



PÁGINA WEB

Yeiner Castro¹, Hermes Martinez², Luis Marlon Casallas³, Carlos Cadena⁴, Deivid Hernandez⁴

1. Cod: 160004204, Ing. Sistemas,
2. Cod: 160004216, Ing. Sistemas.
3. Cod: 160004251, Ing. Sistemas.
4. Cod: 160004240, Ing. Sistemas.
5. Cod: 160004211, Ing. Sistemas.

Facultad de Ciencias Básicas e Ingenierías.
Programa

Resumen

En el siguiente informe se podrá entender el funcionamiento de la página web. Se desarrolló una página web dinámica utilizando React y Python. Se implementó un sistema de inicio de sesión y un cuadro de búsqueda autocompletado basado en las preferencias del usuario. Se utilizó el algoritmo de PageRank para mejorar la relevancia de los resultados de búsqueda.

PageRank

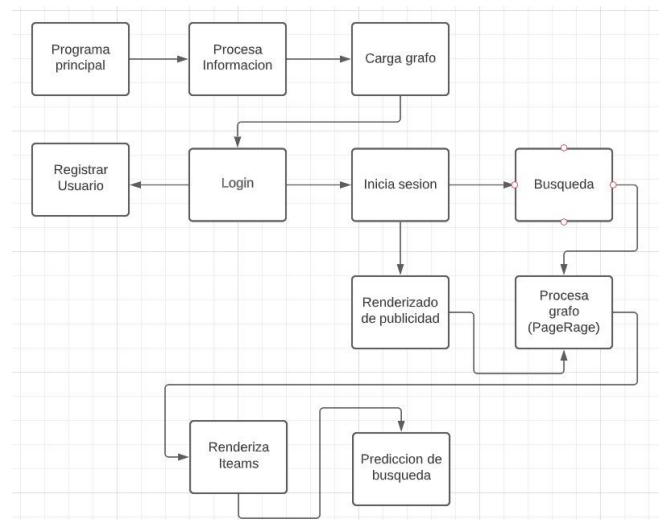
PageRank es un algoritmo desarrollado por Google para clasificar la importancia de las páginas web. Utiliza enlaces entrantes como votos de confianza. Cuantos más enlaces de calidad tenga una página, mayor será su puntuación de PageRank. La fórmula de PageRank es:

$$PR(A) = (1 - d) + d \sum_{i=1}^n \frac{PR(i)}{C(i)}$$

Donde:

- **PR(A)** Es el PageRank de la página A.
- **d** es un factor de amortiguación que tiene un valor entre 0 y 1.
- **PR(i)** son los valores de PageRank que tienen cada una de las páginas **i** que enlazan a A.
- **C(i)** es el número total de enlaces salientes de la página **i** (sean o no hacia A).

Resultados

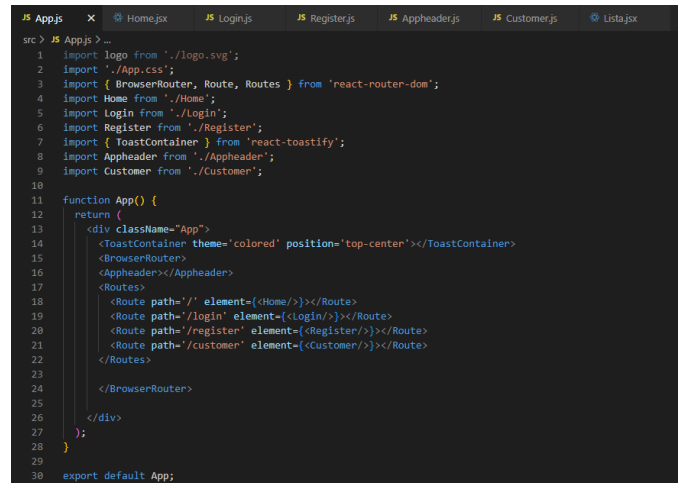


El código proporcionado consiste en una aplicación de React que consta de varios archivos. A continuación, se presenta un resumen de cada archivo y su funcionalidad principal (Front End):

- 1) App.js: Este archivo es el componente principal de la aplicación. Define las rutas de navegación

utilizando React Router y muestra el encabezado de la aplicación. También utiliza el componente ToastContainer de react-toastify para mostrar notificaciones.

- 2) Home.jsx: Este componente representa la página de inicio. Comprueba si el usuario ha iniciado sesión y muestra el nombre de usuario en la página. También muestra una lista de clientes utilizando el componente Lista.
- 3) Login.js: Este componente muestra un formulario de inicio de sesión. Cuando se envía el formulario, se valida el nombre de usuario y la contraseña ingresados. Si son válidos, se establece la información de inicio de sesión en el almacenamiento de sesión y se redirige al usuario a la página de inicio. De lo contrario, se muestra un mensaje de error.
- 4) Register.js: Este componente muestra un formulario de registro de usuario. Cuando se envía el formulario, se validan los campos y se envía una solicitud POST para registrar al usuario. Si el registro es exitoso, se muestra un mensaje de éxito y se redirige al usuario a la página de inicio de sesión.
- 5) Appheader.js: Este componente muestra un encabezado común en la parte superior de la página. Utiliza useLocation de react-router-dom para obtener la ubicación actual y decide si mostrar el encabezado o no en función de la ubicación. También muestra enlaces a las diferentes páginas de la aplicación, muestra el nombre de usuario y proporciona un enlace para cerrar sesión.
- 6) Customer.js: Este componente muestra una lista de clientes. Utiliza useEffect para cargar los datos de los clientes cuando se monta el componente. También verifica los permisos del usuario para acceder a esta página y muestra los botones de agregar, editar y eliminar clientes en función de los permisos del usuario.



```

src > App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3 import { BrowserRouter, Route, Routes } from 'react-router-dom';
4 import Home from './Home';
5 import Login from './Login';
6 import Register from './Register';
7 import { ToastContainer } from 'react-toastify';
8 import Appheader from './Appheader';
9 import Customer from './Customer';
10
11 function App() {
12   return (
13     <div className="App">
14       <ToastContainer theme="colored" position="top-center"></ToastContainer>
15       <BrowserRouter>
16         <Appheader></Appheader>
17         <Routes>
18           <Route path="/" element={Home}></Route>
19           <Route path="/login" element={Login}></Route>
20           <Route path="/register" element={Register}></Route>
21           <Route path="/customer" element={Customer}></Route>
22         </Routes>
23       </BrowserRouter>
24     </div>
25   );
26 }
27
28 export default App;
  
```

A continuación, se presenta un resumen de cada archivo y su funcionalidad principal (Back End):

- 1) BaseDatos.py: Este archivo define una clase llamada BaseDatos. El constructor de la clase inicializa un diccionario vacío y una variable json en None. El método genera_base_data genera datos de ejemplo para la base de datos, creando una lista de elementos de zapato con enlaces aleatorios. El método __str__ imprime los datos almacenados en la base de datos.
- 2) Controlador.py: Este archivo define una función llamada leer_actualizar_json, que recibe el nombre de un archivo como parámetro. La función carga y devuelve los datos almacenados en el archivo JSON especificado.
- 3) GenerarBase.py: En este archivo se encuentra una aplicación Flask. La función hello responde con un saludo que incluye el nombre proporcionado como argumento en la URL. La ruta /json-data utiliza la función leer_actualizar_json del archivo Controlador.py para cargar y devolver los datos JSON de un archivo cuyo nombre se proporciona como parámetro en la URL.
- 4) main.py: En este archivo se realiza la ejecución principal de la aplicación. Se crea una lista de personas y se inicializa una lista de usuarios, donde cada usuario tiene un nombre, una base de datos y un objeto de ranking. Se genera una base de datos y un ranking para cada usuario. Se agregan nodos y aristas al grafo de ranking según los datos de la base de datos. Se genera y se guarda un archivo

JSON con los resultados del ranking para cada usuario.

- 5) Ranking.py: Este archivo define una clase llamada Ranking. El constructor de la clase inicializa un grafo dirigido (DiGraph) utilizando la biblioteca networkx. La clase tiene métodos para generar el ranking de los elementos del grafo utilizando el algoritmo PageRank, agregar valores al ranking según las búsquedas de los usuarios y generar un archivo JSON con los resultados del ranking.

```
BaseDatos.py | Controlador.py | GenerarBase.py | main.py | Ranking.py
pythonSrc > Backen > Estocasticos2 > BaseDatos.py
1 import random
2 class BaseDatos:
3     def __init__(self):
4         self.data = {}
5         self.json=None
6
7     def genera_base_data(self) -> dict:
8         for i in range(1, 10):
9             zapato = f'zapato{i}'
10            enlaces = random.sample(range(1, 6), 3) # Generar 3 enlaces aleatorios para cada artículo
11            enlaces = [f'zapato{j}' for j in enlaces]
12            self.data[zapato] = {'enlaces': enlaces}
13
14    def __str__(self) -> str:
15        for zapato, info in self.data.items():
16            print(f'{zapato}: {info}')
```

- **Página Web:**

Registro de usuario:

User Registration

User Name *

Full Name *

Phone

Address

Gender

Chale@Female

Password *

Email *

Country *

India

Register

Cancel

Login:

User Login

User Name *

yeiner

Password *

Login

New User

Home:

