo-explicamos/

Doonamis. (2021, octubre 4). Para qué sirve React JS: beneficios y ejemplos. Doonamis. https://www.doonamis.es/para-que-sirve-react-js-beneficios-para-tus-apps/

LAS 10 VENTAJAS PRINCIPALES DE USAR REACT.JS. (2023, julio 18). Sistemasgeniales.com - Páginas web, software y redes sociales -. https://sistemasgeniales.com/2023/07/las-10-ventajas-principales-de-usar-react-js/

Usando el Hook de efecto. (s/f). Reactjs.org. Recuperado el 12 de agosto de 2023, de https://es.legacy.reactjs.org/docs/hooks-effect.html

Usando el Hook de estado. (s/f). Reactjs.org. Recuperado el 12 de agosto de 2023, de https://es.legacy.reactjs.org/docs/hooks-state.html