

Documentación: Instrucciones para ejecutar el proyecto

Esta guía proporciona los pasos necesarios para configurar el ambiente de desarrollo para el proyecto 1 de bases de datos II.

Índice

- [Documentación: Instrucciones para ejecutar el proyecto](#)
- [Índice](#)
 - [0. Pruebas unitarias](#)
 - [1. Instalar Google Chrome](#)
 - [2. Instalación de Docker](#)
 - [3. Habilitar Kubernetes en Docker](#)
 - [4. Instalación de Lens](#)
 - [5. Instalación de Python](#)
 - [6. Instalación de Visual Studio Code \(VSCode\)](#)
 - [7. Verificar que kubectl esté funcionando](#)
 - [8. Instalación y verificación de Helm](#)
 - [9. Build.sh](#)
 - [10. Install.sh](#)
 - [11. Instalar AWS CLI](#)
 - [12. Verificar el bucket de almacenamiento](#)
 - [13. Instalar la interfaz de la base de datos.](#)
 - [14. Interfaz de RabbitMQ](#)
- [Ejecución del Programa](#)
 - [15. Instalar en la carpeta ..\2025-01-IC4302-PO\PO\docker\scrapper\app.py las siguientes librerías en la terminal:](#)
 - [16. Ejecutar el Scrapper](#)
 - [17. Verificar que el S3-crawler está descargando los archivos del Bucket en MariaDB](#)
 - [18. Iniciar sesión en Kibana - Elasticsearch.](#)
 - [19. Crear los índices en Elasticsearch](#)
 - [20. Ingresar a la UI](#)
 - [21. Inicio de sesión del UI](#)
 - [20. Funciones del UI](#)
 - [21. Monitoreo](#)
 - [23. Conclusiones y recomendaciones.](#)
 - [Conclusiones](#)
 - [Recomendaciones](#)
 - [24. Referencias](#)

0. Pruebas unitarias

Para las pruebas unitarias se probaron los componentes de manera separada, en cada punto encontrara una breve explicacion de como se realizaron asi como un link a un video para verlo de manera grafica y con mayor detalle:

- Selenium Web Scraper: Para realizar esta prueba se ejecutó el scrapper que se encuentra en "P1\docker\scraper\app.py" posteriormente se ejecutó un comando con AWS CLI para subir el archivo de prueba tal cual se subieron los archivos iniciales [] ()
- Hugging Face API: Mediante postman se probó las funcionalidades del endcode y status [] ()
- S3 Crawler Cron Job:
- Ingest:
- API:
- UI: Se probaron cada una de las funciones que tenia que cumplir la app

1. Instalar Google Chrome

Para la ejecucion del Scrapper es necesario descargar e instalar Google Chrome desde https://www.google.com/intl/es_us/chrome/

2. Instalación de Docker

Se utilizó Docker extensivamente en este proyecto ya que nos permite empaquetar y ejecutar aplicaciones en entornos aislados.

1. Descargar Docker Desktop desde <https://www.docker.com/products/docker-desktop>
2. Escoger la instalación para el sistema operativo correspondiente.

3. Habilitar Kubernetes en Docker

1. Abrir Docker Desktop.
2. Ir a **Settings > Kubernetes**.
3. Activar la opción "Enable Kubernetes".
4. Seleccionar Kubeadm en "Cluster Settings"
5. Aplicar los cambios y esperar a que Kubernetes esté activo.

4. Instalación de Lens

Lens nos facilita la gestión de Kubernetes, los pods, cronjobs y demás servicios.

1. Descargar Lens desde <https://k8slens.dev/>.
2. Instalarlo y abrirlo.
3. El cluster de Kubernetes debería conectarse automáticamente.

5. Instalación de Python

1. Descargar e instalar Python desde <https://www.python.org/downloads/>.

6. Instalación de Visual Studio Code (VSCode)

1. Descargar e instalar VSCode desde <https://code.visualstudio.com/>.

7. Verificar que kubectl esté funcionando

kubectl es el command line tool de kubernetes y nos permite ejecutar comandos para la herramienta. Podemos correr el siguiente comando para verificar la instalación de kubectl y que no tengamos pods activos antes de comenzar a bajar las imágenes.

```
kubectl get pods
```

Si no está instalado, referirse a las instrucciones en <https://kubernetes.io/docs/tasks/tools/install-kubectl/>.

