Universidad Austral de Chile

Conocimiento y Naturaleza

Sistema distribuido basado en la búsqueda a través de índices invertidos

INFO288 Sistemas Distribuidos

Integrantes:

- Gerson Andrade Meza
- Bastián Villanueva Meza
- Diego Troncoso Jara
- José Nicolas Aillapi Gomez
- Alex Garnica Hernández
- Franco Jiménez Muñoz

Carrera: Ingeniería Civil Informática

Introducción

La búsqueda de información en la web es un desafío constante debido a la gran cantidad de datos que se generan y comparten a diario. Para realizar esta tarea se han desarrollado diferentes motores de búsqueda que se dedican a clasificar, indexar y recuperar el contenido de la web.

En este informe se describe el diseño e implementación de un motor de búsqueda distribuido que es capaz de cargar archivos desde diferentes orígenes y entregar resultados de búsqueda mediante índices invertidos, incorporando una memoria caché que nos ayuda a agilizar el proceso. También se detallará la arquitectura propuesta junto a sus características y funcionalidades específicas de cada componente

El sistema está diseñado para ser escalable mediante la arquitectura maestro-esclavo, donde se distribuyen las tareas entre los diferentes nodos, permitiendo una mayor eficiencia en el procesamiento de los datos.

Para lograr este objetivo el sistema contará con los siguientes módulos principales:

- Frontend: permite al usuario interactuar con la aplicación, a través de una interfaz fácil de usar. Aquí el usuario podrá ingresar las palabras clave de su búsqueda y visualizar los resultados.
- Backend: encargado de interactuar con la base de datos y controlar los principales sistemas como la indexación de los documentos y la generación de índices invertidos.
- Base de datos: almacena los documentos mediante su URL junto con su ruta y fecha de scrapeo.
- API: permite la comunicación entre el frontend y el backend.

Preguntas al cliente

Realizar preguntas al cliente es fundamental para poder comprender los requisitos que desea cumplir con el desarrollo de la aplicación. De esta forma podemos obtener información que nos permita diseñar y desarrollar un sistema que se adecúe a sus necesidades. Conocer los detalles como el propósito del sistema, lenguajes de programación, características de la máquina y de usabilidad nos permitirá crear una arquitectura más precisa para el proyecto.

Por lo tanto nos hemos basado en una serie de preguntas y sus supuestos.

1. **¿Qué lenguajes de programación prefiere que se utilicen para el desarrollo del sistema web?**
   R: Deben utilizar Javascript para el frontend y Python para el backend.

2. **En el mercado laboral, ¿estos lenguajes de programación son comunes?**
   R: En la actualidad, todo lo relacionado con desarrollo web está hecho en Javascript, si bien, en temas de backend existen diversos lenguajes que compiten entre ellos pero Python es una muy buena opción para este caso.

3. **¿Cuenta con una máquina (servidor) para hacer el despliegue de la aplicación?**
   R: Si, tengo una pequeña sala de servidores, en la cual tengo una máquina exclusivamente para este proyecto.

4. **¿Qué sistema operativo le acomoda que tenga las máquinas virtuales?**
   R: Me gustaría que las máquinas virtuales contarán con el sistema operativo Ubuntu server, ya que cuenta con una disponibilidad de paquetes y de herramienta enorme, lo que hace que sea fácil encontrar y descargar las herramientas necesarias para nuestra aplicación.

5. **¿Cuál es el propósito de construir esta aplicación?**
   R: Estoy realizando una experimentación para mi trabajo de doctorado.

6. **¿Por qué no desplegar la aplicación en algún servicio en la nube?**
   R: Más que nada por el control y seguridad de los datos que estoy recopilando para mi experimentación.

7. **¿ En su sala de servidores, usted tiene algún tipo de respaldo de datos (backup)?**

R: Si, realizo respaldos una vez al día y solo copio los datos que se han modificado desde el último backup.

