Facultad de Informática
Programación Web Avanzada

Trabajo Práctico: REACT Parte II
Desarrollo de una Aplicación con Múltiples Páginas

Objetivo del Proyecto:

Desarrollar una SPA con React y Tailwind que consuma datos desde una API simulada, que incluya navegación entre páginas, búsqueda, idioma en español e inglés, y persistencia de preferencias del usuario. Se tendrá una página de home donde muestre una lista de elementos en forma de cards o tarjetas y permita al usuario ver detalles adicionales al hacer click en un elemento. También se desarrollará una página de elementos favoritos del usuario.

Requerimientos de la Aplicación:

La aplicación puede ser de cualquier dominio que los alumnos deseen desarrollar. Lo importante es que el dominio seleccionado sea útil para cumplir con los requerimientos de este documento.
Ejemplos de dominios: Películas, videojuegos, juegos de mesa, equipos de fútbol, grupos musicales, tecnologías, artistas.

La aplicación necesita ser multi idioma (Español/Inglés). Para esto usar react-i18next. Es necesario almacenar la preferencia del idioma en localStorage.

Navegación y Estructura:

La aplicación debe incluir un header, un footer, un input de búsqueda y un componente Home.
Debe tener además una segunda página, la de detalles que se cargue al hacer clic en un elemento de la lista del Home.
El usuario debe ser capaz de seleccionar los elementos como favoritos por medio de algún botón o icono y esta lista de favoritos debería ser visible desde una tercera página “Favoritos”. Utilizar Local Storage para almacenar y persistir los elementos favoritos del usuario.
Implementar rutas con react-router-dom para navegar entre la página de inicio, la página de detalles y la página de favoritos.

Datos Mockeados con API Simulada

- Usar la herramienta MockAPI para simular una API REST.
- Endpoints:
  - `/items`: listado principal. (siendo items el elemento de su dominio)
  - `/items/:id`: detalle individual.

Ejemplo para dominio de juegos de Mesa:

`GET /juegos`

```
[
  {
    "id": 1,
    "name": "Catan",
    "description": "Juego de estrategia donde los jugadores construyen
pueblos y rutas.",
    "category": "Estrategia",
    "coverImage": "https://example.com/images/catan-cover.jpg"
  },
  {
    "id": 2,
    "name": "Dixit",
    "description": "Juego de cartas con ilustraciones oníricas para
adivinar historias.",
    "category": "Party Game",
    "coverImage": "https://example.com/images/dixit-cover.jpg"
  }
]
```

`GET /juegos/1`

```
{
  "id": 1,
  "name": "Catan",
  "description": "Catan es un juego de mesa de negociación y estrategia
en el que los jugadores recolectan recursos para construir caminos,
pueblos y ciudades.",
  "category": "Estrategia",
  "images": [
```

```
    "https://example.com/images/catan1.jpg",
    "https://example.com/images/catan2.jpg"
  ],
  "rules": "https://example.com/rules/catan.pdf",
  "publisher": "Kosmos",
  "age_range": "10+"
}
```

Funcionalidad Header:

El header debe tener al menos un ícono de 'home' o el logo de su aplicación para que al ser clickeado se retorne a la página de home. También incluir alguna forma de navegación a la página de favoritos. Puede incluir otras imágenes, textos o descripciones que crean necesarias.
Debe colocarse en la parte superior de la aplicación y debe estar visible en todas las páginas. Si la pagina tuviera un scroll vertical, el header siempre tiene que estar en la parte superior de manera visible (Stiky).
También debe incluir el selector de idioma (Español/Inglés).

Funcionalidad del input de búsqueda:

Debe haber un input text que permita funcionar de buscador entre los elementos. Dependiendo el dominio va a tener sentido hacer búsqueda por un campo u otro. Por ejemplo en nuestro dominio de juegos de mesa se podría hacer una búsqueda por nombre juego de mesa, mientras que en otro dominio podría ser más adecuado buscar por algún otro campo.
Si no hay elementos para mostrar, dar un mensaje al usuario de que no se encontraron elementos para su búsqueda.
Analizar si es necesario hacer una llamada a un endpoint con el campo de búsqueda o no.