8. Instalación y verificación de Helm

1. Instalar Helm siguiendo las instrucciones en <https://helm.sh/docs/intro/install/>.
2. Verificar la instalación ejecutando:

```
helm version
```

9. Build.sh

1. En el archivo **build.sh** localizado en **./P1/docker/**
2. Se debe ejecutar desde el *GitBash* el script **build.sh** seguido de el usuario de *Docker*, en la carpeta **..\2025-01-IC4302\P1\docker**:

```
./build.sh orlkasesina06
```

3. Verificar la instalación de las imágenes correctamente en Docker.
4. Verificar la existencia de los pods en Lens.

10. Install.sh

Luego se debe ejecutar desde el *GitBash* el script **install.sh**, en la carpeta **..\2025-01-IC4302\P1\charts**:

```
./install.sh
```

11. Instalar AWS CLI

Para cargar los documentos iniciales en lo cuales se basa el proyecto es necesario instalar AWS Command Line Interface (AWS CLI) ya que se maneja una mayor cantidad de datos y esto facilita la acción de subir un gran volumen de datos, para ello se debe descargar el instalador desde <https://awscli.amazonaws.com/AWSCLIV2.msi>, luego se ejecuta y se siguen los pasos descritos en el mismo instalador

12. Verificar el bucket de almacenamiento

1. Descarga y ejecutar S3 Browser desde <https://s3browser.com>
2. Conectar al bucket utilizando el nombre 2025-01-ic4302
3. Conectar al bucket utilizando las llaves otorgadas por el profesor: 3.1.
ACCESS_KEY="AKIAQ2VOGXQDUUOEKDNJ" 3.2.
SECRET_KEY="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"

13. Instalar la interfaz de la base de datos.

1. Para poder guardar los datos en maria db con facilidad podemos instalar *DBeaver* en la siguiente página <https://dbeaver.io/>
2. Seguir los pasos de instalación básicos del programa base
3. Para poder conectar a la base, debemos ir a **LENS > Services > Seleccionar el Servicio de maria db > Forward**
4. Colocar en "Local port to forward from:" el puerto 3306

Conexión a la base:

4.1.1. Crear una nueva conexión en el símbolo !. 4.1.2 Seleccionar MariaDB > Siguiente > Dejar los datos como están y buscar la contraseña de conexión en *Lens* 4.1.3. Nos vamos a secrets > Seleccionar *databases-mariadb* > Revelar la contraseña en el símbolo ! > Copiar Contraseña revelada > Pegarla en el apartado de DBeaver *Authentication* > Contraseña

Creación de la base:

4.2.1. Una vez conectada a MariaDB, debemos crear una nueva Base de datos llamada *documents* 4.2.2. Dentro de esa base de datos, presionar click derecho y seleccionar *Editor Sql* > *Script SQL* 4.2.3. Pegar los siguientes scripts :

```
CREATE TABLE objects (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre_objeto VARCHAR(40) NOT NULL UNIQUE,  
    estado ENUM('new', 'processed', 'error') DEFAULT 'new',  
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
)  
ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb3  
COLLATE=utf8mb3_general_ci;  
  
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    password_hash VARCHAR(255) NOT NULL,  
    name VARCHAR(100) NOT NULL  
);  
  
CREATE TABLE prompts (  
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```

    user_id INT NOT NULL,
    prompt_text TEXT NOT NULL,
    likes_count INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE friendships (
    id INT AUTO_INCREMENT PRIMARY KEY,
    follower_id INT NOT NULL,
    following_id INT NOT NULL,
    FOREIGN KEY (follower_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (following_id) REFERENCES users(id) ON DELETE CASCADE,
    UNIQUE KEY unique_friendship (follower_id, following_id)
);

CREATE TABLE likes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    prompt_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (prompt_id) REFERENCES prompts(id) ON DELETE CASCADE,
    UNIQUE KEY unique_like (user_id, prompt_id)
);

-- Índices para mejorar el rendimiento de búsquedas
CREATE INDEX idx_prompts_text ON prompts(prompt_text(255));
CREATE INDEX idx_users_name ON users(name);
CREATE INDEX idx_prompts_created ON prompts(created_at DESC);

```

4.3.1. Y ejecutar el *Script* con el botón ! 4.3.2. Ya estaría la base preparada para poder ejecutar el código

14. Interfaz de RabbitMQ

1. Es necesario para crear la cola de los mensajes y si se quiere tener una vista de los mensajes a RabbitMQ, Primero debes ir a *Lens*
2. Services > databases-rabbitmq > En *Ports* Seleccionar *Forward* en el último > Start
3. Una vez en la web colocar en usuario *user* y en contraseña debemos ir a *Lens* > Secrets > databases-rabbitmq > Revelar la contraseña *rabbitmq-password* copiarla y pegarla en la web.

Ejecución del Programa

15. Instalar en la carpeta `..\2025-01-IC4302-PO\PO\docker\scraper\app.py` las siguientes librerías en la terminal:

```
pip install selenium
```

```
pip install web-driver
```

16. Ejecutar el Scrapper

1. Una vez realizados por completos y sin errores los pasos anteriores, ya se puede ejecutar el archivo *app.py* que se encuentra en *..\P1\docker\scrapper\app.py*, según el editor de texto que se usa, se recomienda usar *Visual Studio Code*,
2. El Scrapper abre una ventana en *Google Chrome* y realiza la búsqueda de un término aleatorio, obtiene la dirección HTML del producto y la envía al bucket.
3. Revisar que en la carpeta *'2023395931/'* se encuentre el HTML del producto seleccionado por el corrimiento más reciente del scrapper.

