



**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

Llumiyinga Luis

Lema Diego

Morales Daniela

Tutor: Msc. Pillajo Bolagay Carlos Andrés

Fecha: 22 de agosto de 2023

1. INTRODUCCIÓN	3
2. OBJETIVO	3
3. DESARROLLO DE PROTOTIPO	4
Consumo de API Pública	
1.1. 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

1. 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.

2. 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.

3. 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/1V90bTVsDbBM3rToUQgZQx9IKAfukNisY/view?usp=drive_link

Programa 2

https://drive.google.com/file/d/1NIAVv40gYqkXvcwvqpd78jIWzdV-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	lmlumiquinga3@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

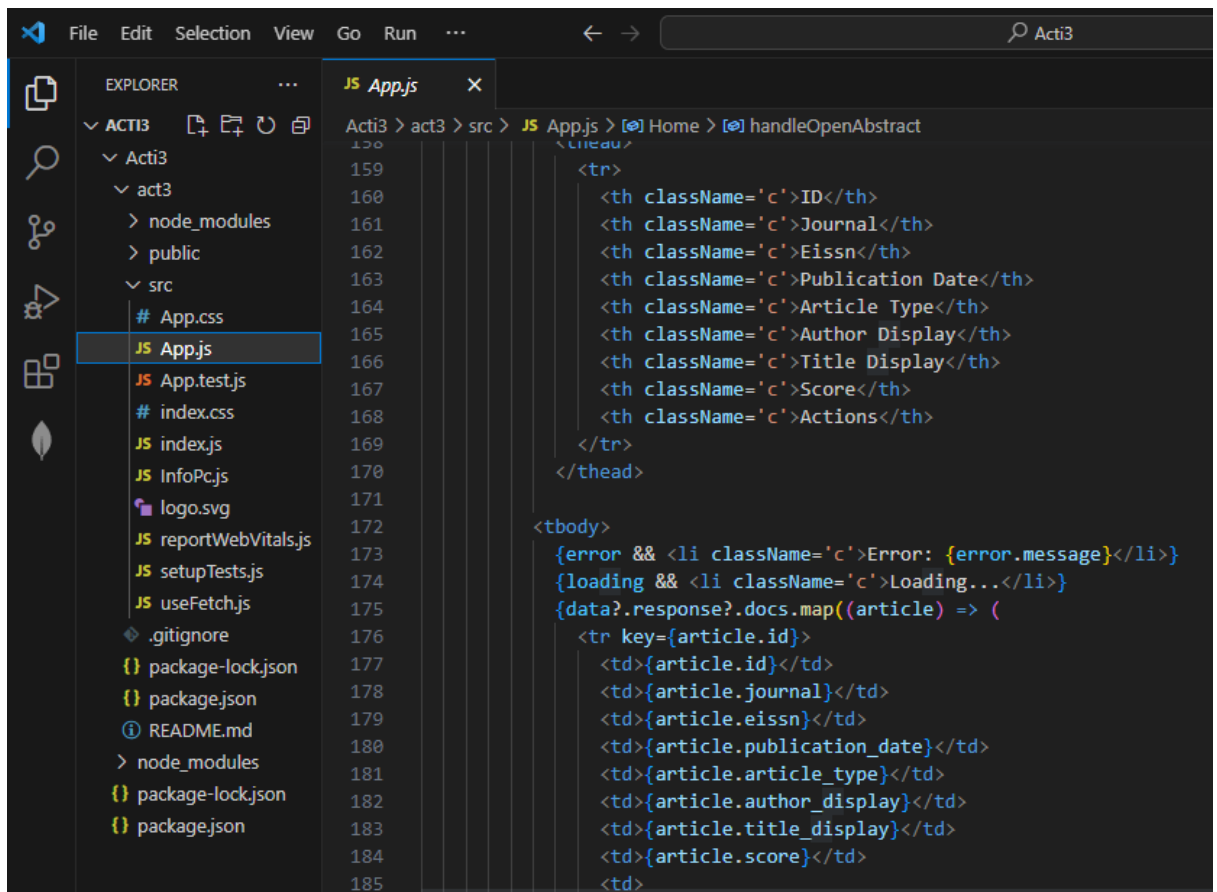
Nombre	Brandon Jimenez
Rol	Diseñador e Investigador
Categoría Profesional	Estudiante
Responsabilidad	Programación
Información de Contacto	bajimenez6@espe.edu.ec

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.



```
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Figura 1: Código de Tabla

El sitio se verá de la siguiente forma

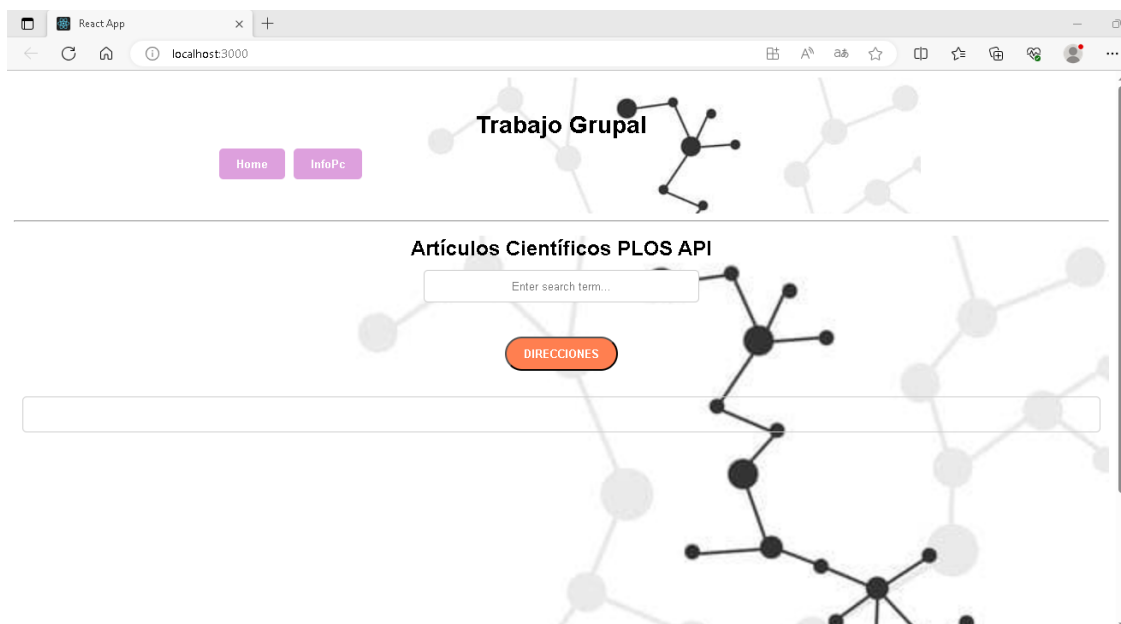
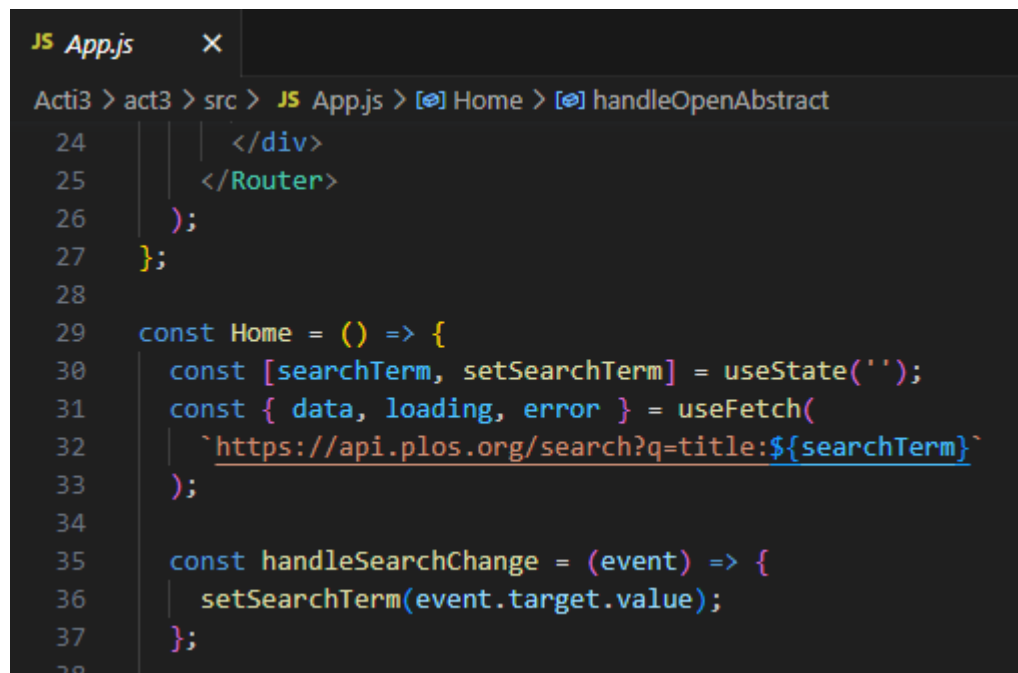


Figura 2: Vista de Página inicial

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.

A screenshot of a code editor with a dark theme. The editor shows a file named 'App.js' with a breadcrumb path: 'Act3 > act3 > src > JS App.js > Home > handleOpenAbstract'. The code is as follows:

```
24     </div>
25   </Router>
26 );
27 };
28
29 const Home = () => {
30   const [searchTerm, setSearchTerm] = useState('');
31   const { data, loading, error } = useFetch(
32     `https://api.plos.org/search?q=title:${searchTerm}`
33   );
34
35   const handleSearchChange = (event) => {
36     setSearchTerm(event.target.value);
37   };
38 }
```

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.


