

Práctica - Ciudades (Noviembre 2022)

Índice

Presentación	3
Introducción	3
Recursos	3
Pre-Requisitos.....	4
1.1 CSS Flexbox Guía completa DESDE CERO	4
1.2 Custom hooks y hook Context para conseguir un estado global.....	5
Enunciado	6
Servicios	7
Servicio zipopotam.....	7
Servicio open meteo	8
Google maps.....	10
Desarrollo.....	11
Pantalla de búsqueda	11
Cards informativas	15
Pantalla de historial.....	18
Buenas prácticas.....	19
Contextos globales.....	19
Parametrización	19
Refactorización de componentes	19

Presentación

Introducción

A continuación, se presenta una práctica guiada para la creación de una aplicación React, el objetivo es poner en práctica los conceptos de react obtenidos en la formación.

Recursos

Todos los recursos necesarios utilizados en la formación se pueden encontrar en el directorio de la práctica, dentro de la carpeta recursos.

- Imágenes: Todas las imágenes necesarias para la construcción de la práctica.

Para una correcta implementación de la práctica es recomendable la comprensión de los siguientes manuales (Disponibles en los recursos de la formación Open Front):

- Estructura de proyectos Open Front.
- Creación de contextos globales.
- Consumo de apis con custom hooks.

Pre-Requisitos

Antes de comenzar con el desarrollo de la práctica, se recomienda la visualización de los vídeos de esta sección.

1.1 CSS Flexbox | Guía completa DESDE CERO

➤ [CSS Flexbox | Guía completa DESDE CERO](#)

00:00 *Introducción*
00:25 *Estructura inicial*
01:45 *display flex vs inline-flex*
03:31 *flex-direction*
05:26 *flex-wrap*
06:35 *justify-content*
08:12 *align-items*
10:35 *align-content*
13:15 *flex-flow*
13:40 *order*
14:35 *align-self*
15:26 *margin auto*
16:21 *flex-basis*
19:27 *flex-grow*
21:40 *flex-shrink*
22:37 *la propiedad flex*
27:04 *Despedida*

1.2 Custom hooks y hook Context para conseguir un estado global

- [Creando Custom Hooks y usando Context para conseguir un estado global en ReactJS \(01:37:53\)](#)

00:00	Bienvenida y presentación del código
07:22	Añadiendo un form que haga de buscador de Gifs
12:10	Usando useLocation de wouter
21:10	Extrayendo lógica a un custom hook: useGifs
34:20	¿Custom Hooks son High Order Components?
36:25	useGifs, haciendo que use el localStorage
44:45	¿Por qué extraer los métodos de callback en constantes?
47:10	Usando Context de ReactJS
50:25	Usando el Provider del Context
52:45	Acceder a la información del Context desde un componente
54:41	Valor inicial de Context por si no usas Provider
01:00:44	El problema de hacer export default
01:03:10	Lograr un estado global usando el Context
01:17:06	Recuperar el detalle del gif del estado global
01:19:59	Preguntas del chat
01:34:38	Buenas prácticas para el Context por Kent C. Dodds
01:36:05	Despedida y cierre

Enunciado

El objetivo de la práctica es crear una aplicación que a partir de un código postal muestre información sobre el lugar correspondiente a dicho código. Esta información se mostrará organizada en diferentes cards, haciendo posible añadir nuevas funcionalidades de forma sencilla.

Además, la aplicación dispondrá de un historial de búsqueda desde donde se podrá volver a cargar un código postal buscado anteriormente.



Servicios

Servicio zipopotam

Este servicio se utilizará para obtener información sobre una ciudad a través de su código postal.

La url de llamada al servicio se construye de la siguiente forma:

`https://api.zippopotam.us/es/{Codigo postal}`

Donde código postal es el código de la ciudad sobre la que queremos buscar información.