8. **¿Qué tipo de base de datos necesita para el sistema web?**
   R: Necesito una base de datos relacional que sea escalable a ser distribuida, para ello sería ideal trabajar con MariaDB ya que para mí es muy cómodo de utilizar porque lo he utilizado anteriormente en otros proyectos.
9. **¿Es necesario utilizar algún framework o biblioteca en específico para el desarrollo del sistema web?**
   R: Pueden utilizar cualquier framework que los desarrolladores estimen conveniente.

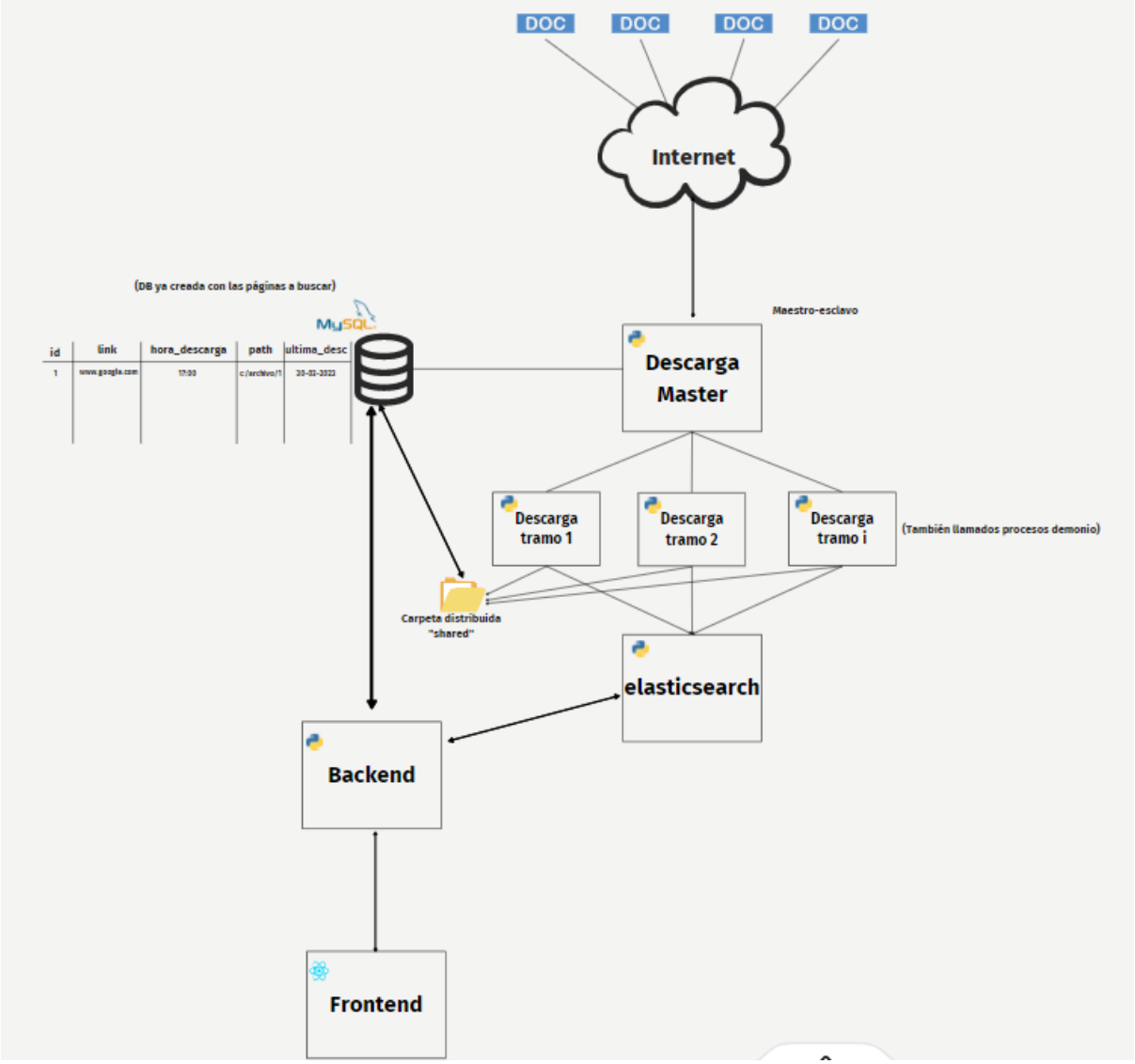
10. **¿Usted utiliza alguna tecnología para el despliegue de la aplicación?**
    R: Me gusta que el despliegue sea con Docker, ya que esta aplicación permite tener escalabilidad, eficiencia y además cuenta con aislamiento para el nuestro software distribuido.