```

JS useFetch.js X
Acti3 > act3 > src > JS useFetch.js > useFetch
1 // useFetch.js
2 import { useState, useEffect } from 'react';
3
4 export function useFetch(url) {
5   const [data, setData] = useState(null);
6   const [loading, setLoading] = useState(true);
7   const [error, setError] = useState(null);
8
9   useEffect(() => {
10     async function fetchData() {
11       try {
12         const response = await fetch(url);
13         const data = await response.json();
14         setData(data);
15         setLoading(false);
16       } catch (error) {
17         setError(error);
18         setLoading(false);
19       }
20     }
21
22     fetchData();
23   }, [url]);
24
25   return { data, loading, error };
26 }
27

```

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.

```

<tbody>
  {error && <li className='c'>Error: {error.message}</li>}
  {loading && <li className='c'>Loading...</li>}
  {data?.response?.docs.map((article) => (
    <tr key={article.id}>
      <td>{article.id}</td>
      <td>{article.journal}</td>
      <td>{article.eissn}</td>
      <td>{article.publication_date}</td>
      <td>{article.article_type}</td>
      <td>{article.author_display}</td>
      <td>{article.title_display}</td>
      <td>{article.score}</td>
      <td>
        <button className='boton' onClick={() => handleOpenAbstract(article.abstract, article.title_display)}>Ver
      </td>
    </tr>
  ))}
</tbody>

```

Figura 5: Recuperación de datos de la API pública

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

ID	Journal	Eissn	Publication Date	Article Type	Author Display	Title Display	Score	Actions
10.1371/journal.pone.0216780	PLOS ONE	1932-6203	2019-05-23T00:00:00Z	Research Article	S. Katherine Nelson-CoffeyPeter M. RubertonJoseph ChancellorJessica E. CornickJim BlascovichSonja Lyubomirsky	The proximal experience of awe	15.641735	Ver Abstract
10.1371/journal.pone.0238204	PLOS ONE	1932-6203	2020-09-30T00:00:00Z	Research Article	Sheila Krogh-JespersenKimberly A. QuinnWilliam L. D. KrenzerChristine NguyenJana GreenslitC. Aaron Price	Exploring the awe-some: Mobile eye-tracking insights into awe in a science museum	15.608955	Ver Abstract
10.1371/journal.pone.0285049	PLOS ONE	1932-6203	2023-04-26T00:00:00Z	Research Article	Ryota TakanoMichio Nomura	Strengthened social ties in disasters: Threat-awe encourages interdependent worldviews via powerlessness	11.167852	Ver Abstract
10.1371/journal.pone.0243243	PLOS ONE	1932-6203	2020-12-17T00:00:00Z	Research Article	Asim KhanUmar NawazAnwaar UlhaqRandal W. Robinson	Real-time plant health assessment via implementing cloud-based scalable transfer learning on AWS DeepLens	10.0860405	Ver Abstract
10.1371/journal.pone.0205381	PLOS ONE	1932-6203	2018-10-11T00:00:00Z	Research Article	Melaku Tadege EngidawMolla Mesele WassieAlemayehu Shimeka Tefera	Anemia and associated factors among adolescent girls living in Aw-Barre refugee camp, Somali regional state, Southeast Ethiopia	9.474207	Ver Abstract
					Shriva MisraNirupa MisraBotumelo	"I would watch her with awe as she swallowed the		

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.

```

39     const handleOpenAbstract = (abstract, title_display) => {
40         // Abrir el abstract en una nueva ventana
41         const newWindow = window.open('', '_blank');
42
43         newWindow.document.write(`<html>

```

Figura 7: Especificación de Abract Inicio

```

67         <body>
68         <center><h1 style="color:#ffffff">Artículo: ${title_display}</h1></center>
69         <div class='p'> ${abstract} </div>
70         <center><button id="returnButton" class='boton'>Regresar</button></center>
71         </body>
72         </html>`);
73         newWindow.document.close();
74
75         // Llamar a handleReturn al hacer clic en el botón "Regresar"
76         newWindow.document.getElementById('returnButton').addEventListener('click', () => {
77             newWindow.close(); // Cierra la ventana actual al regresar a app.jsx
78         });

```

Figura 8: Recuperación de la información del artículo

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

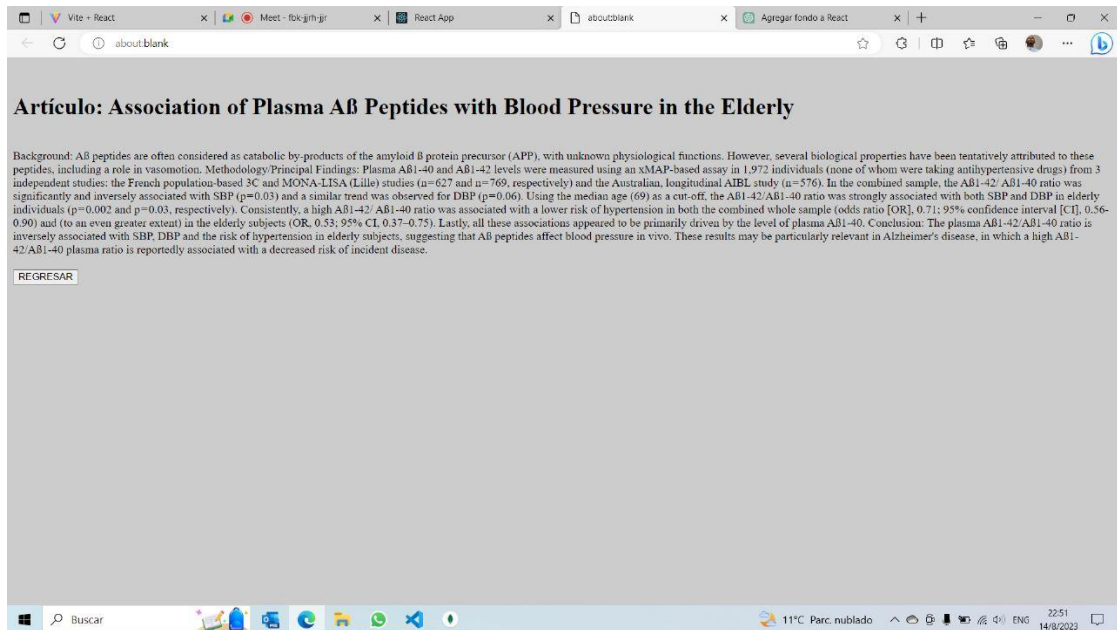


Figura 9: Abstract del Artículo

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.

```
const handleDir = () => {  
  // Abrir el abstract en una nueva ventana  
  const newWindow = window.open('', '_blank');  
  
  newWindow.document.write(`<html>`  
}
```

Figura 10: Especificación de Dirección

```
114 <tbody>  
115   ${data?.response?.docs.map((article) => `  
116     <tr key=${article.id}>  
117       <td>${article.title_display}</td>  
118       <td>  
119         <a href="${apiUrl}${article.id}" target="_blank" rel="noopener noreferrer">  
120           ${apiUrl}${article.id}  
121         </a>  
122       </td>  
123     </tr>  
124   `).join('')}  
125 </tbody>  
126 </table>  
127 <br>  
128 <center><button class='boton' id='returnButton'>Regresar</button></center>  
129 </body>  
130 </html>`;   
131 newWindow.document.close();
```

Figura 11: Obtención de datos para la tabla

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

Titulos de Artículos	
Association of Plasma Aβ Peptides with Blood Pressure in the Elderly	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0018536
Tailoring the Antibody Response to Aggregated Aβ Using Novel Alzheimer-Vaccines	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0115237
The Polyphenol Oleuropein Aglycone Protects TgCRND8 Mice against Aβ Plaque Pathology	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0071702
Extracellular vesicles enriched with amylin receptor are cytoprotective against the Aβ toxicity <i>in vitro</i>	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0267164
Taming the late Quaternary phylogeography of the Eurasian wild ass through ancient and modern DNA	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0174216
Aggregation of Aβ(25-35) on DOPC and DOPC/DHA Bilayers: An Atomic Force Microscopy Study	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0115780
Concordant Association of Insulin Degrading Enzyme Gene (<i>IDF</i>) Variants with <i>IDF</i> mRNA, Aβ, and Alzheimer's Disease	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0008764
Space-Use Patterns of the Asiatic Wild Ass (<i>Equus hemionus</i>): Complementary Insights from Displacement, Recursion Movement and Habitat Selection Analyses	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0143279
Earliest evidence for equid bit wear in the ancient Near East: The "ass" from Early Bronze Age Tell es-Sāfi/Gath, Israel	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0106335
Correction: Earliest evidence for equid bit wear in the ancient Near East: The "ass" from Early Bronze Age Tell es-Sāfi/Gath, Israel	https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0202382

Figura 12: Visualización de los resultados de las búsquedas

Informacion de la PC	
Contenido detallado	
Archivos aceptados:	"/"
Idioma:	en-US,en;q=0.9
Host remoto:	httpbin.org
IP que originó la solicitud:	
Nombre del navegador:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36 Edg/115.0.1901.200
Sistema operativo:	"Windows"

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, maneja 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.