17. Verificar que el S3-crawler está descargando los archivos del Bucket en MariaDB

1. Opcional: Instalar DBBeaver y realizar la conexión con la base de datos.
2. El S3-crawler se ejecuta de manera automática, lo que se puede hacer para ver su funcionamiento es seguir el rastreo de los archivos: 2.1. Primero llendo a refrescar los archivos del bucket de la carpeta *2023395931/* 2.2. Segundo llendo a RabbitMQ y ver los ultimos mensajes de la cola *ingest* 2.3. Tercero llendo a MariaDB y refrescas los datos de la tabla *objects*
3. Verificar los pasos anteriores.

18. Iniciar sesión en Kibana - Elasticsearch.

1. En Lens, navegar a *Network > Services > ic4302-kb-http* y abrir el puerto.
2. Iniciar sesión en Elasticsearch con el usuario *elastic*
3. Introducir su contraseña, ubicada en *Config > Secrets > ic4302-es-elastic-user*

19. Crear los índices en Elasticsearch

1. Seguir los siguientes pasos *Side menu > Management > Dev Tools*
2. Crear el índice *"products"*

```
PUT /products
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "name": {
        "type": "text"
      },
      "id": {
        "type": "text"
      },
      "price": {
```

```

    "type": "text"
  },
  "description": {
    "type": "text"
  },
  "comments": {
    "type": "text"
  },
  "embeddings": {
    "type": "dense_vector",
    "dims": 768,
    "index": true,
    "similarity": "cosine"
  }
}
}
}

```

3. Crear el índice "nproducts"

```

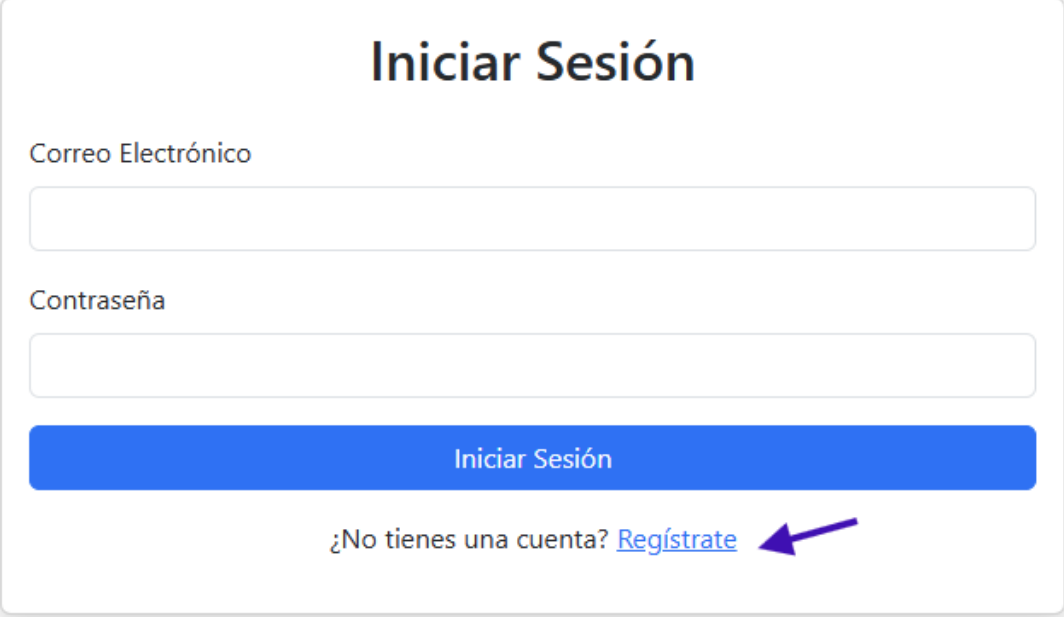
PUT /nproducts
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "name": {
        "type": "text"
      },
      "id": {
        "type": "text"
      },
      "price": {
        "type": "text"
      },
      "description": {
        "type": "text"
      },
      "comments": {
        "type": "text"
      }
    }
  }
}
}

```

3. Verificar la creación navegando a su ubicación en *Side menu > Management > Stack Management > Data > Index Management*

20. Ingresar a la UI

1. Para poder entrar a la UI, debemos ir a **LENS > Services > Seleccionar el Servicio de "prompt-ui" > Ports > Forward**
2. Tocar en el link de registro



The image shows a login form titled "Iniciar Sesión". It contains two input fields: "Correo Electrónico" and "Contraseña". Below these fields is a blue button labeled "Iniciar Sesión". At the bottom of the form, there is a text prompt "¿No tienes una cuenta?" followed by a blue link "Regístrate". A purple arrow points to the "Regístrate" link.