Por ejemplo:

<https://api.zippopotam.us/es/24006>

<https://api.zippopotam.us/es/49155>

El resultado del servicio es el siguiente:

```
{
  "post code": "24006",
  "country": "Spain",
  "country abbreviation": "ES",
  "places": [{
    "place name": "Le\u00f3n",
    "longitude": "-5.5667",
    "state": "Castilla - Leon",
    "state abbreviation": "CL",
    "latitude": "42.6"
  }]
}
```

Entre los datos que nos devuelve el servicio, hay que tener en cuenta, el primer lugar del array **places**, este primer elemento será el que se mostrará en las cards informativas. Es posible que existan códigos postales que tengan mas de un elemento dentro de places, sin embargo, no se tendrán en cuenta.

También encontramos:

<i>state abbreviation</i>	Siglas de la comunidad autónoma a la que pertenece el lugar
<i>place name</i>	Nombre del lugar
<i>state</i>	Comunidad autónoma a la que pertenece el lugar
<i>latitude</i>	Coordenadas latitudinales del lugar
<i>longitude</i>	Coordenadas longitudinales del lugar

La relación entre el campo "state abbreviation" y cada comunidad son las siguientes:

AR	Aragón
AN	Andalucía
O	Asturias
IB	Baleares
CN	Canarias
S	Cantabria
CL	Castilla y León
CM	Castilla la Mancha
CT	Cataluña
VC	Comunitat Valenciana
EX	Extremadura
GA	Galicia
M	Madrid
MU	Murcia
NA	Navarra
PV	País Vasco
LO	La Rioja
CE	Ceuta
ML	Melilla

Servicio open meteo

El servicio open meteo puede utilizarse para obtener información sobre el clima de unas coordenadas en concreto.

La url para construir la llamada a este servicio es la siguiente:

https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&hourly=temperature_2m

Donde latitude y longitude son las coordenadas del lugar.

Por ejemplo:

https://api.open-meteo.com/v1/forecast?latitude=42.6&longitude=-5.5667&hourly=temperature_2m

