En la siguiente prueba tecnica esperamos poder conocer como el desarrollador aborda ó toma desde una maqueta hecha en html y la procesa para generar un WebApp hecha en React

Primer Ejercicio

Assets

Maqueta: http://especiales.lanacion.com.ar/arc-css/acumulado.asp

Endpoint: https://api-test-ln.herokuapp.com/articles [get method available]

Objetivos

- A partir del tag main de html, componetizar todos los elementos que considere necesario.
- El endopoint provisto contiene un json con lo que representarian 30 notas o articulos. con estos datos se deben alcanzar los siguientes objetivos:
    - Agrupar, totalizar y ordenar de mayor a menor los tags encontrados en las notas
        - los tags seran encontrados en cada articulo dentro del atributo “taxonomy.tags”
    - Del ordenamiento anterior mostrar los primeros 10 tags debajo del titulo
    - El href a contener debe ser: “/tema/[tag.slug]”
    - Mostrar los 30 articulos en la grilla de articulos
        - listar solo articulos con el "subtype":"7",
        - la fecha a mostrar debe ser la de display_date
        - el formato de la fecha debe salir como sale en la maqueta
        - el titulo a mostrar se encuentra en en el atributo “headline.basic”
        - tomar de imagen para el articulo la que sale en el “promo_items.url”

Se espera observar:

Una web app hecha en React from Scratch

Debe cumplir con Server Side Rendering

Repositorio con el codigo en GitHub

Implementar Hooks

Usar solo React y/o Redux si lo considera necesario para manejo de estados entre componentes, evitando usar dependencias de terceros

Será un plus si agrega (aunque es opcional)

Uso de programación declarativa

Para manejo de estado entre componentes, se valorará Context API de React .

Implementar una API Client patterns para consultas del Endpoint.

Implementar Async/Await para la consulta de los datos remotos

Children Props

Implementacion de cache del lado del cliente

Test unitarios o implementar TDD

Usar docker o bien implementar el codigo resultante en alguno Paas gratuito como heroku o now.sh, etc.

Segundo ejercicio

Se debe entregar el código fuente (o repo git) junto con la documentación que crea necesaria para comunicar el diseño, si lo considera útil, también se pueden agregar test unitarios.

Escriba un script que a partir de un array de ints devuelva un array de strings aplicando las siguientes reglas:

- Devuelve Fizz si el número es divisible por 3 o si incluye un 3 en el número.
- Devuelve Buzz si el número es divisible por 5 o si incluye un 5 en el número.
- Devuelve FizzBuzz si el número es divisible por 3 y por 5.
- Puedes utilizar cualquier lenguaje que consideres apropiado.

Algunas preguntas:

- ¿De qué forma guardarías los archivos que un usuario suba por la aplicación al servidor y porque?
- ¿Implementaría un cache del lado del cliente? ¿Porque?