

Memoria

Experiencia de usuario - Buscador

Diego Martín Fernández

UO276345

Índice

Índice	1
Buscador	1
Informe sobre la Implementación de un Buscador Interno en un Sitio Web	2
1. Soluciones para Construir Buscadores Internos en un Sitio Web	2
2. Diseño de la Arquitectura de Software del Buscador	2
3. Batería de pruebas	4
4. Mejoras	6

Buscador

Página personal

Inicio

Curriculum

Sobre Mi

Buscar...

Buscar

Página Principal

Bienvenido a la página web personal de Diego Martín Fernández, ingeniero de software apasionado por la tecnología, la lectura y la música.

libros

Buscar

Página Principal - Diego Martín Fernández

sobre mí hola, soy diego martin fernández, un ingeniero de sof

rdf

Buscar

Curriculum - Diego Martín Fernández

mis proyectos - rdf in zarr proyecto de investigación sobre motores rdf usando tecnología de particionamiento de arrays en chunks - proyecto dataspace...

Informe sobre la Implementación de un Buscador Interno en un Sitio Web

1. Soluciones para Construir Buscadores Internos en un Sitio Web

Implementar un buscador interno eficaz en un sitio web es esencial para mejorar la experiencia del usuario y facilitar el acceso a la información. A continuación, se presentan algunas soluciones y consideraciones clave:

- **Uso de JavaScript y Archivos Estáticos:** Para sitios web con contenido estático, se puede utilizar JavaScript para crear un buscador que recorra los archivos locales, como se muestra en el código proporcionado. Este método es adecuado para sitios pequeños con contenido limitado.
- **Integración de Motores de Búsqueda de Terceros:** Servicios como Google Custom Search permiten incorporar un buscador potente sin necesidad de desarrollar uno desde cero. Estos servicios ofrecen funcionalidades avanzadas y son fáciles de implementar.
- **Desarrollo de Soluciones Personalizadas:** Para sitios con contenido dinámico o grandes volúmenes de información, es recomendable desarrollar un motor de búsqueda interno que indexe el contenido y proporcione resultados relevantes. Esto puede implicar el uso de bases de datos y algoritmos de búsqueda avanzados.
- **Uso de Plataformas de Gestión de Contenidos (CMS):** Sistemas como WordPress, Joomla o Drupal suelen ofrecer plugins o extensiones que facilitan la implementación de buscadores internos sin necesidad de codificación adicional.

[Coko54](#)

2. Diseño de la Arquitectura de Software del Buscador

El diseño de la arquitectura de un buscador interno debe considerar los siguientes componentes y procesos:

- **Indexación de Contenido:** El buscador debe ser capaz de recorrer y analizar el contenido del sitio web para crear un índice que facilite búsquedas rápidas y precisas.

- **Interfaz de Usuario (UI):** Se debe diseñar una interfaz intuitiva que permita a los usuarios ingresar consultas y visualizar resultados de manera clara y eficiente.
- **Procesamiento de Consultas:** El sistema debe interpretar las consultas de los usuarios, manejar sinónimos y errores tipográficos, y aplicar algoritmos de relevancia para ordenar los resultados.
- **Almacenamiento de Datos:** Es esencial contar con una base de datos o estructura de almacenamiento que soporte el índice de búsqueda y los metadatos asociados.
- **Actualización y Mantenimiento:** El sistema debe permitir la actualización periódica del índice para reflejar cambios en el contenido del sitio web.

Para una implementación sencilla, el código proporcionado utiliza un objeto JavaScript que simula un mapa del sitio (`sitemap.xml`) y permite realizar búsquedas en el contenido de las páginas definidas. Este enfoque es adecuado para sitios web pequeños con contenido estático.

Disclaimer, en mi página web el `sitemap.xml` es sustituido por esto por temas de fallos de CORS, por eso está hardcodeado:

```
// Definimos el contenido hardcodeado de sitemap.xml como un objeto en JS
const sitemap = {
  site: {
    page: [
      {
        url: 'index.html',
        title: 'Página Principal - Diego Martín Fernández',
        content: 'Sobre Mí Hola, soy Diego Martín Fernández, un ingenier
      },
      {
        url: 'curriculum.html',
        title: 'Currículum - Diego Martín Fernández',
        content: 'Mis proyectos - RDF in Zarr Proyecto de investigación
      },
      {
        url: 'sobre_mi.html',
        title: 'Sobre Mí - Diego Martín Fernández',
        content: 'Hola, soy Diego Martín Fernández, un ingeniero de soft
      }
    ]
  }
};
```

Para sitios más complejos, se recomienda utilizar herramientas y técnicas avanzadas, como el Lenguaje de Modelado Unificado (UML) para diagramar la arquitectura, y patrones de diseño establecidos que brinden soluciones comprobadas a problemas de diseño recurrentes.