```
JS useFetch.js X
Acti3 > act3 > src > JS useFetch.js > useFetch > useEffect() callback
2 import { useState, useEffect } from 'react';
3
4 export function useFetch(url) {
5   const [data, setData] = useState(null);
6   const [loading, setLoading] = useState(true);
7   const [error, setError] = useState(null);
8
9   useEffect(() => {
10     async function fetchData() {
11       try {
12         const response = await fetch(url);
13         const data = await response.json();
14         setData(data);
15         setLoading(false);
16       } catch (error) {
17         setError(error);
18         setLoading(false);
19       }
20     }
21
22     fetchData();
23   }, [url]);
24
25   return { data, loading, error };
26 }
```

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.


```

JS App.js x
Acti3 > act3 > src > JS App.js > Home > handleDir > data.response.docs.map()
27   };
28
29   const Home = () => {
30     const [searchTerm, setSearchTerm] = useState('');
31     const { data, loading, error } = useFetch(
32       `https://api.plos.org/search?q=title:${searchTerm}`
33     );
34
35     const handleSearchChange = (event) => {
36       setSearchTerm(event.target.value);
37     };
38
39     const handleOpenAbstract = (abstract, title_display) => {
40       // Abrir el abstract en una nueva ventana
41       const newWindow = window.open('', '_blank');
42
43       newWindow.document.write(`<html>

```

Figura.-2 App.js

```

138
139   const apiUrl = `https://journals.plos.org/plosone/article?id=`
140

```

Figura.-2.1 App.js (API)

```

159
160     <th className='c'>ID</th>
161     <th className='c'>Journal</th>
162     <th className='c'>Eissn</th>
163     <th className='c'>Publication Date</th>
164     <th className='c'>Article Type</th>
165     <th className='c'>Author Display</th>
166     <th className='c'>Title Display</th>
167     <th className='c'>Score</th>
168     <th className='c'>Actions</th>
169   </tr>
170 </thead>
171
172   <tbody>
173     {error && <li className='c'>Error: {error.message}</li>}
174     {loading && <li className='c'>Loading...</li>}
175     {data?.response?.docs.map((article) => (
176       <tr key={article.id}>
177         <td>{article.id}</td>
178         <td>{article.journal}</td>
179         <td>{article.eissn}</td>
180         <td>{article.publication_date}</td>
181         <td>{article.article_type}</td>
182         <td>{article.author_display}</td>
183         <td>{article.title_display}</td>
184         <td>{article.score}</td>
185         <td>

```

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

```

<tbody>
  ${data?.response?.docs.map((article) => `
    <tr key=${article.id}>
      <td>${article.title_display}</td>
      <td>
        <a href="${apiUrl}${article.id}" target="_blank" rel="noopener noreferrer">
          ${apiUrl}${article.id}
        </a>
      </td>
    </tr>
  `).join('')}
</tbody>
</table>
</br>
<center><button class='boton' id="returnButton">Regresar</button></center>
</body>
</html>`);
newWindow.document.close();

// Llamar a handleReturn al hacer clic en el botón "Regresar"
newWindow.document.getElementById('returnButton').addEventListener('click', () => {
  newWindow.close(); // Cierra la ventana actual al regresar a app.jsx
});
};

const apiUrl = `https://journals.plos.org/plosone/article?id=`

```

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>


```

1 {
2   "response": {
3     "numFound": 7,
4     "start": 0,
5     "maxScore": 13.419994,
6     "docs": [
7       {
8         "id": "10.1371/journal.pcbi.1004668",
9         "journal": "PLOS Computational Biology",
10        "eissn": "1553-7358",
11        "publication_date": "2016-01-19T00:00:00Z",
12        "article_type": "Education",
13        "author_display": [
14          "John D. Blischak",
15          "Emily R. Davenport",
16          "Greg Wilson"
17        ],
18        "abstract": [
19          ""
20        ],
21        "title_display": "A Quick Introduction to Version Control with Git and GitHub",
22        "score": 13.419994
23      },
24      {
25        "id": "10.1371/journal.pone.0205898",
26        "journal": "PLOS ONE",
27        "eissn": "1932-6203",
28        "publication_date": "2018-10-31T00:00:00Z",
29        "article_type": "Research Article",
30        "author_display": [
31          "Pamela H. Russell",
32          "Rachel L. Johnson",
33          "Shreyas Ananthan",
34          "Benjamin Harnke",
35          "Nichole E. Carlson"
36        ]
37      }
38    ]
39  }
40 }

```

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

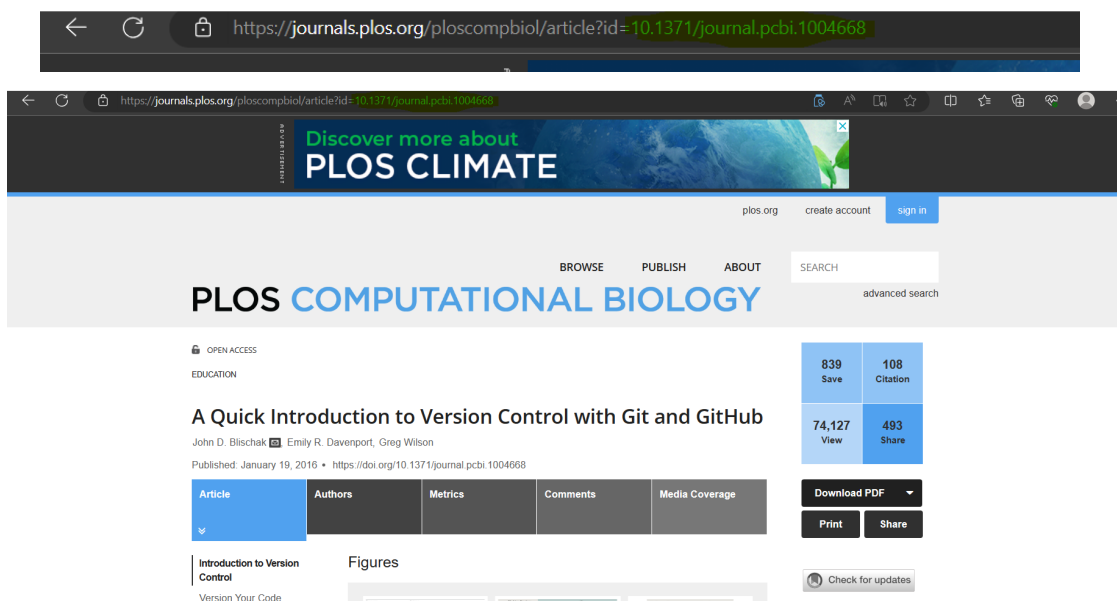


Figura.-3 página de investigación redireccionada.

5. Validación y prueba del proyecto.

Pruebas:

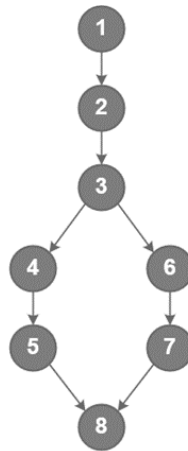
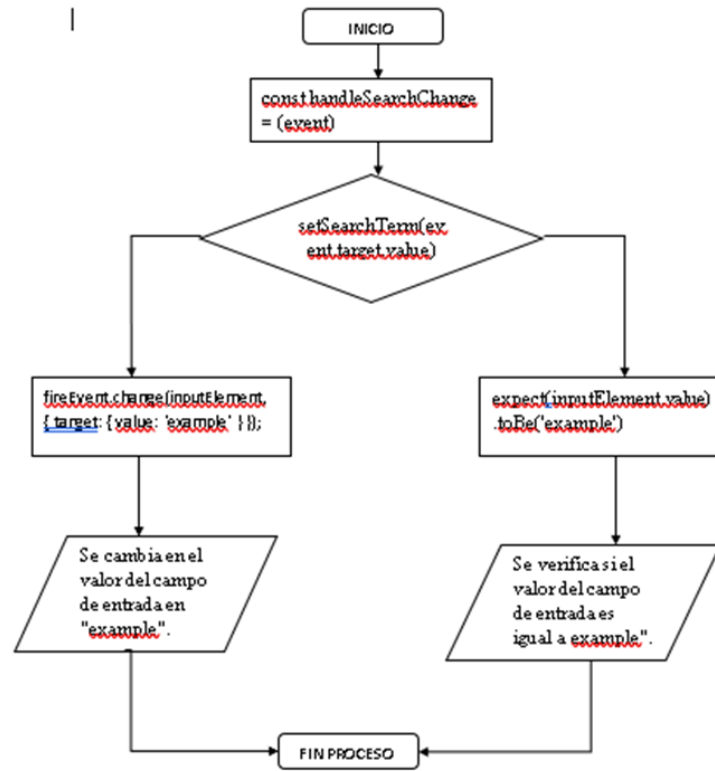
Caja Blanca

```

41 const Home = () => {
42   const [searchTerm, setSearchTerm] = useState('');
43   const { data, loading, error } = useFetch(
44     `https://api.plos.org/search?q=title:${searchTerm}`
45   );
46
47   const handleSearchChange = (event) => {
48     setSearchTerm(event.target.value);
49   };
50 }

```

```
118     return (  
119       <div className="home-container" style={backgroundStyle}>  
120         <h1 className="main-title">Articulos Científicos PLOS API</h1>  
121         <input  
122           type="text"  
123           value={searchTerm}  
124           onChange={handleSearchChange}  
125           placeholder="Enter search term..."  
126           className="search-input"  
127         />  
128  
129         <br />  
130         <br />  
131         <button onClick={() => handleDir()} className="dir-button">  
132           DIRECCIONES  
133         </button>  
134         <br />  
135         <br />  
136         <div className="card">  
137         </div>  
138       </div>  
139     );
```



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 Mongoddb 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 tarea y salir.