La respuesta del servicio es la siguiente:

```
{
  "latitude": 42.625,
  "longitude": -5.5625,
  "generationtime_ms": 1.323103904724121,
  "utc_offset_seconds": 0,
  "timezone": "GMT",
  "timezone_abbreviation": "GMT",
  "elevation": 844.0,
  "hourly_units": {
    "time": "iso8601",
    "temperature_2m": "°C"
  },
  "hourly": {
```



```
"time": ["2022-11-02T00:00", "2022-11-02T01:00", "2022-11-02T02:00", "2022-11-02T03:00", "2022-11-02T04:00", "2022-11-02T05:00", "2022-11-02T06:00", "2022-11-02T07:00", "2022-11-02T08:00", "2022-11-02T09:00", "2022-11-02T10:00", "2022-11-02T11:00", "2022-11-02T12:00", "2022-11-02T13:00", "2022-11-02T14:00", "2022-11-02T15:00", "2022-11-02T16:00", "2022-11-02T17:00", "2022-11-02T18:00", "2022-11-02T19:00", "2022-11-02T20:00", "2022-11-02T21:00", "2022-11-02T22:00", "2022-11-02T23:00", "2022-11-03T00:00", "2022-11-03T01:00", "2022-11-03T02:00", "2022-11-03T03:00", "2022-11-03T04:00", "2022-11-03T05:00", "2022-11-03T06:00", "2022-11-03T07:00", "2022-11-03T08:00", "2022-11-03T09:00", "2022-11-03T10:00", "2022-11-03T11:00", "2022-11-03T12:00", "2022-11-03T13:00", "2022-11-03T14:00", "2022-11-03T15:00", "2022-11-03T16:00", "2022-11-03T17:00", "2022-11-03T18:00", "2022-11-03T19:00", "2022-11-03T20:00", "2022-11-03T21:00", "2022-11-03T22:00", "2022-11-03T23:00", "2022-11-04T00:00", "2022-11-04T01:00", "2022-11-04T02:00", "2022-11-04T03:00", "2022-11-04T04:00", "2022-11-04T05:00", "2022-11-04T06:00", "2022-11-04T07:00", "2022-11-04T08:00", "2022-11-04T09:00", "2022-11-04T10:00", "2022-11-04T11:00", "2022-11-04T12:00", "2022-11-04T13:00", "2022-11-04T14:00", "2022-11-04T15:00", "2022-11-04T16:00", "2022-11-04T17:00", "2022-11-04T18:00", "2022-11-04T19:00", "2022-11-04T20:00", "2022-11-04T21:00", "2022-11-04T22:00", "2022-11-04T23:00", "2022-11-05T00:00", "2022-11-05T01:00", "2022-11-05T02:00", "2022-11-05T03:00", "2022-11-05T04:00", "2022-11-05T05:00", "2022-11-05T06:00", "2022-11-05T07:00", "2022-11-05T08:00", "2022-11-05T09:00", "2022-11-05T10:00", "2022-11-05T11:00", "2022-11-05T12:00", "2022-11-05T13:00", "2022-11-05T14:00", "2022-11-05T15:00", "2022-11-05T16:00", "2022-11-05T17:00", "2022-11-05T18:00", "2022-11-05T19:00", "2022-11-05T20:00", "2022-11-05T21:00", "2022-11-05T22:00", "2022-11-05T23:00", "2022-11-06T00:00", "2022-11-06T01:00", "2022-11-06T02:00", "2022-11-06T03:00", "2022-11-06T04:00", "2022-11-06T05:00", "2022-11-06T06:00", "2022-11-06T07:00", "2022-11-06T08:00", "2022-11-06T09:00", "2022-11-06T10:00", "2022-11-06T11:00", "2022-11-06T12:00", "2022-11-06T13:00", "2022-11-06T14:00", "2022-11-06T15:00", "2022-11-06T16:00", "2022-11-06T17:00", "2022-11-06T18:00", "2022-11-06T19:00", "2022-11-06T20:00", "2022-11-06T21:00", "2022-11-06T22:00", "2022-11-06T23:00", "2022-11-07T00:00", "2022-11-07T01:00", "2022-11-07T02:00", "2022-11-07T03:00", "2022-11-07T04:00", "2022-11-07T05:00", "2022-11-07T06:00", "2022-11-07T07:00", "2022-11-07T08:00", "2022-11-07T09:00", "2022-11-07T10:00", "2022-11-07T11:00", "2022-11-07T12:00", "2022-11-07T13:00", "2022-11-07T14:00", "2022-11-07T15:00", "2022-11-07T16:00", "2022-11-07T17:00", "2022-11-07T18:00", "2022-11-07T19:00", "2022-11-07T20:00", "2022-11-07T21:00", "2022-11-07T22:00", "2022-11-07T23:00", "2022-11-08T00:00", "2022-11-08T01:00", "2022-11-08T02:00", "2022-11-08T03:00", "2022-11-08T04:00", "2022-11-08T05:00", "2022-11-08T06:00", "2022-11-08T07:00", "2022-11-08T08:00", "2022-11-08T09:00", "2022-11-08T10:00", "2022-11-08T11:00", "2022-11-08T12:00", "2022-11-08T13:00", "2022-11-08T14:00", "2022-11-08T15:00", "2022-11-08T16:00", "2022-11-08T17:00", "2022-11-08T18:00", "2022-11-08T19:00", "2022-11-08T20:00", "2022-11-08T21:00", "2022-11-08T22:00", "2022-11-08T23:00"],  
"temperature_2m": [3.7, 3.3, 3.0, 2.4, 2.0, 1.7, 1.6, 1.7, 3.2, 6.5, 8.9, 10.6, 11.6, 12.3, 12.7, 12.5, 11.8, 10.9, 10.0, 8.8, 7.9, 7.5, 6.9, 6.5, 6.6, 6.9, 7.5, 7.8, 8.0, 8.1, 8.2, 8.5, 8.7, 9.2, 9.5, 10.6, 11.7, 12.9, 14.2, 13.2, 11.6, 10.1, 8.3, 6.9, 5.8, 4.8, 3.6, 2.8, 2.4, 2.0, 1.7, 1.5, 1.4, 1.4, 1.5, 2.0, 2.7, 5.2, 7.9, 9.2, 10.1, 11.3, 12.5, 12.2, 11.8, 10.1, 8.1, 5.9, 4.5, 4.0, 3.4, 2.8, 2.5, 1.8, 1.0, 0.3, 0.1, -0.2, -0.4, 0.5, 1.9, 4.1, 5.9, 7.9, 10.0, 10.9, 11.3, 11.3, 10.9, 10.1, 9.1, 8.7, 8.3, 7.9, 7.7, 7.4, 7.3, 7.3, 7.4, 7.5, 7.5, 7.5, 7.6, 7.7, 7.8, 8.4, 9.5, 10.9, 12.5, 13.3, 13.8, 14.0, 13.6, 12.9, 11.8, 10.7, 9.4, 8.0, 7.5, 7.3, 7.2, 6.4, 6.2, 6.1, 6.2, 6.3, 6.7, 7.2, 7.7, 8.5, 9.1, 9.8, 10.6, 11.1, 11.7, 12.2, 12.4, 12.4, 12.5, 12.6, 12.7, 12.7, 12.6, 12.5, 12.2, 11.9, 11.6, 11.3, 11.0, 10.7, 10.6, 10.7, 10.9, 11.2, 11.6, 12.0, 12.5, 12.9, 13.1, 13.3, 13.0, 12.5, 12.0, 11.8, 11.7, 11.5, 11.4, 11.3]  
}
```