3. Ingresar los datos del usuario y tocar el boton de registro

Registro

Nombre

Correo Electrónico

Contraseña



Registrarse

¿Ya tienes una cuenta? [Inicia Sesión](#)

4. Luego será redirigido a la pantalla de inicio de sesion


21. Inicio de sesion del UI

1. Se debe iniciar sesion con los datos ingresados previamente en el registro y tocar el boton de iniciar sesion

Iniciar Sesión

Correo Electrónico

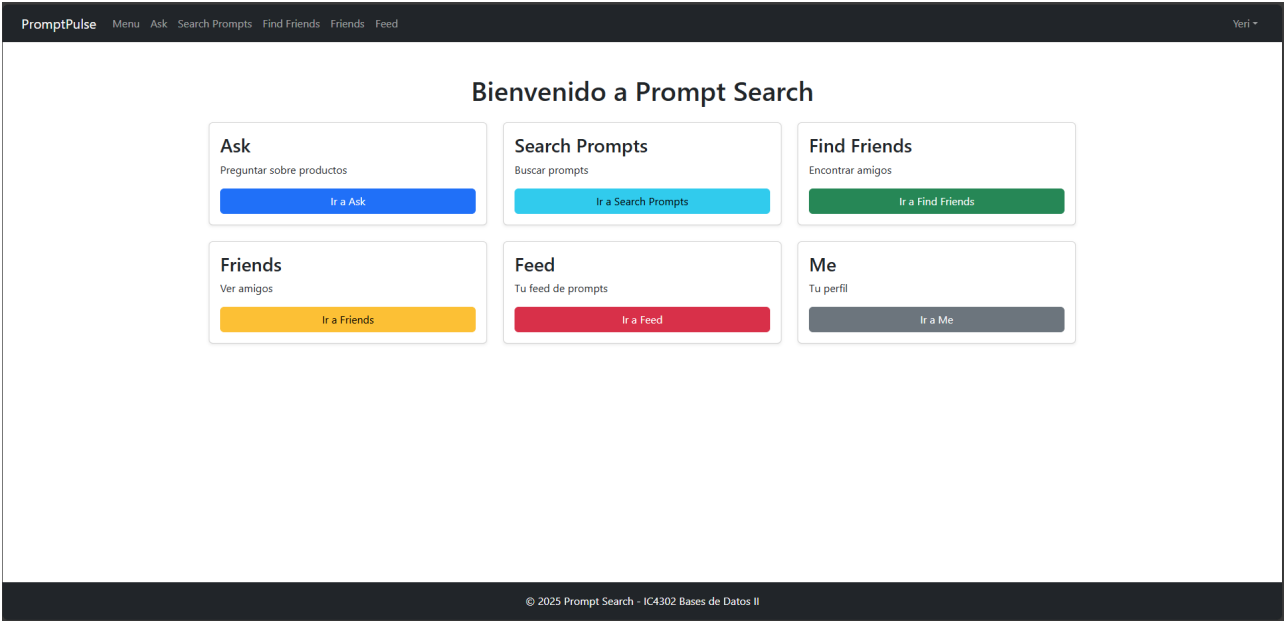
Contraseña



Iniciar Sesión

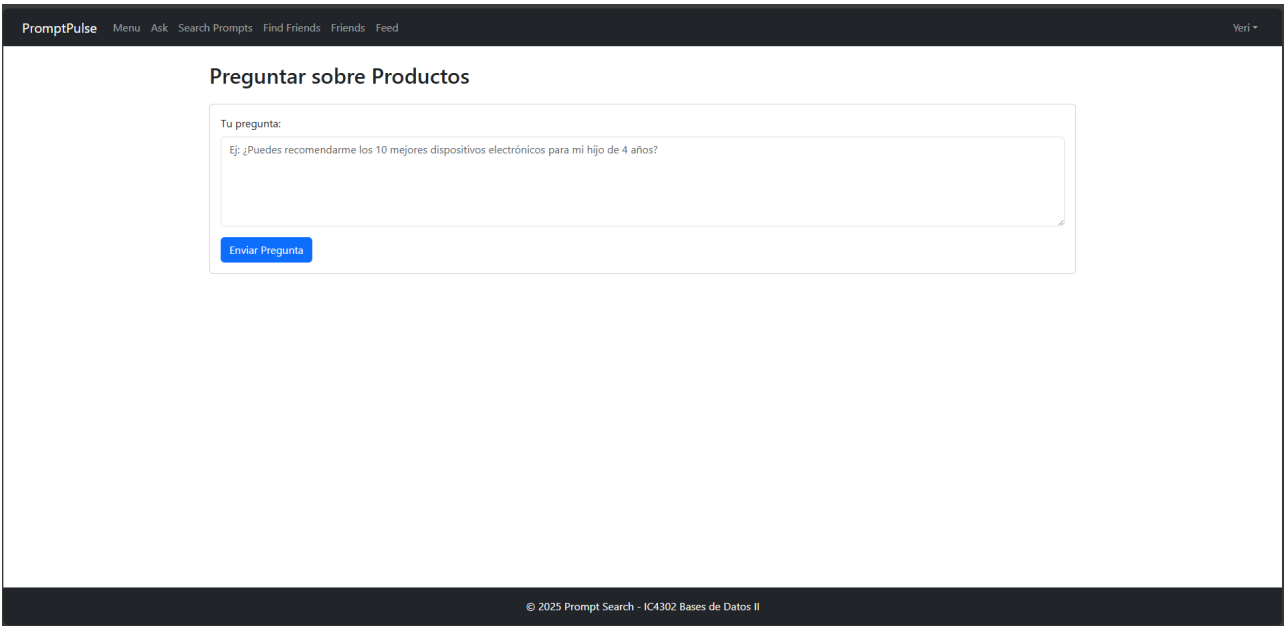
¿No tienes una cuenta? [Regístrate](#)

2. Luego aparecera la ventana principal en la cual se tienen las funciones

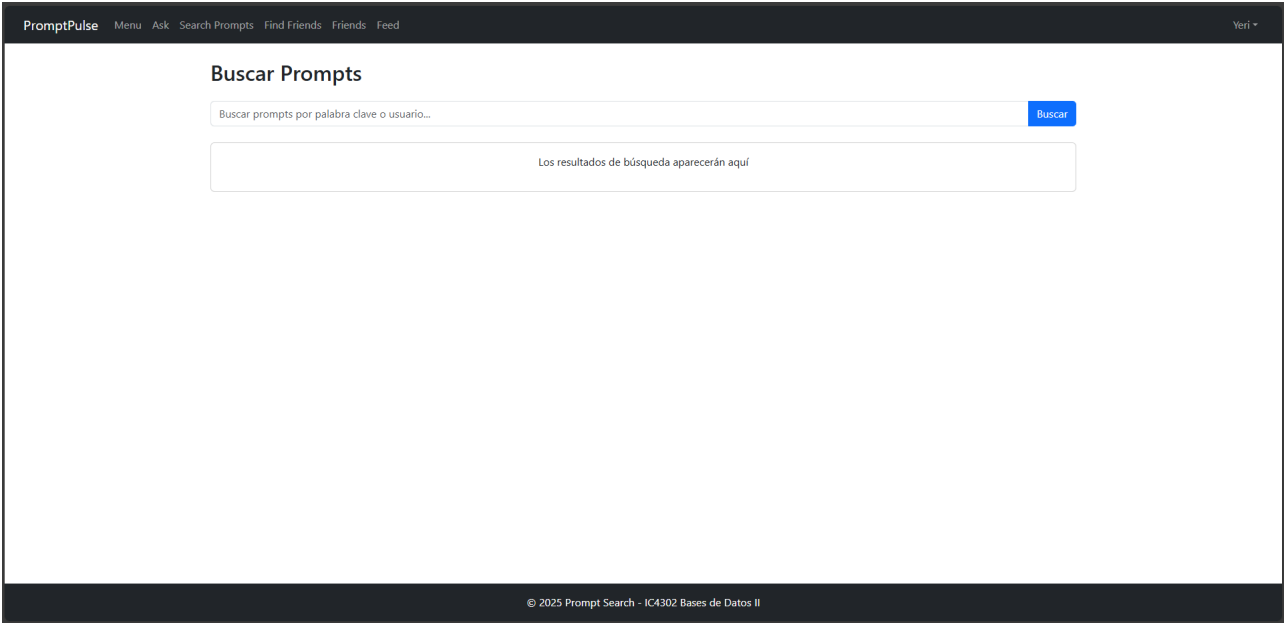


20. Funciones del UI

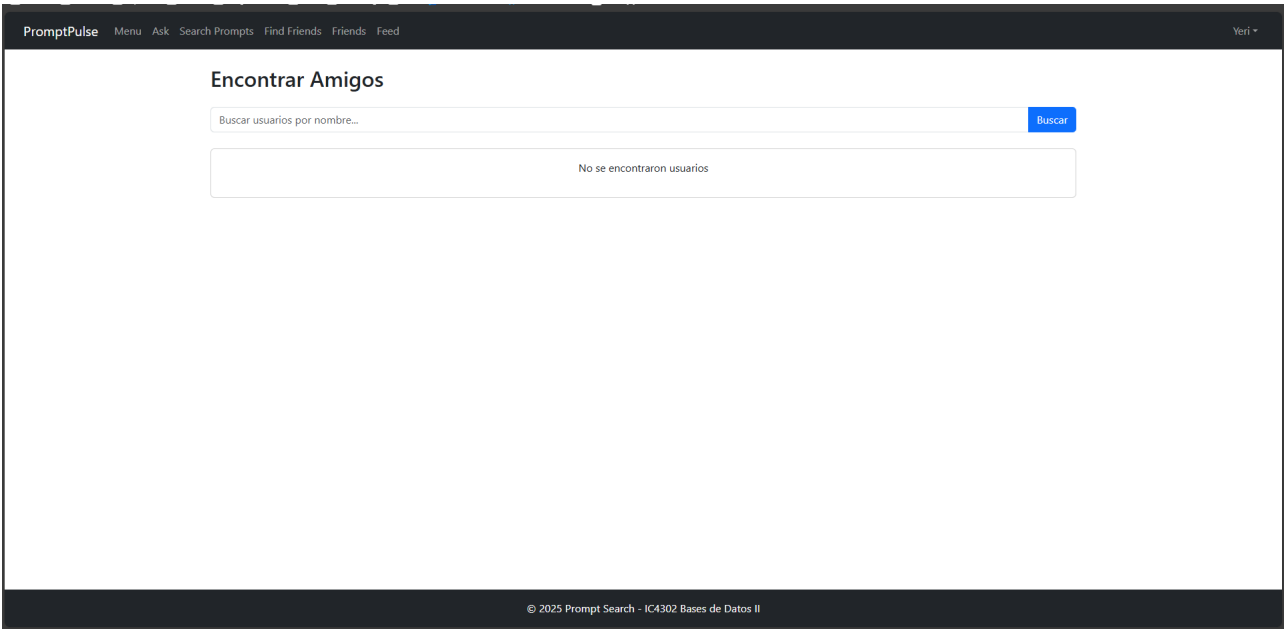
- Ask



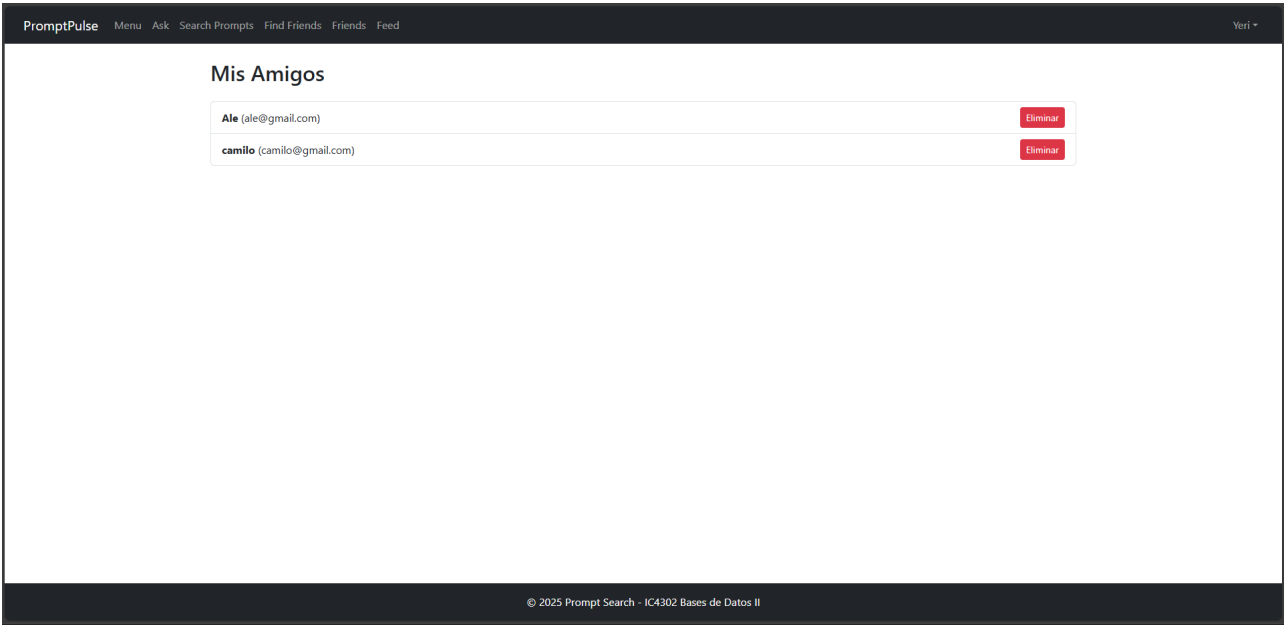