11. **¿Cómo se espera que los usuarios interactúen con el sistema y qué características de usabilidad son importantes para ellos?**
    R: El sistema debe ser fácil de usar, que sea minimalista pero debe ser rápido y seguro**.**

12. **¿Es necesario que el sistema web tenga una API para integrar con otros sistemas o aplicaciones?**
    R: Sí, necesito que la aplicación cuente con “back-end master”, el cual se pueda acoplar con otros sistemas.

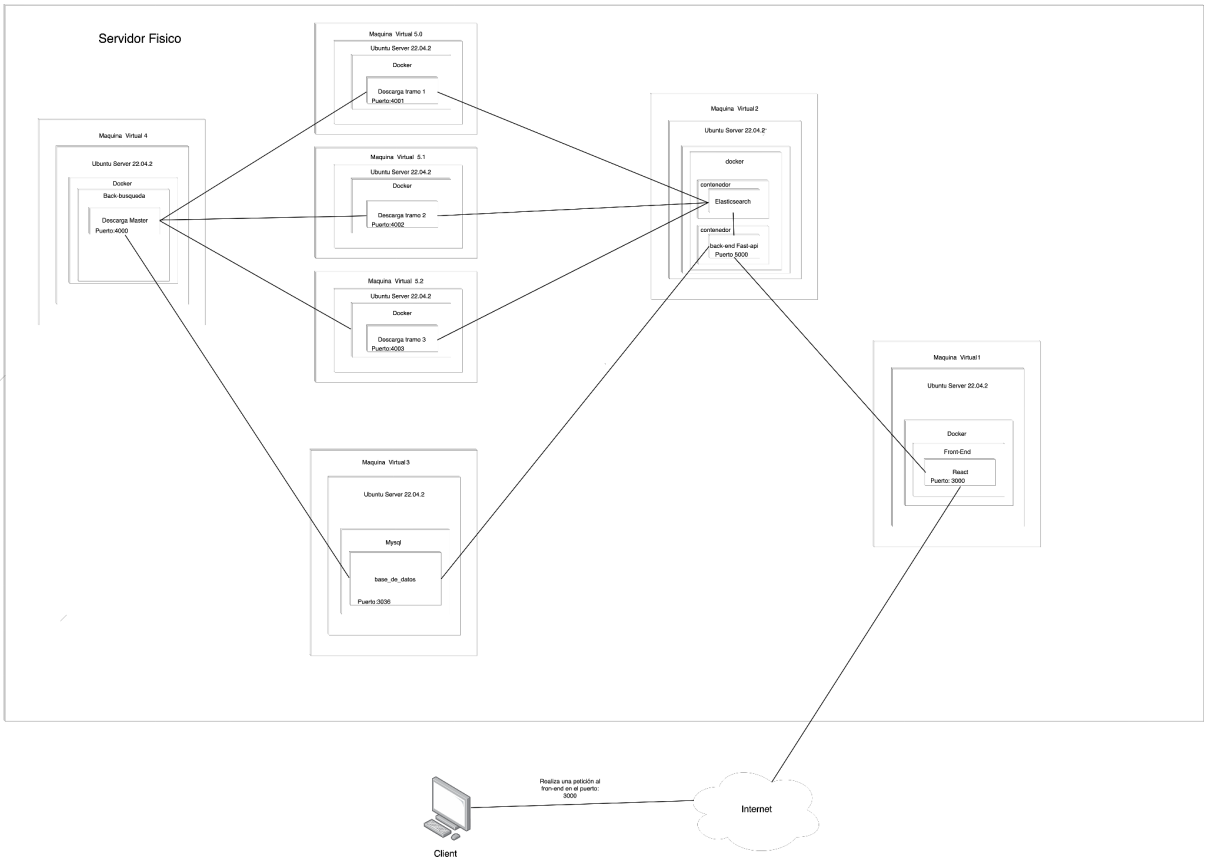
Arquitectura propuesta

Debe presentar el diagrama de arquitectura de su solución, ejemplo: puede utilizar el diagrama UML de componentes de software o cualquier diagrama de arquitectura que muestre el diseño de su solución



ver

Diagrama de despliegue



[Ver]