```

4   try {
5     const tasks = await Task.find({ user : req.user.id }).populate("user");
6     res.json(tasks);
7   } catch (error) {
8     return res.status(500).json({ message: error.message });
9   }
10  };
11
12  export const createTask = async (req, res) => {
13    try {
14      const { title, description, date } = req.body;
15      const newTask = new Task({
16        title,
17        description,
18        date,
19        user: req.user.id,
20      });
21      await newTask.save();
22      res.json(newTask);
23    } catch (error) {
24      return res.status(500).json({ message: error.message });
25    }
26  };
27

```

```

PS C:\Users\Karen\OneDrive\Documents\Github\Programa 2\prog\.-vscode> npm start
> mern-tasks@1.0.0 start
> cross-env NODE_ENV=production node src/index.js

"cross-env" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
PS C:\Users\Karen\OneDrive\Documents\Github\Programa 2\prog\.-vscode>

```

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

El Registro de usuario de visualizar de la siguiente manera:

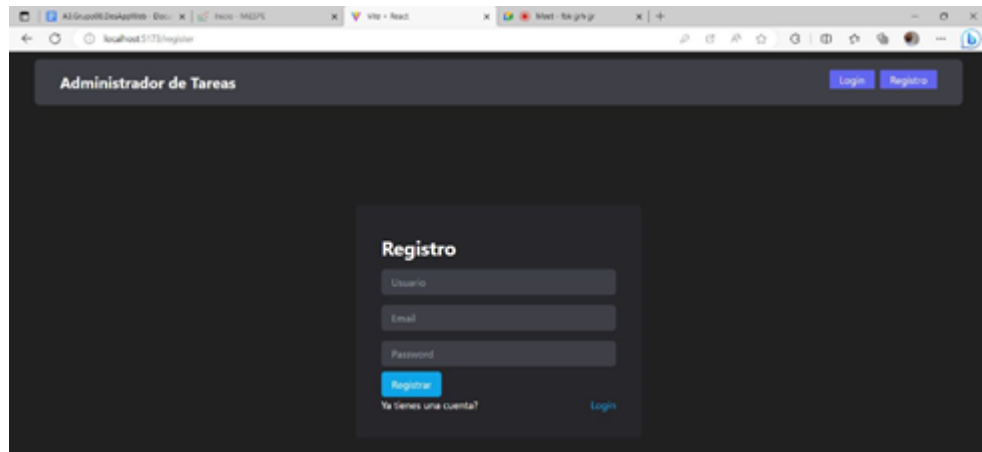


Ilustración 2 Registro de usuario. Autor: Propio.

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

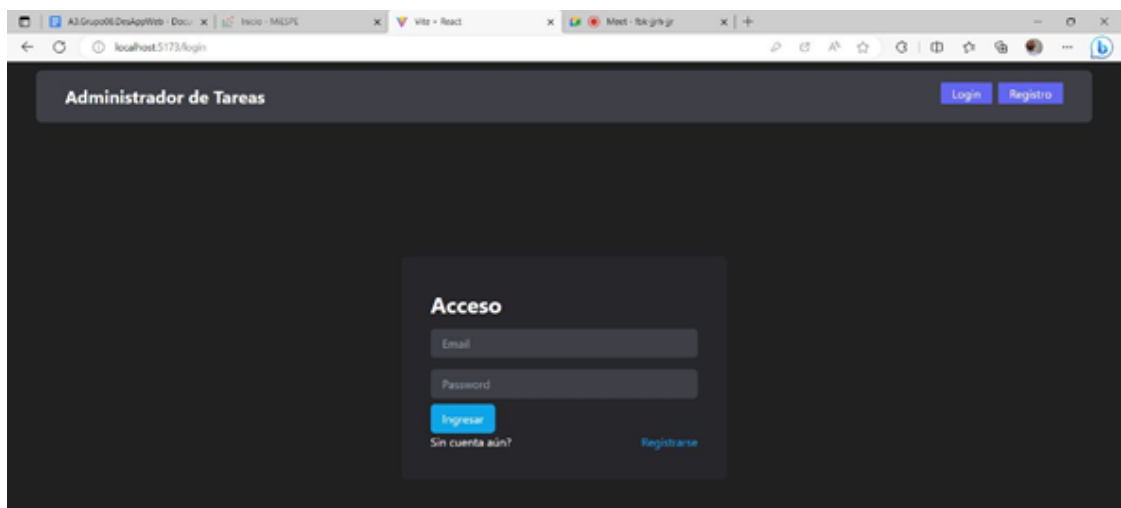


Ilustración 3. Login del usuario. Autor: Propio

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

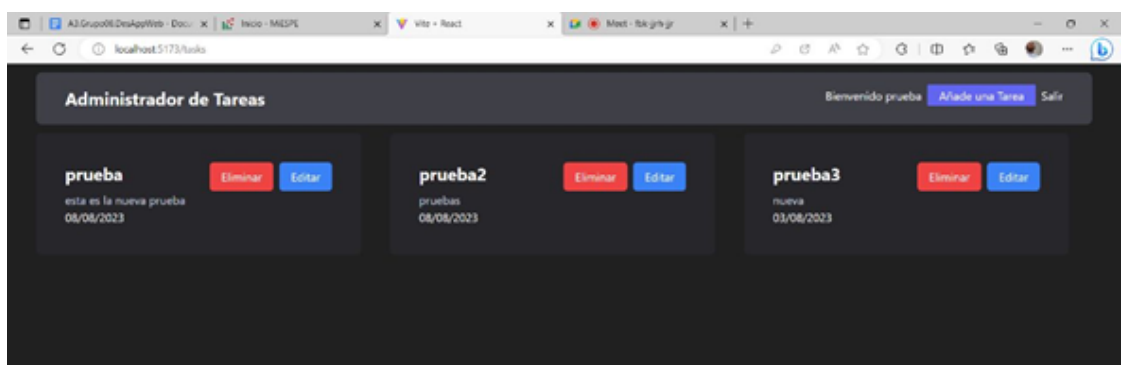
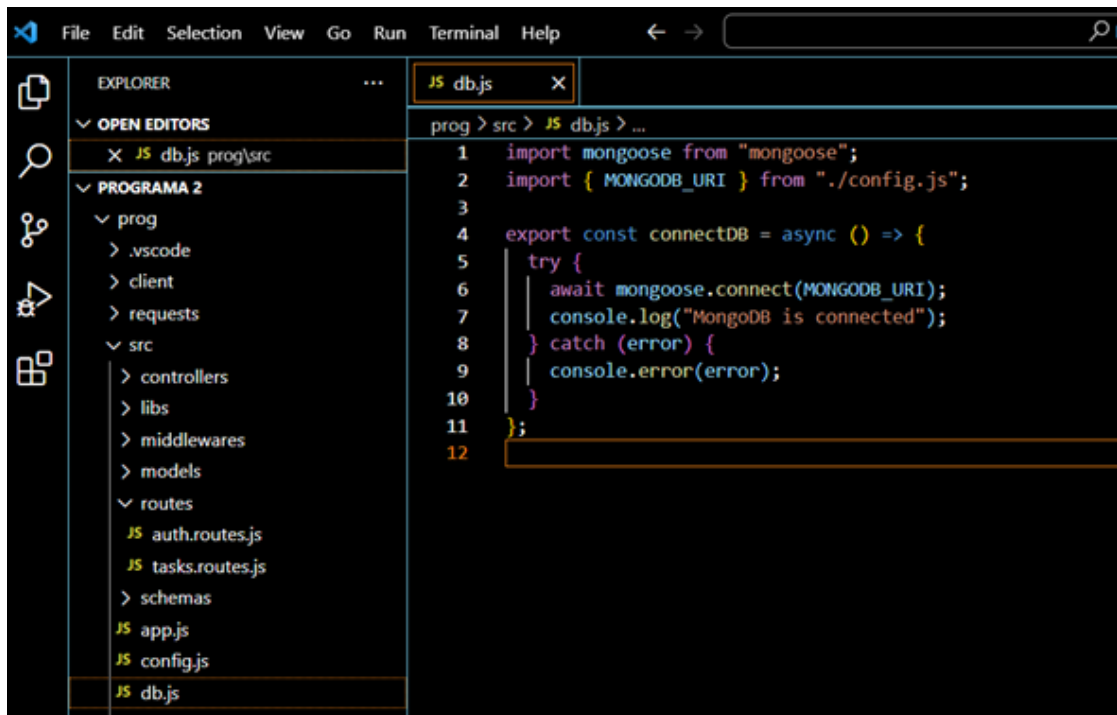


Ilustración 4. Visualización de Administración de tareas. Autor: Propio

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



The image shows a Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders like 'prog', 'client', 'requests', and 'src'. Under 'src', there are subfolders 'controllers', 'libs', 'middlewares', 'models', 'routes', and 'schemas'. The 'routes' folder contains 'auth.routes.js', 'tasks.routes.js', and 'schemas'. The 'schemas' folder contains 'app.js', 'config.js', and 'db.js'. The 'db.js' file is selected and open in the editor. The code in 'db.js' is as follows:

```
1 import mongoose from "mongoose";
2 import { MONGODB_URI } from "../config.js";
3
4 export const connectDB = async () => {
5   try {
6     await mongoose.connect(MONGODB_URI);
7     console.log("MongoDB is connected");
8   } catch (error) {
9     console.error(error);
10  }
11 };
12
```

Ilustración 5. Base de datos. Autor: Propio.

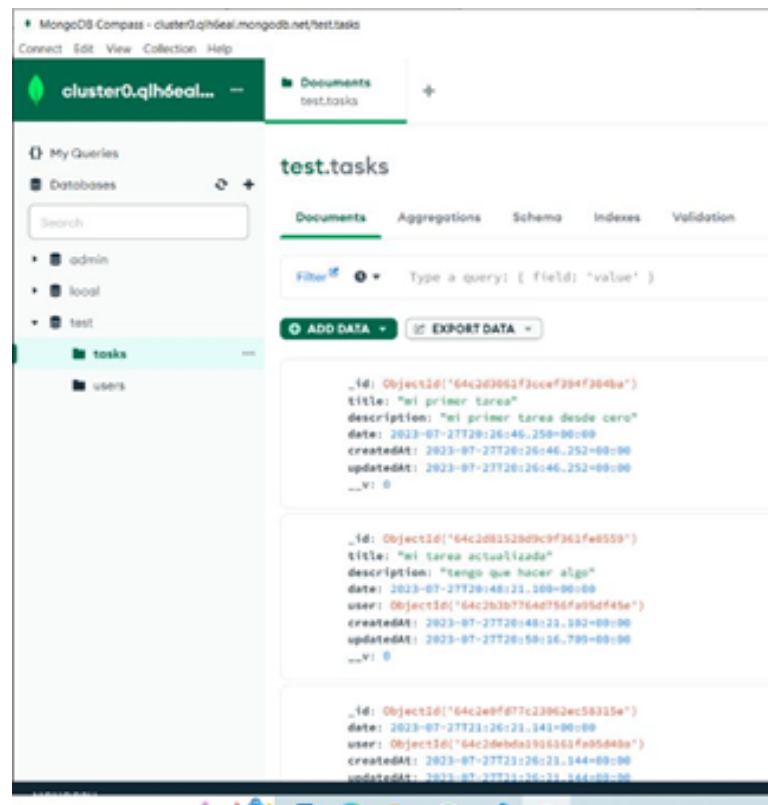


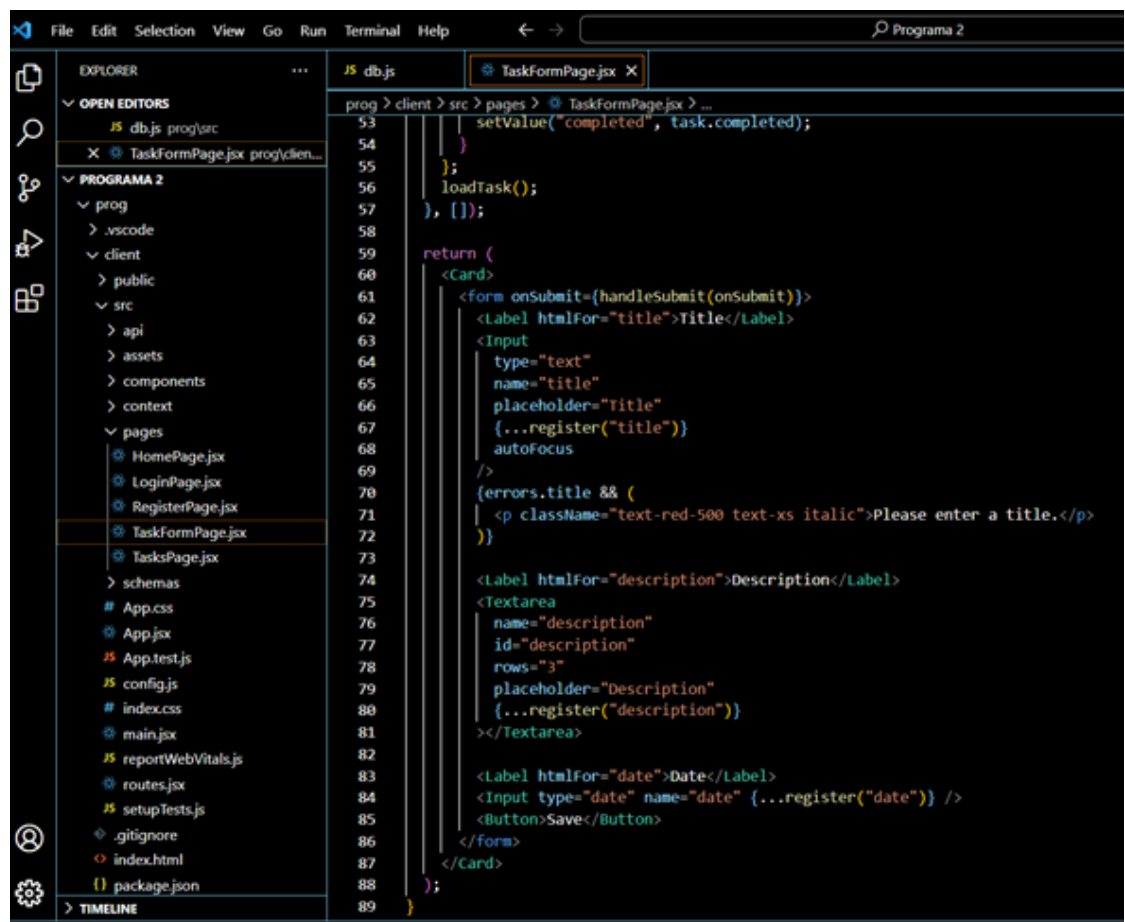
Ilustración 6. MongoDB. Autor: Propio

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.