- Search prompts



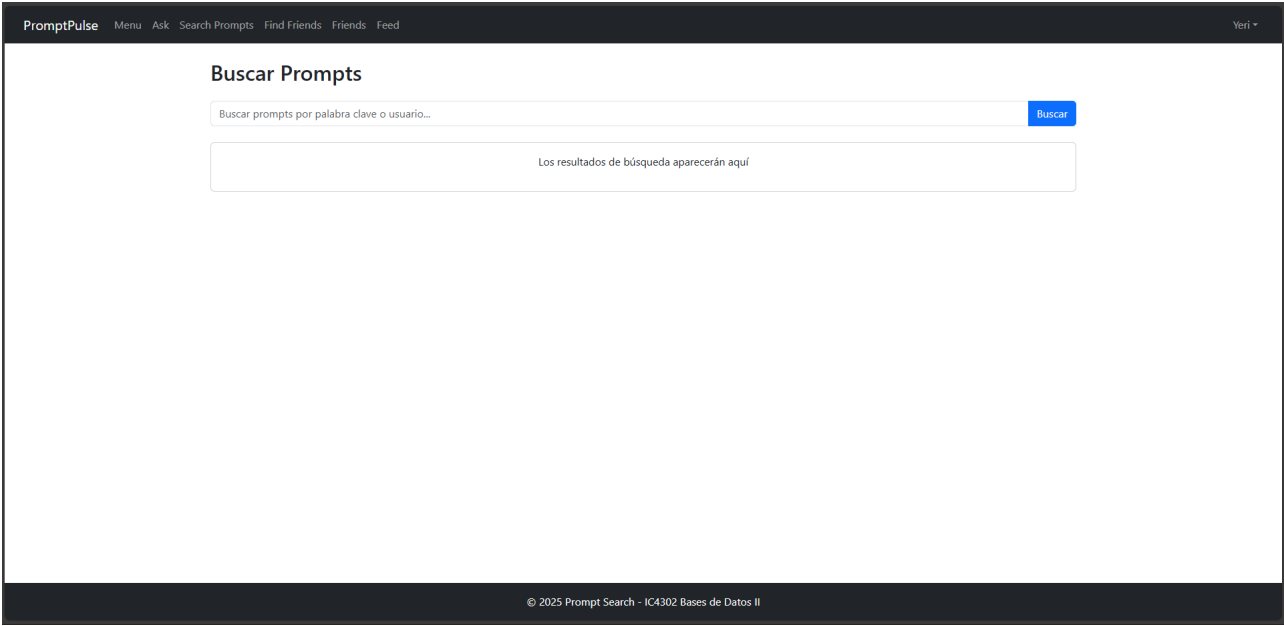
- Find Friends



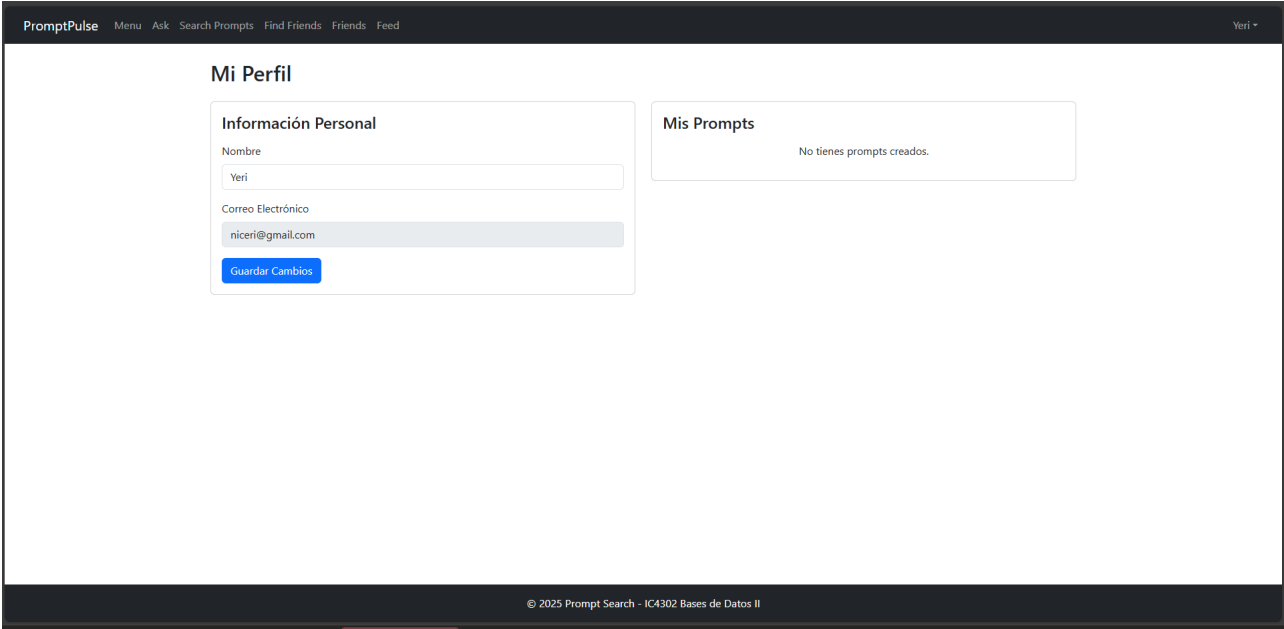
- Friends



- Feed



- Me



21. Monitoreo

El monitoreo se da en los siguientes elementos y de la siguiente manera

- MariaDB y RabbitMQ: Se monitorea por medio de grafana
- Ingest: Se monitorea la suma de tiempo, la cantidad de documentos con error y la cantidad de objetos procesados totales.