Los datos más importantes que hay que tener en cuenta, son los dos arrays que se obtienen dentro del objeto hourly. El array “time”, almacena las horas de cada una de las muestras de temperaturas del array “temperature_2m”.

Google maps

En este caso no se trata de un servicio REST, se trata de un servicio que nos permite mostrar una ubicación en el mapa de google maps a utilizando las coordenadas de la misma.

La url a construir para mostrar un lugar es la siguiente:

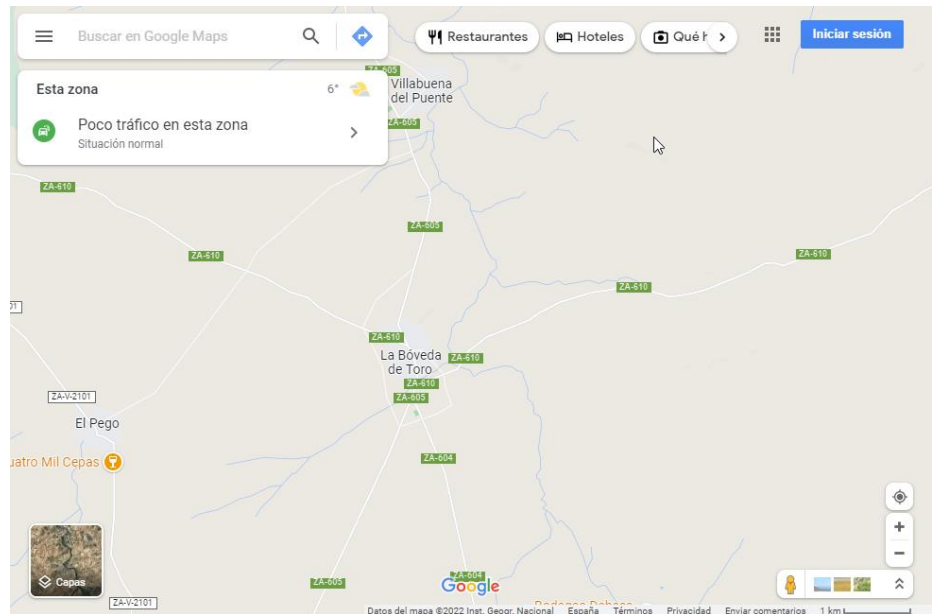
<https://www.google.com/maps/@{latitud},{longitud},{zoom}z>

Donde latitud y longitud son las coordenadas del lugar a mostrar y zoom indica el tamaño sobre el mapa. Para mostrar ciudades, un zoom de 13 puede ser adecuado.