Funcionalidad de la Lista:
Realizar una llamada a una Fetch a la api simulada de `MockAPI` en el componente Home para obtener todos los datos necesarios para la lista.

Mostrar los elementos de la lista en cards o tarjetas, con imágenes y descripciones

breves obtenidas desde el json.
Al hacer click en un elemento de la lista, redirigir al usuario a una página de detalles del elemento clickeado.

Funcionalidad Footer:
Incluir información ficticia de la aplicación, como redes sociales, dirección, datos de contacto y cualquier otra descripción o información que crean necesarias. Debe colocarse en la parte más baja de la aplicación y debe estar visible en ambas páginas.

Página de Detalles:
La página de detalles debe recibir el ID del elemento a través de la URL. Realizar una llamada Fetch a la URL de la api correspondiente para obtener más información basada en el ID del elemento.

Mostrar información detallada del elemento seleccionado, incluyendo descripciones adicionales.

Si se ingresa un id en la URL que no existe en la API deberá mostrar una página de error 404.

Página de Favoritos:
La página de favoritos debe consumir la lista de elementos favoritos del usuario desde el local storage.
Si no tiene ningún favorito, mostrar un mensaje acorde.

Diseño y Estilo:
Utilizar Tailwind CSS para aplicar estilos a la aplicación. Asegurar que todos los componentes utilicen clases de Tailwind para manejar layout (flexbox o grid), espaciado (padding y margin), tipografía, colores de fondo y de texto, bordes, sombras, y transiciones para interacciones del usuario (como hover). No hay restricciones de diseño ni como tiene que lucir estéticamente la aplicación.

Gestión de Estado:
Utilizar useState para manejar el estado de la lista de elementos y los detalles del elemento seleccionado.
Considerar el uso de useEffect para las llamadas a Fetch.
La aplicación deberia guardar el lenguaje seleccionado por el usuario en Local Storage, de tal manera que si se refresca la pagina, se continue con el mismo lenguaje

Facultad de Informática
Programación Web Avanzada

Estructura de Directorios:

Estructura de archivos y carpetas recomendada:

```
-Pages
        ---Home
                ---Home.js
                —Home.module.css
        —Details
                ---Details.js
                —Details.module.css

-Components
        ---Button
                ---Button.js
                —Button.module.css
```

```
        ---OtroComponente
                ---OtroComponente.js
                —OtroComponente.module.css

-services
    getAllXYZ.ts
    getXYZBYId.ts
```

Instrucciones para el Desarrollo:

1) Elegir un dominio adecuado para resolver la consigna. Es totalmente libre de elección.
2) Crear un proyecto react y agregarlo a un repositorio de Github o crear un repositorio en github, clonarlo y luego crear el proyecto.
    a) Compartir el link del repositorio con los profesores en la planilla de los grupos.

3) Instalar Tailwind CSS y configurarlo dentro del proyecto.
    a) https://tailwindcss.com/docs/installation

4) Desarrollar la aplicación siguiendo los requerimientos.
5) Analizar y crear los componentes que sean necesarios para cumplir los objetivos.
6) Documentar decisión importante de diseño o código en el readme.md.

Opcional: Utilizar Trello para la organización del proyecto dentro del grupo por medio de tarjetas: https://trello.com/es

Opcional 2: Agregar un botón en la página de detalles para exportar la información en un pdf descargable. Buscar una librería de terceros que ayude en esta tarea.

Opcional 3: Paginar los endpoints para implementar un scroll infinito en el home.

Opcional 4: Hacer las vistas responsive para dispositivos móviles.

Readme.md:

1) Actualizar el archivo readme.md para tener una documentación adecuada:
   a) Armar una carátula con los datos de los miembros del grupo.
   b) Incluir una descripción básica de la aplicación.
   c) Incluir una guía e instrucciones de instalación paso a paso (clonar el repositorio - correr el comando `npm i`…)
   d) Agregar capturas de pantalla.
   e) Cualquier otra información que crean relevante.