Descripción de los componentes de la arquitectura propuesta

Maquina virtual 1: La función de esta máquina virtual es alojar todo el front-end del sistema, el cual está destinado en funcionar en un entorno con sistema operativo Ubuntu versión 22.04.2. Todo el front-end será diseñado con React y será alojado en el puerto 3000.

Máquina virtual 2: Esta máquina virtual se encargará de alojar el back-end, el cual enviará los datos al front-end utilizando FastApi. El back-end estará alojado en el puerto 5000. Además, la máquina virtual también tendrá instalado Elasticsearch como sistema de búsqueda y análisis de datos.

Máquina virtual 3: El propósito de esta máquina virtual será contener la base de datos que se explicará más adelante, la cual será Mysql y será alojado en el puerto 3036.

Máquina virtual 4: Esta máquina virtual contendrá la arquitectura maestro-esclavo, el cual será el responsable de coordinar y administrar operaciones de los nodos esclavos, en este caso, sería dividir el trabajo en hacer el scraping de datos. Esta máquina virtual será alojada en el puerto 4000.

Máquina virtual versión 5: Las máquinas virtuales versión 5 alojarán diferentes nodos esclavos que estarán encargados de realizar la extracción de datos en los distintos PDF según la búsqueda deseada. Estos nodos esclavos estarán disponibles desde el puerto 4001 hacia adelante.

modelo fundamental

Nuestro sistema tiene que cumplir con los siguientes principios:

1) **coordinación**: nuestros procesos trabajan en conjunto para proporcionar nuestra aplicación.

2) **Comunicación**: los procesos alojados en distintos nodos se comunican a través de la red utilizando mecanismos de comunicación estándar, como un protocolo de red.

3) **Seguridad**: nuestro sistema debe proteger la privacidad y la integridad de los datos.

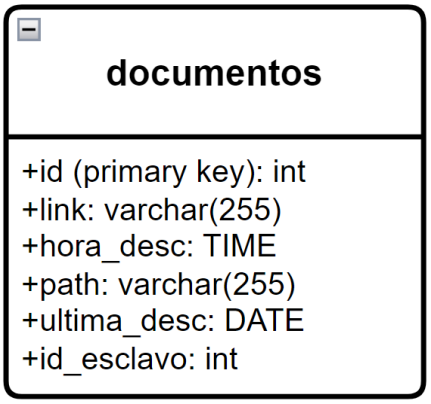
4) **Escalabilidad**: El sistema debe contar con la capacidad de gestionar una gran cantidad de usuarios y procesos. En consecuencia, si se requiere de mayor capacidad de cómputo, nuestro sistema debe ser capaz de proporcionar de manera eficiente y efectiva

5) **Tolerancia a fallo**s: nuestro sistema sistema tiene ser capaz a manejar fallos sin interrupciones significativas en el servicio global. Por ejemplo, si se cae un esclavo de nuestro sistema este tiene que seguir funcionando sin mayor inconveniente.