Por ejemplo:

<https://www.google.com/maps/@41.35,-5.4,13z>

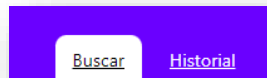
Como resultado se visualizará el mapa correspondiente a las coordenadas:



Desarrollo

La aplicación cuenta con dos pantallas, la pantalla de búsqueda para realizar búsquedas de códigos postales y la pantalla de historial para revisar las búsquedas realizadas.

Es posible cambiar entre una pestaña y otra utilizando la barra de navegación de la cabecera:



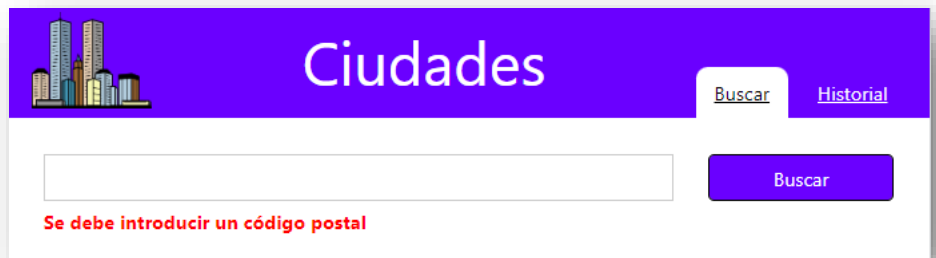
Pantalla de búsqueda

Esta pantalla es la pantalla principal, aparece al acceder a la aplicación y permite insertar un código postal y buscar la información.



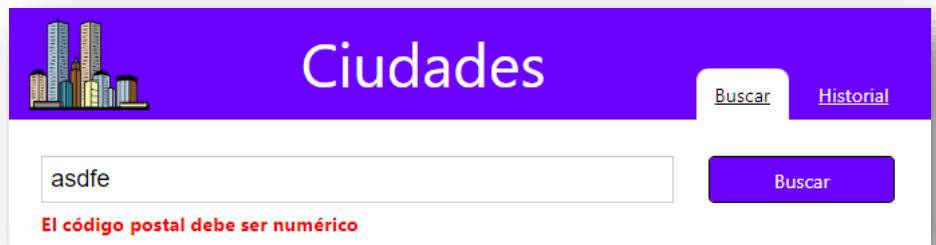
En el momento de buscar el código postal, es necesario que la aplicación valide si el formato es correcto, para evitar llamadas innecesarias al servicio. En el caso de que no lo sea hay que informar al usuario del error:

- Código postal vacío:



The screenshot shows the 'Ciudades' application interface. It has a purple header with a city skyline icon on the left, the title 'Ciudades' in the center, and two links 'Buscar' and 'Historial' on the right. Below the header is a white input field for the postal code. The field is empty. Below the input field, a red error message reads: 'Se debe introducir un código postal'. To the right of the input field is a purple button labeled 'Buscar'.

- Código no numerico:




The screenshot shows the 'Ciudades' application interface. The input field now contains the text 'asdfe'. Below the input field, a red error message reads: 'El código postal debe ser numérico'. The rest of the interface, including the header and the 'Buscar' button, remains the same.

- Código con longitud incorrecta:



The screenshot shows the 'Ciudades' application interface. The input field now contains the text '123456'. Below the input field, a red error message reads: 'El código postal debe tener 5 dígitos'. The rest of the interface, including the header and the 'Buscar' button, remains the same.

Una vez que el código introducido es correcto, se realizará la petición al servicio zipopotam. En este punto, es posible que el código postal no exista, por lo que se deberá indicar que no se ha encontrado.



Ciudades

[Buscar](#)
[Historial](#)

Sin resultados

En caso de que el código sea correcto, se mostrarán las cards de información correspondientes.