[Appmaster](#)

En resumen, la implementación de un buscador interno efectivo requiere una planificación cuidadosa de la arquitectura de software, considerando aspectos como la indexación, la interfaz de usuario y el procesamiento de consultas, para garantizar una experiencia de búsqueda óptima para los usuarios.

3. Batería de pruebas

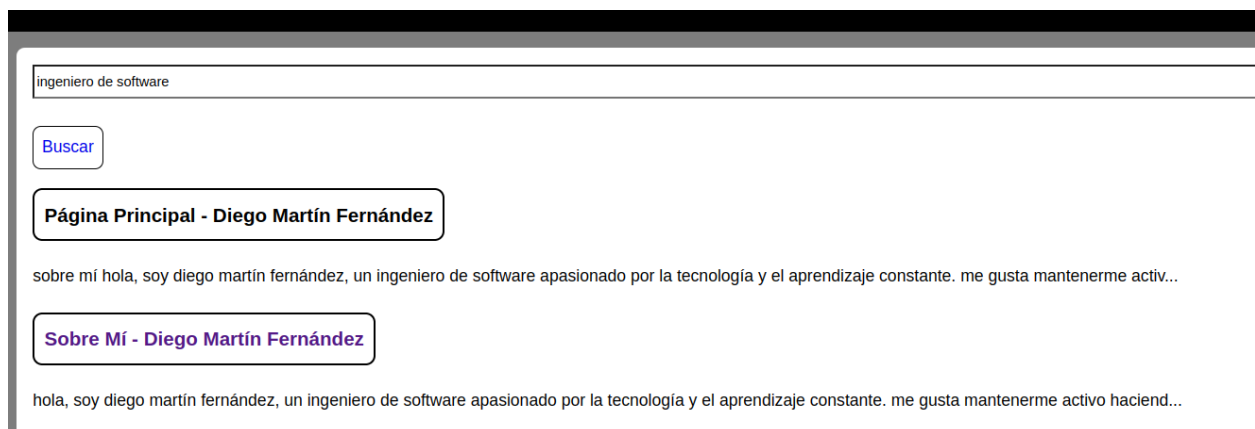
Prueba de Búsqueda Básica

Objetivo: Verificar que el buscador devuelve resultados precisos cuando se introduce un término de búsqueda.

Pasos:

- Acceder a la página principal del sitio.
- Ingresar el término de búsqueda "ingeniero de software".
- Hacer clic en el botón de búsqueda.
- Verificar que se muestren resultados relevantes relacionados con el término buscado.

Resultado Esperado: Se deben mostrar resultados de las páginas que contienen la palabra "ingeniero de software".



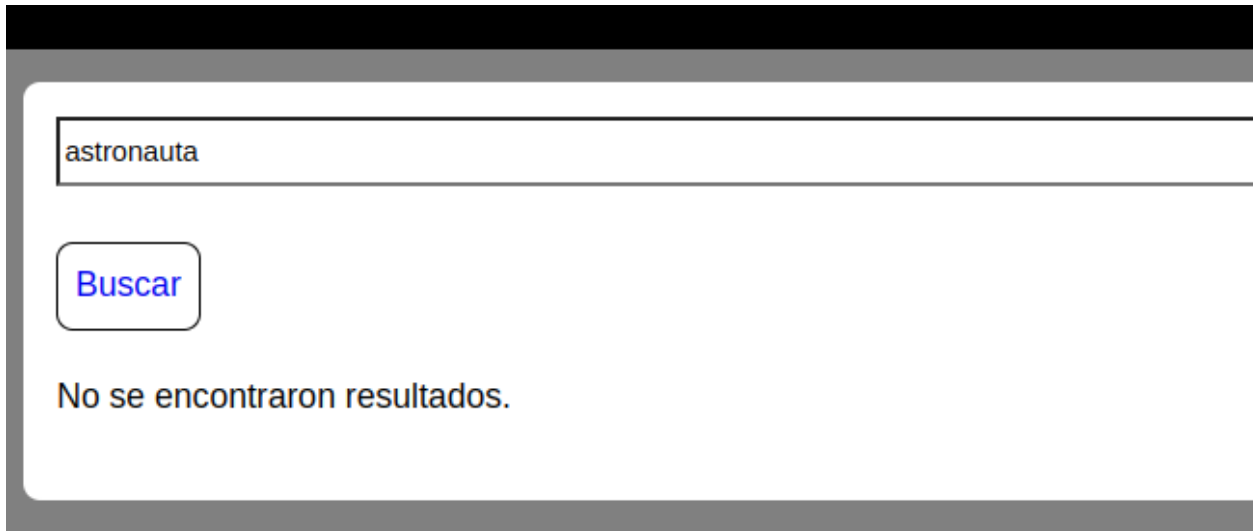
Prueba de Búsqueda Sin Resultados

Objetivo: Asegurar que el sistema maneje adecuadamente una búsqueda sin resultados.

Pasos:

- Ingresar un término de búsqueda no relacionado, como "astronauta".
- Hacer clic en el botón de búsqueda.
- Verificar que el mensaje "No se encontraron resultados." se muestre en el área de resultados.

Resultado Esperado: El sistema debe mostrar un mensaje indicando que no hay coincidencias.



astronauta

Buscar

No se encontraron resultados.

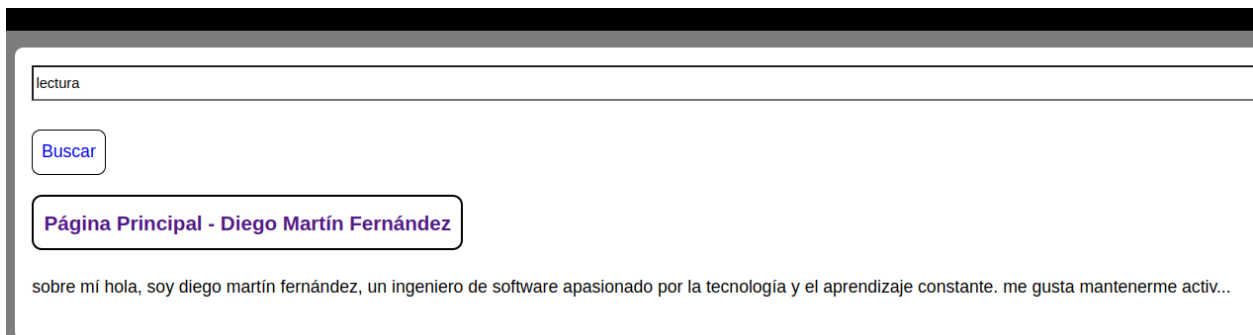
Prueba de Búsqueda por Fragmentos de Texto

Objetivo: Comprobar que el sistema sea capaz de mostrar fragmentos de contenido de las páginas que contienen el término buscado.

Pasos:

- Buscar el término "lectura".
- Verificar que, junto con el título de la página, se muestre un fragmento de contenido donde se mencione "lectura".

Resultado Esperado: El sistema debe mostrar un fragmento que contenga la palabra "lectura" junto con el enlace correspondiente.



lectura

Buscar

Página Principal - Diego Martín Fernández

sobre mí hola, soy diego martín fernández, un ingeniero de software apasionado por la tecnología y el aprendizaje constante. me gusta mantenerme activ...

4. Mejoras

1. Implementación de la Distancia de Levenshtein:

- En lugar de buscar coincidencias exactas con el término de búsqueda (usando `.includes()`), ahora se utiliza la *Distancia de Levenshtein*. Esta medida calcula cuántas modificaciones (inserciones, eliminaciones o sustituciones de caracteres) se requieren para transformar una cadena en otra.
- Esto permite encontrar coincidencias aproximadas. Por ejemplo, si el usuario escribe "ingeniero de softwre" en lugar de "ingeniero de software", el código identificará que la diferencia es mínima y aún lo considerará un resultado relevante.

2. Definición de un Límite de Distancia Máxima (`maxDistance`):

- Para controlar la flexibilidad de coincidencias, se ha añadido una variable `maxDistance` que define cuántos cambios son aceptables entre el término de búsqueda y el contenido del sitio para considerarlo como coincidencia.
- Esto permite ajustar la sensibilidad de la búsqueda. Por ejemplo, si `maxDistance` está en 10, el buscador permitirá hasta 10 diferencias en los caracteres antes de descartar el contenido como no coincidente.

3. Búsqueda en Subcadenas del Contenido:

- En lugar de revisar si el término de búsqueda está completamente incluido en el contenido, el código ahora recorre subcadenas del contenido para encontrar aquellas que se parezcan lo suficiente al término de búsqueda, según el valor de `maxDistance`.
- Esta mejora hace que la búsqueda sea más robusta y útil en situaciones donde el término de búsqueda podría estar incompleto o tener errores tipográficos.

4. Resultados Relevantes con Coincidencias Aproximadas:

- Gracias a la distancia de Levenshtein, el buscador ahora muestra resultados más relevantes, incluso si el usuario comete errores menores al escribir el término de búsqueda. Esto mejora la experiencia del usuario, ya que no necesita ser exacto en sus consultas para obtener buenos resultados.

Ejemplo

Search:

rdf

Buscar

Currículum - Diego Martín Fernández

mis proyectos - rdf in zarr proyecto de investigación sobre motores rdf.....

Search:

rdff

Buscar

Currículum - Diego Martín Fernández

mis proyectos - rdf in zarr proyecto de investigación sobre motores rdf.....

He puesto la distancia de Levenshtein a 1 para corregir faltas ortográficas