Base de datos

Para el trabajo se utilizará una base de datos relacional (MariaDB), esto por que cuenta con muchas ventajas, entre ellas:
-Conocimiento previo en el uso de esta base de datos, lo que agilizará la implementación.
-La posibilidad de escalarla, usando su versión distribuida.
-Es un sistema rápido y ligero, perfecto para nuestros objetivos.
Esta base de datos contará con una sola tabla llamada **documentos**, la cual contendrá lo siguiente:

Modelo relacional:



Diccionario de datos:

	Columna	Tipo de dato	Descripción	Null
PK	id	int	Número de identificación del documento (Página web)	false
	link	varchar(255)	Enlace para acceder al documento	false
	hora_desc	TIME	Hora a la que se debe hacer web scraping al documento.	false
	path	varchar(255)	Ruta donde se guarda el documento luego de hacerle web scraping.	true
	ultima_desc	DATE	Fecha de la última vez que se le hizo web scraping al documento.	true
	id_esclavo	int	Identificador del esclavo que hizo el último scraping	true

Se decidió usar una sola tabla como parte de la arquitectura, estas guardarán links, rutas, fechas e información adicional.
Estos documentos a los que se les efectuará web scraping son del tipo HTML, ya que por ahora al ser un prototipo es más que suficiente y nos permitirá tener una idea de cómo funcionará con otros tipos de documentos, a sabiendas de que esta estructura los soportará sin problemas.

El objetivo de esta base de datos es que un “demonio”, consuma estos datos (rutas, links y horas planificadas). ejecutando los web scraping, para luego guardar el path del archivo, la fecha e id del que lo hizo.

Algunos datos pueden ser inicializados como tipo null, como el “path”, “id_esclavo” y “ultima_desc”, esto ya que es necesario hacer un primer web scraping para poder darles un valor.

Especificaciones técnicas de Hardware

Nuestra solución se desplegará en una máquina entregada por el cliente que cuenta con las siguientes especificaciones:

Servidor Dell PowerEdge T40

- Procesador Intel® Xeon® E-2336 2.90GHz
- Total Cores: 16
- Total de threads: 12
- Max Turbo Frequency 4.80 GHz
- Memoria RAM Instalada 16 GB DDR4 2666 MHz
- Disco Duro SATA de 1TB a 7200 RPM
- 1 PCIe x16 3.0 (velocidad x4)
- 1 PCIe x4 3.0
- 1 PCI

para nuestra arquitectura se necesitarán 5 máquinas virtuales en las cuales estarán alojados nuestros componentes

Maquina virtual 1: encargada del frontend

- Sistema operativo: Ubuntu Server 22.04.2
- Cores de procesador: 2
- Memoria ram: 2 GB
- Espacio en disco: 50 GB

Maquina virtual 2: encargada del backend de FastAPI y ElasticSearch.
Podrá necesitar escalar sus recursos, ya que está ejecutando dos procesos.

- Sistema operativo: Ubuntu Server 22.04.2
- Cores de procesador: 4
- Memoria ram: 4 GB
- Espacio en disco: 70 GB

Máquina virtual 3: encargada de la base de datos. Se debe tener en consideración que si el sistema escala en archivos, se puede llegar a necesitar más espacio de almacenamiento en disco.

- Sistema operativo: Ubuntu Server 22.04.2
- Cores de procesador: 2
- Memoria ram: 3 GB
- Espacio en disco: 50 GB

Máquina virtual 4: encargada de descarga-master

- Sistema operativo: Ubuntu Server 22.04.2
- Cores de procesador: 2

- Memoria ram: 2 GB
- Espacio en disco: 50 GB

Máquina virtual 5.x: encargada de la descarga por tramos.

obs: si esta máquina llega a requerir recursos, se levantarán otras máquinas virtuales con el mismo componente de software para la escalabilidad del servicio.

- sistema operativo: Ubuntu Server 22.04.2
- cores de procesador : 2 cores
- memoria ram: 2 gb
- espacio en disco: 50 gb

-> esta máquina puede llegar a necesitar escalabilidad.

Herramientas de software

De manera general las herramientas seleccionadas siguen una línea en lo que es **software libre o gratuito,** son tecnologías más bien **modernas** con una documentación vigente. Además de, como integrantes, haber tenido algunas experiencias de uso con algunas de estas herramientas.

Entorno de trabajo

VSCode
Tipo de licencia: Licencia MIT.
Se utilizará **VSCode** como herramienta de desarrollo y así agilizar el trabajo.