```
53 |         setValue("completed", task.completed);
54 |     };
55 |     };
56 |     loadTask();
57 | }, []);
58 |
59 | return (
60 |     <Card>
61 |         <form onSubmit={handleSubmit(onSubmit)}>
62 |             <Label htmlFor="title">Title</Label>
63 |             <Input
64 |                 type="text"
65 |                 name="title"
66 |                 placeholder="title"
67 |                 {...register("title")}
68 |                 autoFocus
69 |             />
70 |             {errors.title && (
71 |                 <p className="text-red-500 text-xs italic">Please enter a title.</p>
72 |             )}
73 |             <Label htmlFor="description">Description</Label>
74 |             <Textarea
75 |                 name="description"
76 |                 id="description"
77 |                 rows="3"
78 |                 placeholder="Description"
79 |                 {...register("description")}
80 |             />
81 |             <Label htmlFor="date">Date</Label>
82 |             <Input type="date" name="date" {...register("date")} />
83 |             <Button>Save</Button>
84 |         </form>
85 |     </Card>
86 | );
87 |
88 |
89 | }
```

Ilustración 7. Código Crear tarea. Autor: Propio

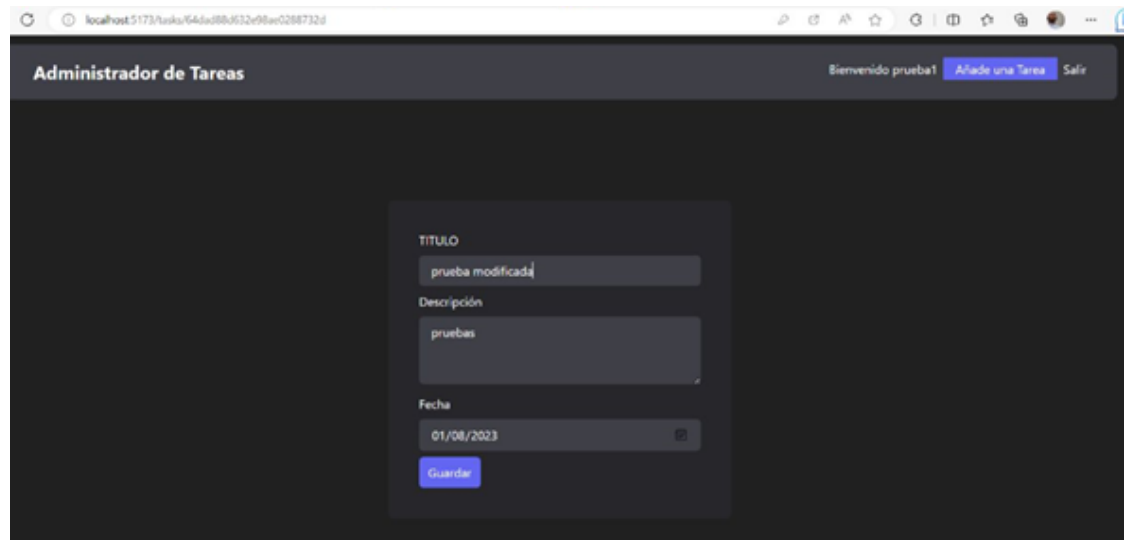


Ilustración 8. URI Crear tareas. Autor: Propio

- Visualizar tareas.

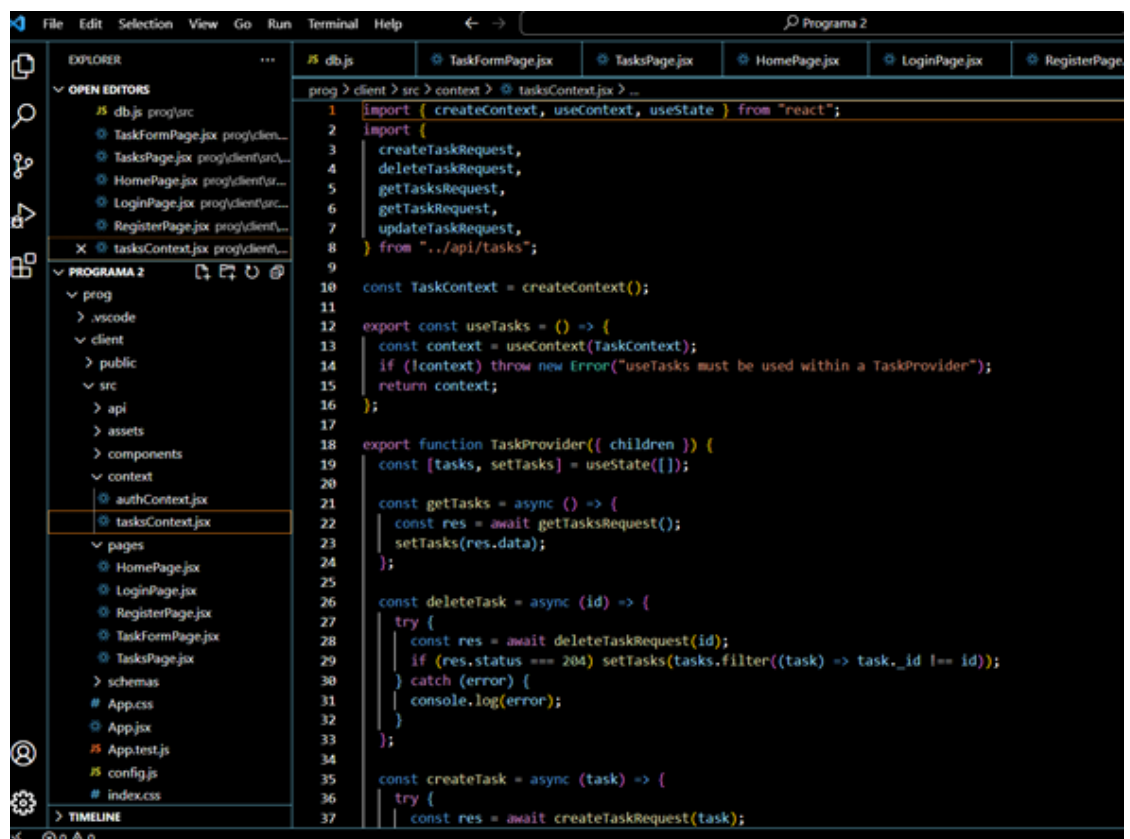


Ilustración 9. Código de la administración de tareas. Autor: Propio

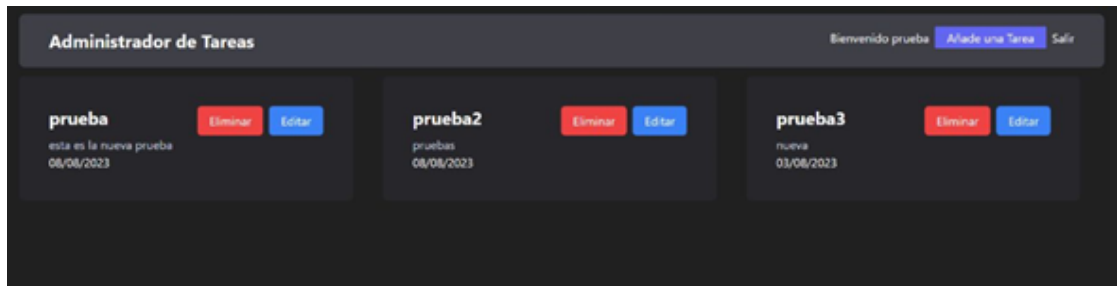


Ilustración 10. URI Visualizar tareas. Autor: Propio

- Editar tarea

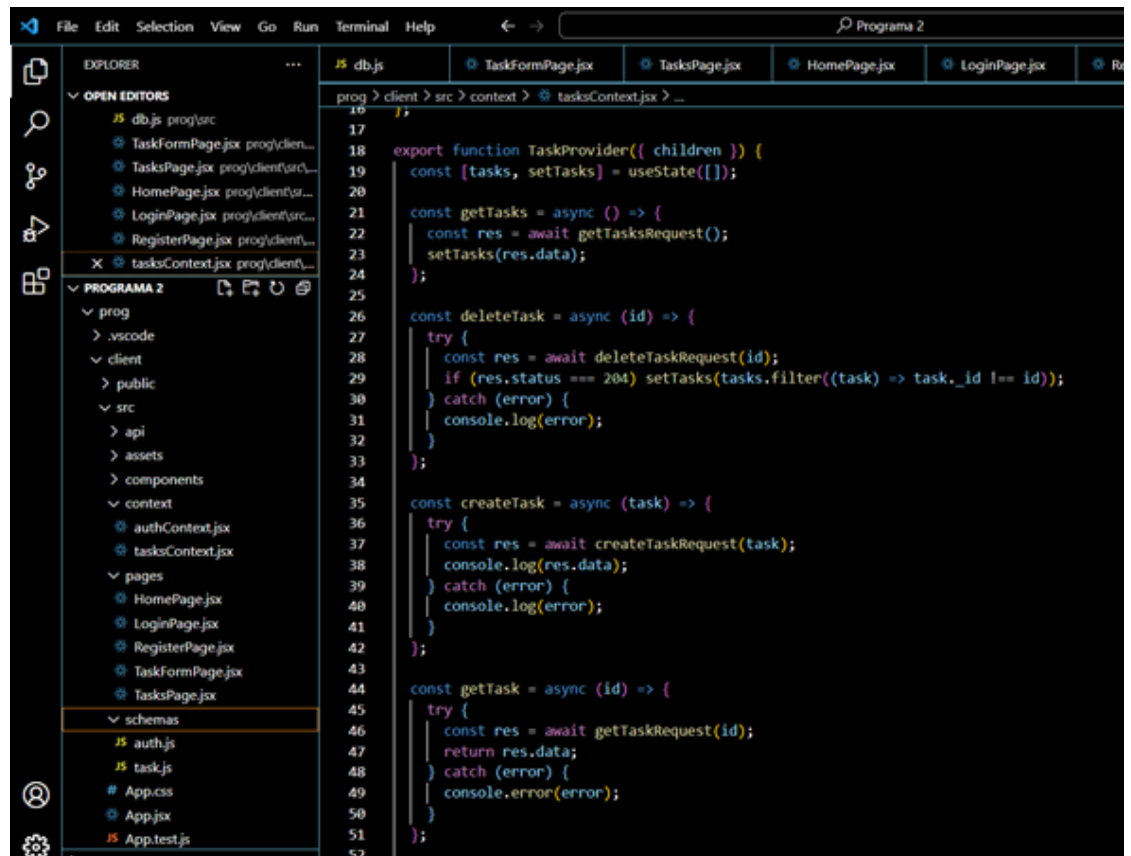


Ilustración 11. Código de editar tarea. Autor: Propio

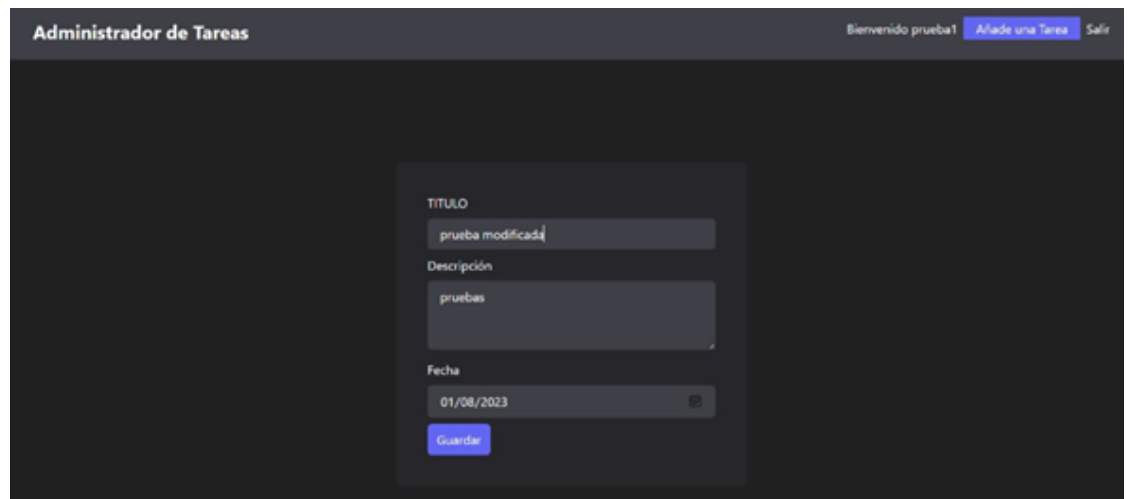


Ilustración 12 URI Editar tarea. Autor: Propio

- Eliminar tarea.

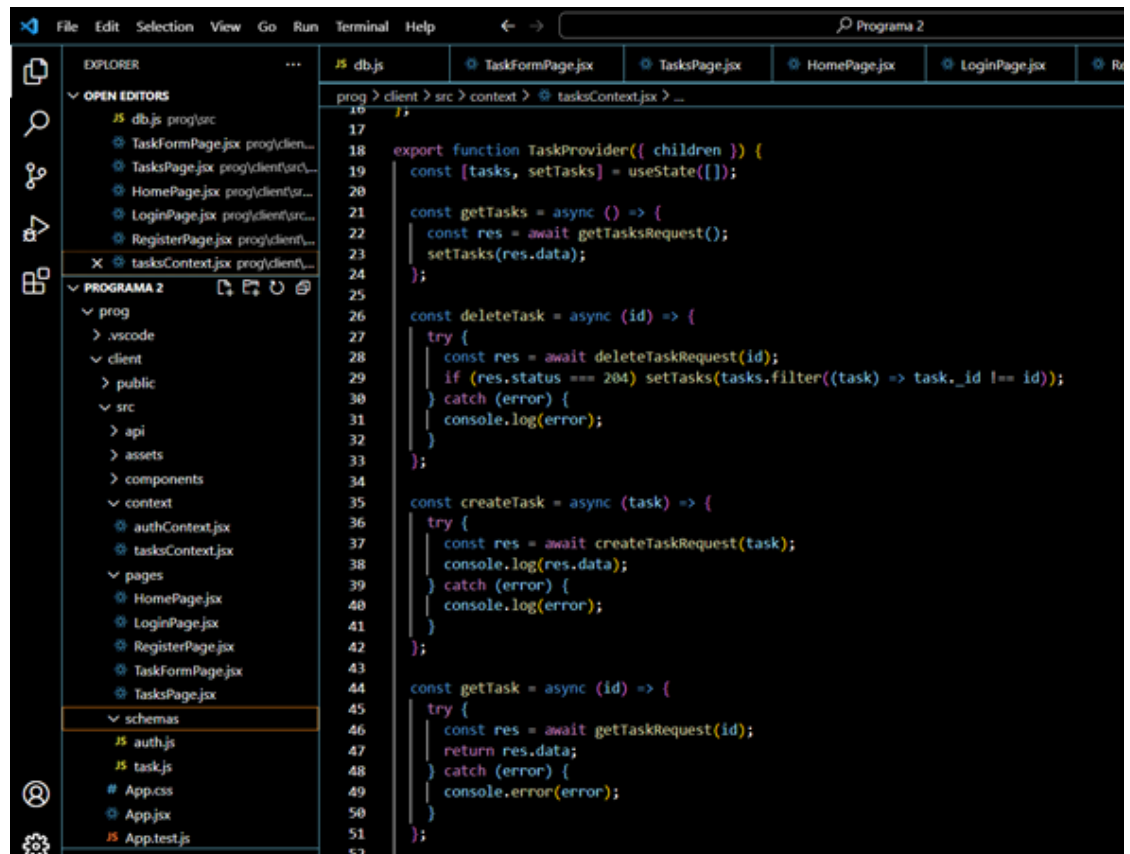


Ilustración 13. Código de eliminar tarea. Autor: Propio

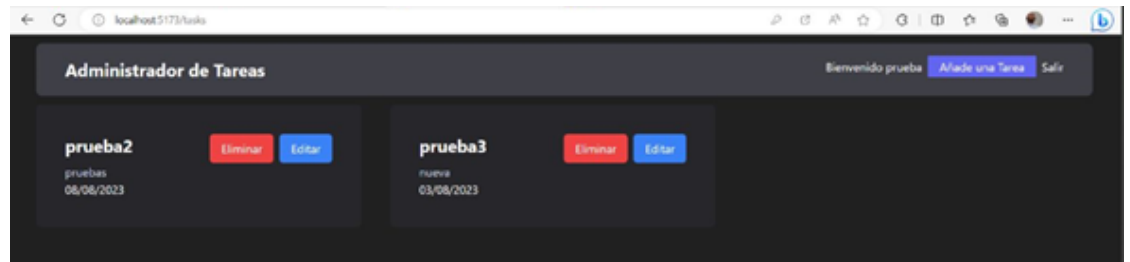


Ilustración 14. URI Eliminar tarea. Autor: Propio.

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"