Ciudades

[Buscar](#)
[Historial](#)

Información política



Ciudad: León
Comunidad: Castilla - Leon

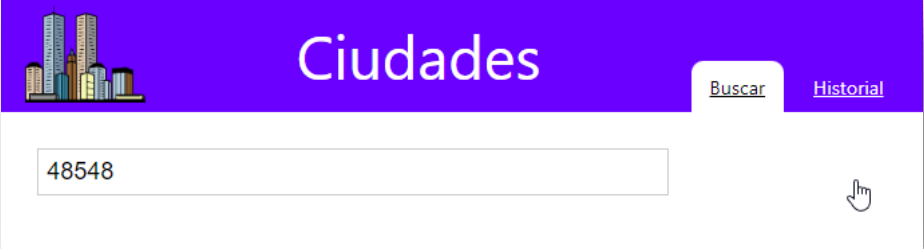
Información climática

Información geográfica

Latitud: 42.6
Longitud: -5.5667


[Ver mapa](#)

Al pinchar el botón buscar, se realizará una llamada al servicio zipopotam y **durante el transcurso de la misma, el botón buscar debe ser deshabilitado**, para evitar una acumulación de llamadas si el usuario pincha sobre el mismo en numerosas ocasiones:



The screenshot shows a web interface with a purple header. On the left of the header is a city skyline icon. The title 'Ciudades' is centered in the header. On the right, there are two buttons: 'Buscar' (highlighted) and 'Historial'. Below the header is a white search bar containing the text '48548'. To the right of the search bar is a hand cursor icon, indicating that the search bar is active or clickable.

Cards informativas

Todas las cards obtenidas muestran el mismo aspecto. Se trata de un pequeño recuadro que puede ser redimensionado para ocultar su contenido, utilizando el botón de la parte superior derecha.

Como se puede ver en la siguiente imagen las dos primeras cards se encuentran colapsadas, mientras que la última se encuentra expandida.

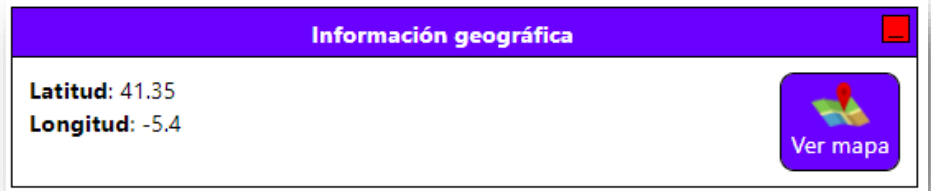


La primera card que encontramos tiene que ver con la información política del lugar asociado al código postal buscado. En ella encontramos además, la imagen de la comunidad autónoma a la que pertenece. En la carpeta de recursos de la práctica, podemos encontrar todas las imágenes necesarias para su construcción, incluidas las banderas a mostrar en esta card.

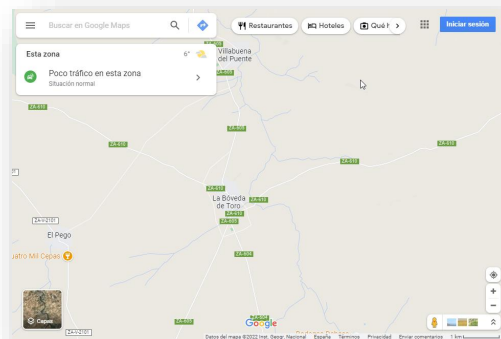
En la parte derecha de la card, se muestra el nombre de la ciudad y el nombre de la comunidad autónoma.



La siguiente card muestra la latitud y la longitud del lugar, además de un icono para ver las coordenadas en el mapa utilizando el servicio google maps.



Al hacer click sobre el icono se debe abrir el mapa en **una nueva pestaña del navegador**.



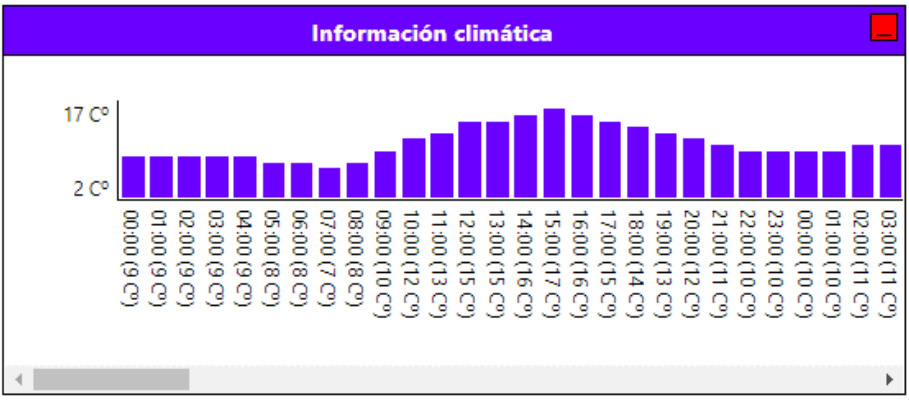
La última card muestra información sobre el clima del lugar, la construcción de ésta propone un grado más de dificultad, pues será necesario el uso del servicio "open meteo" para obtener la información relacionada en el momento en el que la card sea renderizada, es decir, **es la propia card la que debe hacer uso del servicio**.

