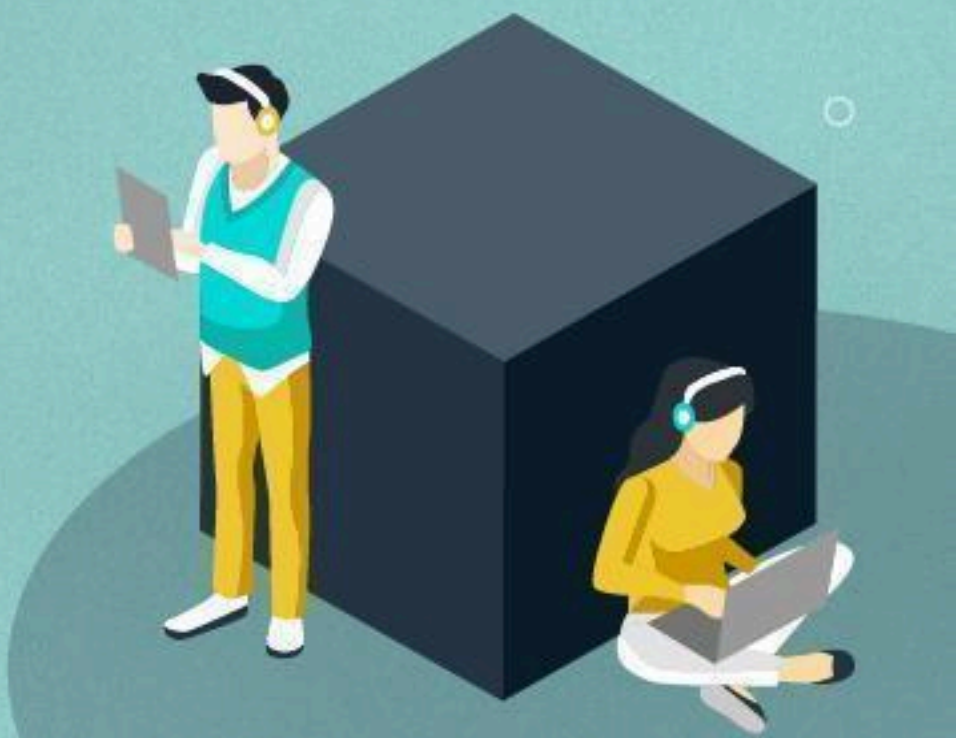
«Talento Tech»

Desarrollo Web 3

Clase 10



Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 10

Temario:

- Creación de Aplicación de Gestión de Usuarios - Parte 3

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Creación de Aplicación de Gestión de Usuarios - Parte 3

Tercera Parte: Eliminación y Context API

En esta fase, concluiremos con el proyecto que fuimos trabajando la segunda parte del curso de DW3. Hoy haremos uso del **Context API** y del método **DELETE** para completar el **CRUD** de nuestra App.

1. **Crea un componente UserDelete.jsx para la confirmación de eliminación**

```
import React from 'react';
import { useParams, useNavigate } from 'react-router-dom';
const UserDelete = () => {
  const { id } = useParams();
const navigate = useNavigate();
  const handleDelete = async () => {
    try {
      const response = await fetch(`URL_DE_TU_API/${id}`, {
        method: 'DELETE',
      });

      if (response.ok) {
       navigate('/');
      } else {
        console.error('Error al eliminar usuario');
      }
    } catch (error) {
      console.error('Error en la solicitud: ', error);
    }
  };
  return (
    <div>
      <h1>Eliminar Usuario</h1>
      <p>¿Estás seguro de que deseas eliminar este usuario?</p>
```

```
      <button onClick={handleDelete}>Sí, Eliminar</button>
    </div>
  );
};
export default UserDelete;
```

2. Define el contexto para la obtención de datos (READ)

Crea un archivo UserContext.jsx para el contexto. En esta oportunidad estaremos solo haciendo el verbo READ. Si te interesa aplicar el resto del CRUD podes seguir trabajando a partir de este código que te presentamos

```
import React, { createContext, useState, useEffect, useContext } from 'react';
const UserContext = createContext();
export const UserProvider = ({ children }) => {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  useEffect(() => {
    const fetchUsers = async () => {
      try {
        const response = await fetch('URL_DE_TU_API');
        const data = await response.json();
        setUsers(data);
      } catch (error) {
        setError(error);
      } finally {
        setLoading(false);
      }
    };
    fetchUsers();
  }, []);
  return (
    <UserContext.Provider value={{ users, loading, error }}>
```

```
    {children}
   </UserContext.Provider>
  );
};
export const useUserContext = () => useContext(UserContext);
```

3. Provee el contexto en el nivel más alto de la aplicación (App.jsx):

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import { UserProvider } from './UserContext';
import UserList from './UserList';
import UserDetails from './UserDetails';
import UserForm from './UserForm';
import UserEdit from './UserEdit';
import UserDelete from './UserDelete';
const App = () => {
  return (
    <UserProvider>
      <Router>
        <Routes>
          <Route path="/" element={<UserList />} />
          <Route path="/users/:id" element={<UserDetails />} />
          <Route path="/create" element={<UserForm />} />
          <Route path="/edit/:id" element={<UserEdit />} />
          <Route path="/delete/:id" element={<UserDelete />} />
        </Routes>
      </Router>
    </UserProvider>
  );
};
export default App;
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