```

export const register = async (req, res) => {
  const { email, password, username } = req.body;

  try {
    const userFound = await User.findOne({ email });
    if (userFound) return res.status(400).json(["El email ya esta en uso"]);

    const passwordHash = await bcrypt.hash(password, 10);

    const newUser = new User({
      username,
      email,
      password: passwordHash,
    });

    const userSaved = await newUser.save();
    const token = await createAccessToken({ id: userSaved._id });

    res.cookie("token", token);
    res.json({
      id: userSaved._id,
      username: userSaved.username,
      email: userSaved.email,
      createdAt: userSaved.createdAt,
      updatedAt: userSaved.updatedAt,
    });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

```

Figura 1.- validación de Registro email.

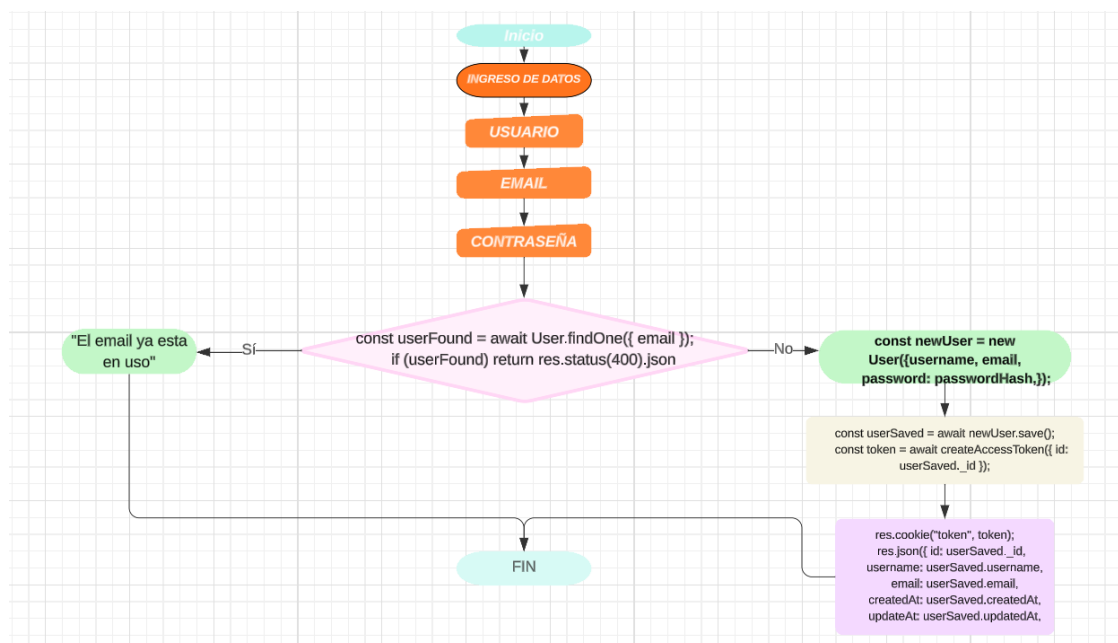


Figura 2.- Algoritmo validación de Registro con email.

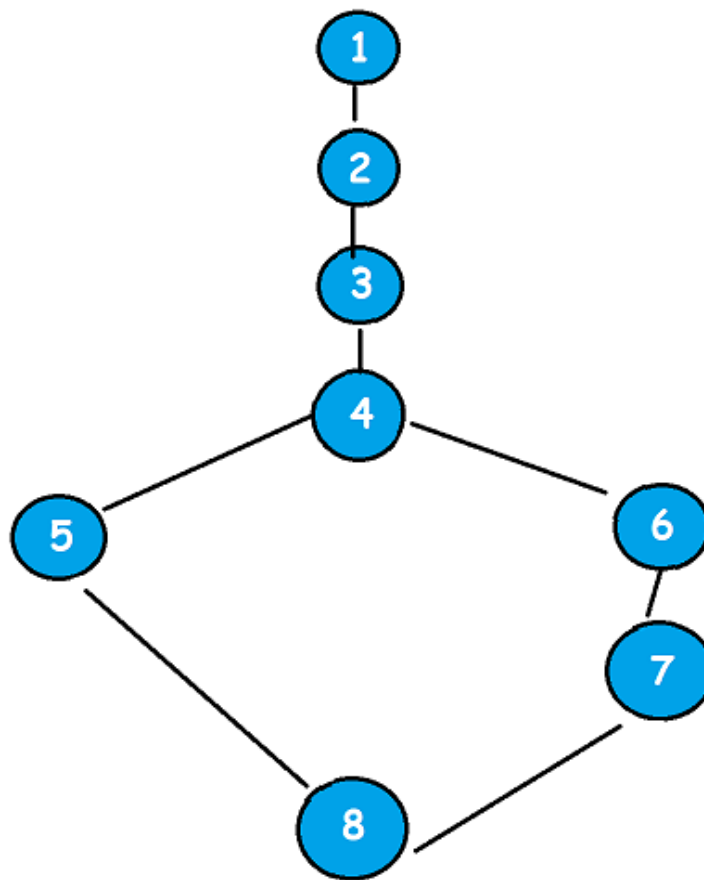


Figura 3.- Diagrama de flujo validación de Registro con email.

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	<pre>password: z .string({ required_error: "Password es requerido", })</pre>	Válido	LuisLL

	<pre>password: z .string({ required_error: 'Password es requerido', }) .min(6, { message: 'Password minimo con 6 caracteres', }), })</pre>	No válido	Luis1234
Registro	<pre>const { data, loading }</pre>	Válido	Usuario: Test12 Email: tes@gmail.com Password: 123456
	<pre>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>)}</pre>	No válido	Usuario: Test123 Email: tes@gmail.com 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”

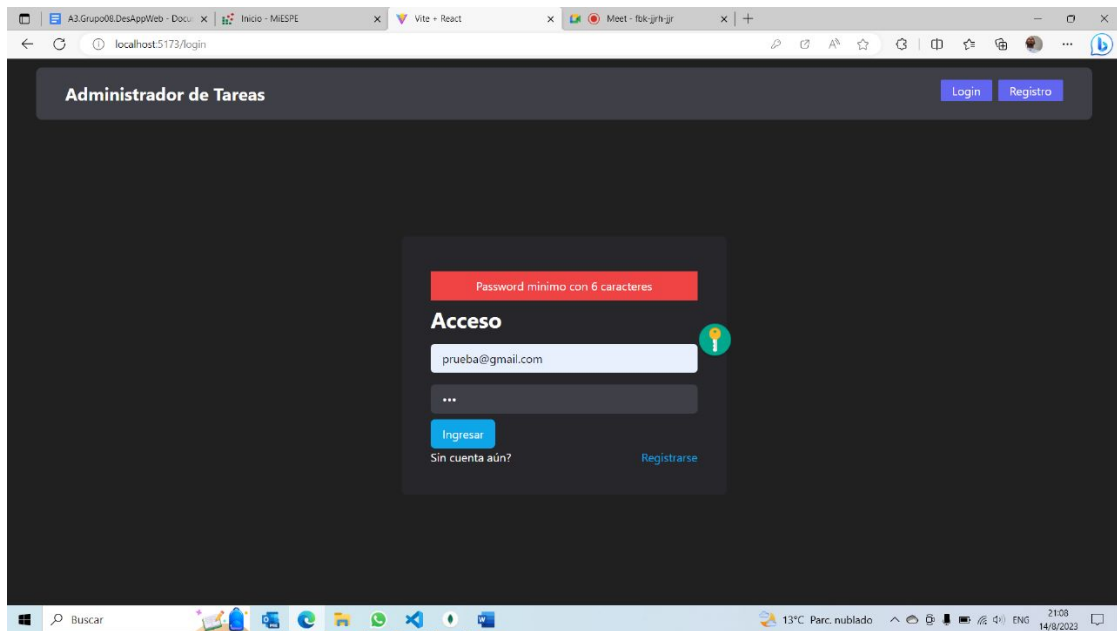


Figura 2.- Validación de Password mínimo 6 caracteres.

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”

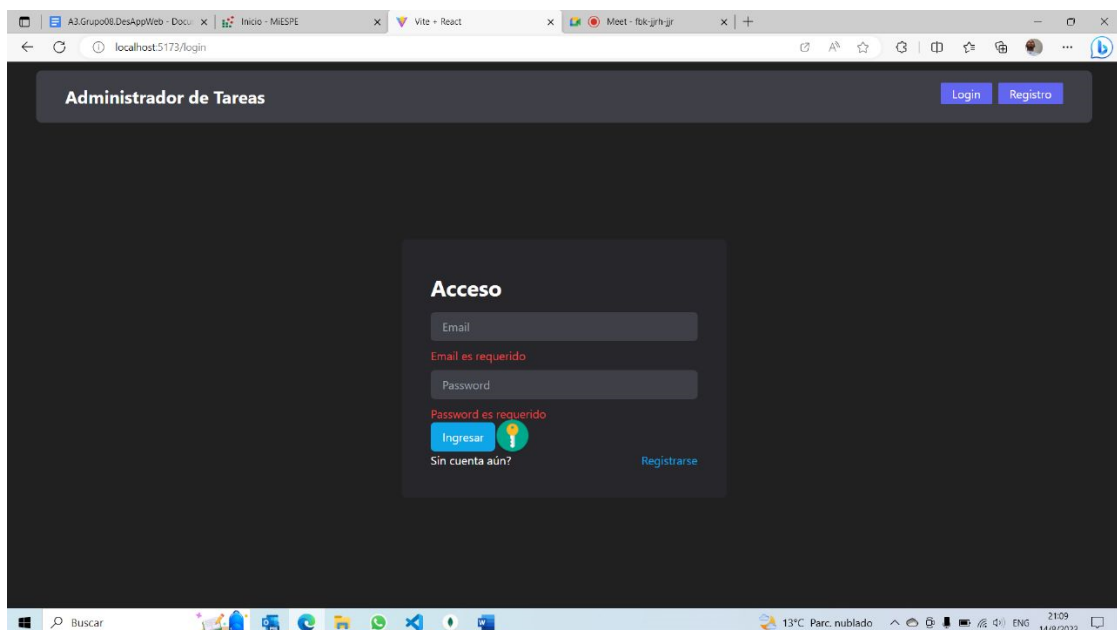


Figura 3.- Validación de Acceso (login) .

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.

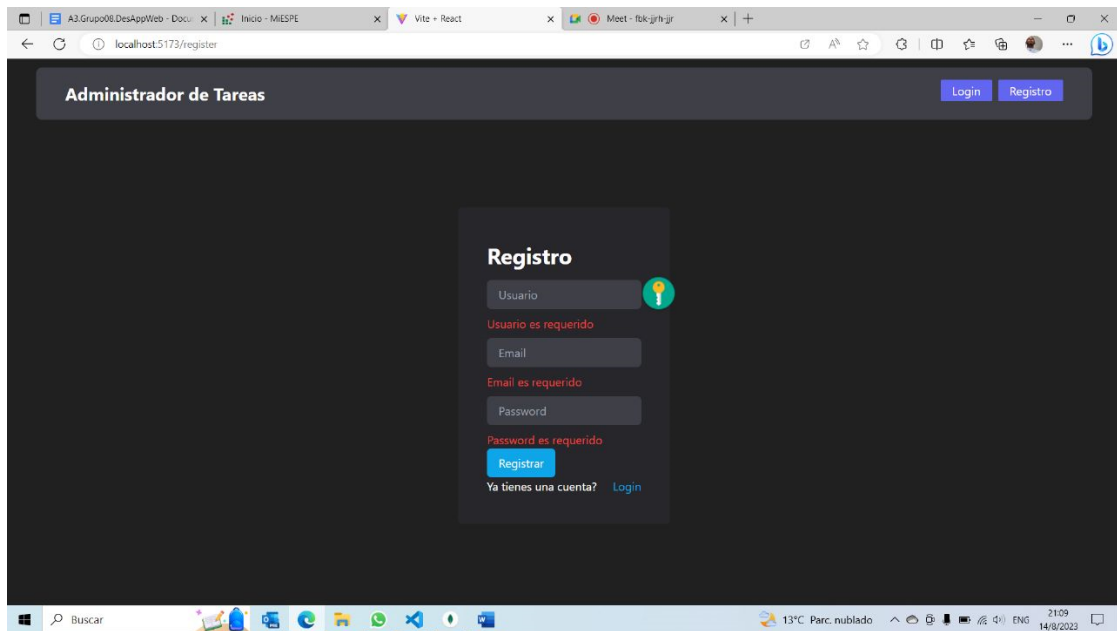


Figura 4.- Figura 3.- Validación de Registro (login) .

4. 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.

5. 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.

6. 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>