23. Conclusiones y recomendaciones.

Conclusiones

1. El sistema cumple con la funcionalidad de la app
2. El uso de herramientas como dbeaver, lens y postman es de suma utilidad para ver de manera grafica y verificar el funcionamiento individual de diversos procesos

3. Para la implementacion de un codigo es necesario saber y debatir las fortalezas y debilidades de cada miembro del grupo para tener un mejor manejo
4. El uso de github hace mas facil el traslado del codigo a los demas miembros del grupo
5. La integracion de Selenium, RabbitMQ y Elasticsearch permite un mejor manejo de datos
6. El uso de funciones para el seguimiento del estado de los documentos hace que sea mas facil su seguimiento
7. Las pruebas unitarias funcionan muy bien para poder verificar y recordar el funcionamiento de los componentes del proyecto
8. Trabajar la documentacion conforme se trabaja es mejor para darle seguimiento a lo que se va haciendo
9. El proyecto logro un buen manejo de los vectores al integrar varias tecnologias aplicables al ambito laboral
10. Un buen manejo de puertos hace que sea mas fluida la interaccion y prueba de codigos

Recomendaciones

1. Realizar un mejor manejo del git y git hub para optimizar el manejo de versiones
2. Hacer una mejor reparticion de trabajos para que cada integrante tenga una carga similar
3. Tener mejor manejo del tiempo y aprovechar espacios por mas pronto que sea
4. No hacer cambios que no sean funcionales para no tener codigo erroneo
5. Trabajar con metas periodicas para no postergar ninguna parte del desarrollo
6. Documentar de mejor manera el codigo para poder tener un buen entendimiento del codigo con los demas compañeros
7. Evitar el trabajo bajo presion debido al empezar da forma tardia
8. Realizar reuniones mas periodicas con el fin de avanzar de manera mas fluida
9. Buscar areas de mejora de manera individual y grupal para mejorar el desempeño no solo del grupo sino de cada integrante
10. Fomentar el aprovechamiento de las herramientas y usar buenas practicas

24. Referencias

1. Amazon Web Services, "Amazon S3 - Cloud Object Storage," [En línea]. Disponible en: <https://aws.amazon.com/s3>
2. Amazon Web Services, "Boto3 1.38.8 documentation," [En línea]. Disponible en: <https://boto3.amazonaws.com/v1/documentation/api/latest>
3. Axios Authors, "Axios - Promise based HTTP client," [En línea]. Disponible en: <https://axios-http.com>
4. bcrypt Developers, "bcrypt," [En línea]. Disponible en: <https://pypi.org/project/bcrypt>
5. Elastic, "Elastic — The Search AI Company," [En línea]. Disponible en: <https://www.elastic.co>
6. Elastic, "Kibana: Explore, Visualize, Discover Data," [En línea]. Disponible en: <https://www.elastic.co/kibana>
7. Flask-Cors Team, "Flask-Cors," [En línea]. Disponible en: <https://pypi.org/project/Flask-Cors>
8. Grafana Labs, "Grafana: The open and composable observability platform," [En línea]. Disponible en: <https://grafana.com>

9. Kenneth Reitz, "Requests: HTTP for Humans," [En línea]. Disponible en: <https://docs.python-requests.org/en/latest>
10. Leonard Richardson, "beautifulsoup4," [En línea]. Disponible en: <https://pypi.org/project/beautifulsoup4>
11. MariaDB Foundation, "MariaDB Foundation - MariaDB.org," [En línea]. Disponible en: <https://mariadb.org>
12. Memcached, "memcached - a distributed memory object caching system," [En línea]. Disponible en: <https://memcached.org>
13. Meta Platforms, Inc., "React – A JavaScript library for building user interfaces," [En línea]. Disponible en: <https://reactjs.org>
14. Pallets Projects, "Flask Documentation (3.1.x)," [En línea]. Disponible en: <https://flask.palletsprojects.com>
15. pika Developers, "Pika 1.3.2 documentation," [En línea]. Disponible en: <https://pika.readthedocs.io>
16. Prometheus Authors, "Prometheus - Monitoring system & time series database," [En línea]. Disponible en: <https://prometheus.io>
17. Prometheus Authors, "Prometheus Python Client," [En línea]. Disponible en: https://github.com/prometheus/client_python
18. Prometheus Community, "Elasticsearch stats exporter for Prometheus," [En línea]. Disponible en: https://github.com/prometheus-community/elasticsearch_exporter
19. PyMySQL Developers, "PyMySQL Documentation," [En línea]. Disponible en: <https://pymysql.readthedocs.io>
20. pymemcache Developers, "pymemcache," [En línea]. Disponible en: <https://pypi.org/project/pymemcache>
21. RabbitMQ, "RabbitMQ: One broker to queue them all," [En línea]. Disponible en: <https://www.rabbitmq.com>
22. Reactstrap Authors, "reactstrap - React Bootstrap 4 components," [En línea]. Disponible en: <https://reactstrap.github.io>
23. sentence-transformers Developers, "sentence-transformers," [En línea]. Disponible en: <https://pypi.org/project/sentence-transformers>
24. urllib3 Developers, "urllib3," [En línea]. Disponible en: <https://pypi.org/project/urllib3>