4. Utiliza el contexto en donde se ubica la lista de usuarios:

```
import React from 'react';
import { Link } from 'react-router-dom';
import { useUserContext } from './UserContext';
const UserList = () => {
  const { users, loading, error } = useUserContext();
  if (loading) {
    return <div>Cargando...</div>;
  }
  if (error) {
    return <div>Error al cargar los usuarios: {error.message}</div>;
  }
  return (
    <div>
      <h1>Lista de Usuarios</h1>
      <Link to={`/create`}>Agregar Usuario</Link>
      {/* Mostrar lista de usuarios */}
      <ul>
        {users.map((user) => (
          <li key={user.id}>
            <Link to={`/users/${user.id}`}>{user.name}</Link>
          </li>
        ))}
      </ul>
    </div>
  );
};
export default UserList;
```

Con estos pasos, deberás tener una aplicación funcional de gestión de usuarios que implementa CRUD, navegación y uso de la Context API con React Router DOM 6.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desafío #4

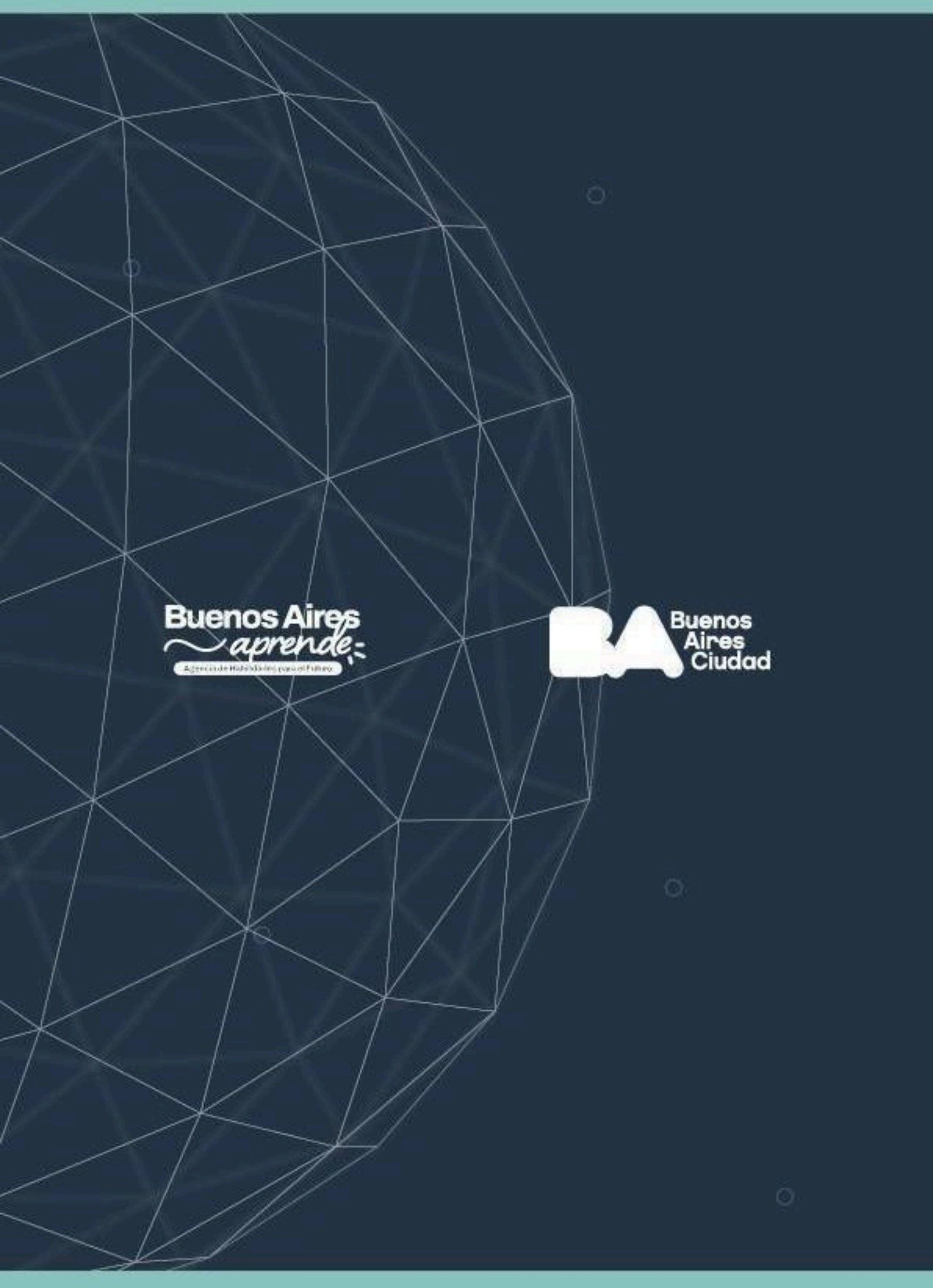
¿Cómo entrego el desafío🤔?

Deberás subir tu proyecto a Github y mandar el link del repositorio para que podamos ver tu trabajo.

Atención: La carpeta node_modules debe ser eliminada antes de subir el proyecto funcional a Github. Esto se debe a que es una carpeta muy pesada, y si queremos probar tu proyecto, solamente clonamos lo demás y usando ‘npm i’ se instala exactamente la misma carpeta.

¡Con esto concluye tu recorrido por DW3! Si llegaste hasta acá, contás con dos proyectos y un repositorio para poder seguir subiendo tus trabajos para en un futuro usarlo como portfolio 😎.

Esperamos que hayas aprendido un montón y esperamos encontrarte nuevamente en otro curso de AP. ¡Gracias! 😁✌️


Buenos Aires
aprende
Buenos
Aires
Ciudad