En la card podemos encontrar una gráfica, con una leyenda vertical que muestra la temperatura máxima y la temperatura mínima. Este es el rango de temperaturas que debe abarcar la gráfica.

En la leyenda horizontal aparece cada una de las horas obtenidas y la temperatura correspondiente.

Finalmente en el interior de la gráfica se muestra una barra que relaciona cada hora con su temperatura.

La información climática puede contener un amplio rango de horas, por lo que el ancho de la gráfica habitualmente va a superar el ancho de la card, por ello, la card deberá ser capaz de hacer scroll de forma horizontal.

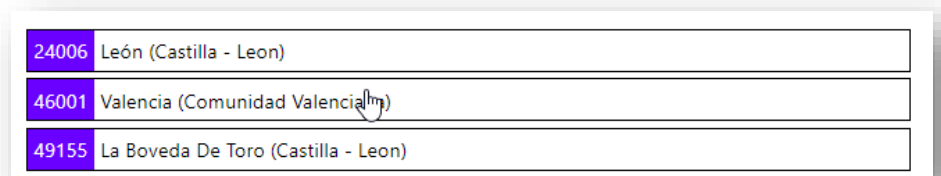


Pantalla de historial

El propósito de esta pantalla es mostrar un historial de búsqueda, este historial debe actualizarse con cada ciudad válida que se obtenga a través del buscador.



Al hacer click los elementos del historial, se debe mostrar la página de búsqueda con el código postal del clickado.



Buenas prácticas

Hooks para llamadas a servicios

La utilización de hooks para la realización de llamadas a servicios mejora el código de forma considerable, evitando que la lógica relacionada con la llamada se encuentre “mezclada” con la lógica de presentación propia de los componentes y pueda ser reutilizada. Además, permite que el mantenimiento sea más sencillo, y las modificaciones relacionadas con la llamada al servicio se centralicen en un solo punto. Para esta práctica, se realizan dos llamadas a servicios (zipopotam y openmeteo), por lo que **se recomienda que estas llamadas se encuentren separadas de la lógica de presentación utilizando hooks.**

Contextos globales

La creación de contextos globales en ocasiones reduce la complejidad de la aplicación, evitando el paso innecesario de datos entre componentes. En este caso, para la implementación del historial de búsquedas, **se recomienda la creación de un contexto global.** De esta forma es posible acceder al mismo desde cualquier punto de la aplicación, ya sea para añadir nuevos elementos, como para obtener los elementos almacenados.

Parametrización

Una forma de mejorar el mantenimiento de las aplicaciones es la parametrización de las mismas. En aplicaciones React, se puede realizar esta parametrización utilizando archivos JSON. Por ejemplo, algo que siempre es recomendable separar de la aplicación son los textos que se utilizan en la misma. De esta forma, es posible realizar modificaciones de una forma fácil e incluso traducir por completo la aplicación a otros idiomas.

También es interesante separar otro tipo de configuraciones de la aplicación, que son propensas a sufrir cambios, para que puedan ser ajustadas posteriormente.

Refactorización de componentes

La creación de componentes simples permite una mejor lectura del código y mejora la reutilización de ciertas partes del mismo. Es importante identificar que componentes pueden ser divididos en subcomponentes, y realizar refactorizaciones de estos si es necesario. Un componente con mucho código no es un componente de calidad y se recomienda evitarlo si es posible.