Ventajas
- Amplio soporte para diversos lenguajes de programación.
- Proporciona autocompletado de código y diversas herramientas para simplificar el tiempo de desarrollo ya que es intuitivo y tiene un alto grado de personalización.
- Tiene debugging integrado para depurar código
- Muchas extensiones por parte de la comunidad o empresas de software conocidas, ya sea para simplificar el tiempo empleado en ciertas tareas o para compatibilizar ciertos archivos.
- Es una herramienta multiplataforma, se puede utilizar en Windows, macOS y Linux.

Desventajas
- Puede llegar a consumir muchos recursos para un computador más antiguo.

Software

Frontend

Javascript
Tipo de licencia: Licencia MIT.

Ventajas

- JavaScript es compatible con la mayoría de los navegadores web.
- Es un lenguaje de programación muy flexible y permite una gran cantidad de personalización.
- javaScript permite crear interfaces de usuario interactivas que mejoran la experiencia de usuario de un sitio web.

Desventajas:

- JavaScript puede ser más lento que otros lenguajes de programación, lo que puede afectar el rendimiento de las aplicaciones web más complejas.
- javaScript se ejecuta en diferentes navegadores, pueden haber inconsistencias en el comportamiento y la visualización de las aplicaciones en diferentes plataformas y dispositivos

BackEnd

Python (FastAPI)
Tipo de licencia: Python Software Foundation License, licencia MIT.

Se utilizará FastAPI para las conexiones principales entre elasticsearch, la base de datos y el frontend.

Ventajas

- FastApi es conocido por ser rápido y escalable, lo que lo hace adecuado para aplicaciones que manejen grandes cantidades de datos y peticiones.
- Es fácil de utilizar, ya que cuenta con la sintaxis y semántica de python, lo que hace fácil aprender para los desarrolladores que están familiarizados con python.

Desventajas:

- tiene limitaciones en aplicaciones web complejas que requieren características avanzadas, lo que puede requerir la implementación manual de soluciones personalizadas.

Backend - búsqueda

Python (Elasticsearch)
Tipo de licencia: Licencia Elastic, licencia SSPL.

Se utilizará Elasticsearch para gestionar la información recolectada y la función de búsqueda, debido a que facilita el uso de índice invertido.

Ventajas

- Es una herramienta muy potente para la búsqueda de datos y cuenta con una gran cantidad de funciones integradas.
- Ya aplica el índice invertido para el motor de búsqueda.
- En sí es una librería enfocada en motores de búsqueda, eso garantiza otros tipos de herramientas que podríamos usar (de ser necesarias).

Desventajas

- Elasticsearch es una aplicación que consume muchos recursos, especialmente en términos de memoria y almacenamiento.

Herramientas

Sistema Operativo (máquinas virtuales)
Ubuntu Server 22.04.2
Tipo de licencia: Licencia Pública General (GNU)

Se utilizará Ubuntu Server como sistema operativo en las máquinas virtuales.

Ventajas

- Instalación de paquetes con mayor facilidad
- Libre y gratuito.
- Seguridad y estabilidad.

Desventajas

- Ubuntu server es compatible con muchos sistemas y dispositivos, aún puede tener problemas de compatibilidad de hardware, esencialmente en máquinas más nuevas.

Base de datos
MariaDB
Tipo de licencia: Licencia MariaDB, Licencia GPLv2

La base de datos será MariaDB relacional, de momento contemplamos una entidad, al ser relacional podríamos evaluar y escalar su contenido.

Ventajas

- Es una base de datos relacional de código abierto que ofrece una gran cantidad de funciones integradas y es compatible con MySQL.
- Soporte clustering.

Desventajas

- Menos soporte: Aunque MariaDB cuenta con una gran comunidad de usuarios y desarrolladores, no tiene tanto soporte comercial como otras bases de datos como MySQL o PostgreSQL.

Despliegue
Docker
Tipo de licencia: Licencia Apache 2.0

Docker será utilizado para el despliegue de componentes para conservar un versionamiento fijo y replicable desde cualquier máquina.

Ventajas
- Replicable.
- Portabilidad.
- Aislamiento de las aplicaciones.

Desventajas
- Puede resultar complejo sincronizar muchos componentes y tecnologías distintas (sin conocimiento previo).

Librerías o Frameworks Python

- **FastAPI:** FastAPI es un framework de Python diseñado para facilitar la creación rápida y sencilla de APIs. Se utilizará para las conexiones entre base de datos, frontend y elasticsearch.
    - Documentación FastAPI

- **Elasticsearch:** En lo que respecta a la actividad es el componente principal ya que es el encargado del motor de búsqueda (optimización de consultas). Dado su índice invertido y almacenaje NoSql (documentos).
    - Documentación Elasticsearch

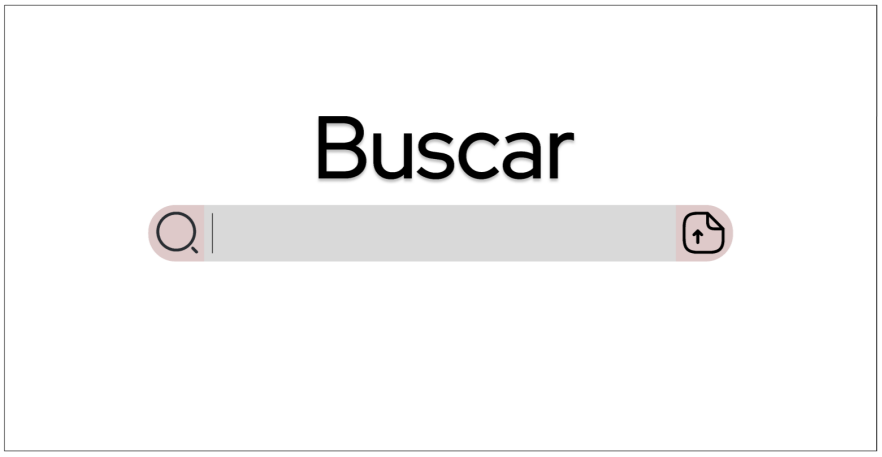
- **React:** Es una biblioteca de JavaScript para front-end, orientado a la creación de interfaces de usuario basado en componentes y se encarga de renderizarlos de manera eficiente.
    - Documentación React

Incorporación Frontend

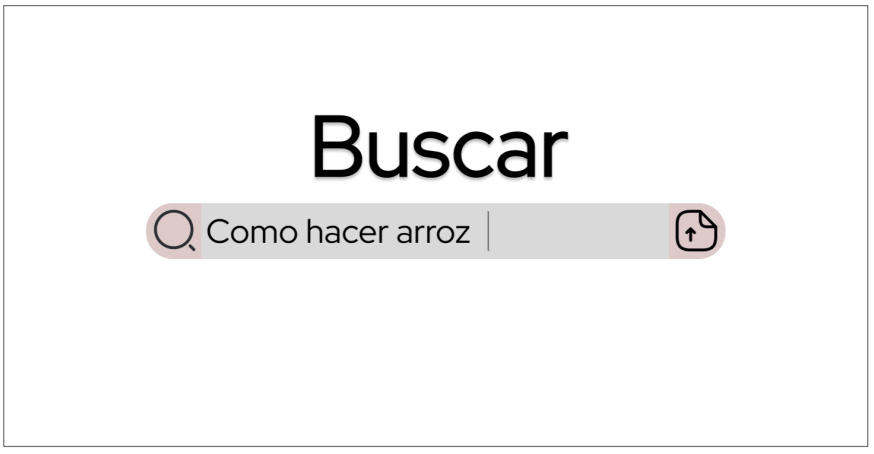
Para la visualización del contenido (interfaz de usuario) utilizaremos la librería React de Javascript. En sí es una librería que facilita el manejo de estados, componentización y facilita la manera de declarar el contenido.

Es una librería bastante completa en caso de que quisiéramos escalar el contenido del apartado visual, ya sea para agregar funcionalidades extras, crear nuevas páginas (enrutado) o utilizar otro tipo de librerías basadas en React.

Un mockup sobre el contenido principal sería el siguiente:
Página principal:



Iniciar búsqueda:



Una vez encontrados los resultados: