

Documentación: Instrucciones para ejecutar el proyecto

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

0. Instalar Google Chrome

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

1. 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.

2. 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.

3. 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.

4. Instalación de Python

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

5. Instalación de Visual Studio Code (VSCode)

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

6. 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/>.

7. 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
```

8. Build.sh

1. En el archivo `build.sh` localizado en `./PO/docker/`, cambiar `$1` por el nombre de usuario del dockerhub.
2. Ejecutar `build.sh`
3. Verificar la instalación de las imágenes correctamente en Docker.
4. Verificar la existencia de los pods en Lens.

9. Install.sh

1. Cambiamos `docker_registry: nereo08` a nuestro usuario de Docker Hub en `values.yaml` dentro de `PO/charts/application`
2. Ejecutar el archivo `install.sh` en `./PO/charts/`
3. Verificar la instalación de Elasticsearch, Kibana, MariaDB y RabbitMQ. 2.1 En Docker, verificar la instalación de `bitnami/rabbitmq`, `bitnami/mariadb`, docker.elastic.co/elasticsearch/elasticsearch y docker.elastic.co/kibana/kibana.

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

10. 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="AKIAQ2VOGXQDVCA5KS4V" 3.2.
SECRET_KEY="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"


11. 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 *control* 4.2.2. Dentro de esa base de datos, presionar click derecho y seleccionar *Editor Sql* > *Script SQL* 4.2.3. Pegar el siguiente script :

```
CREATE TABLE objects (
  id int(11) NOT NULL AUTO_INCREMENT,
  path_documento varchar(255) NOT NULL,
  estado enum('new','updated','downloaded','processed') NOT NULL,
  md5_hash char(32) NOT NULL,
  fecha_registro timestamp NULL DEFAULT current_timestamp(),
  fecha_actualizacion timestamp NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  PRIMARY KEY (id),
  UNIQUE KEY path_documento (path_documento)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_general_ci;
```

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

12. Crear un Cronjob para la ejecución automatizada del s3-spider

1. Nos vamos a *Lens* > Buscamos el apartado de *Workloads* > Seleccionamos *Cron Jobs* > Seleccionamos



, abre la pestaña para crear un cronjob y guardarlo

2. En esa pestaña se debe pegar el siguiente *Script*:

```

apiVersion: batch/v1
kind: CronJob
metadata:
  name: s3-spider
spec:
  schedule: "*/2 * * * *" # Modificar si se desea que corra con otra frecuencia
                             en minutos, en este ejemplo son 2 minutos
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: s3-spider
              image: orlkasesina06/s3-spider:latest # Cambia <orlkasesina06> por su
usuario de Docker
              env:
                - name: BUCKET
                  value: "2025-01-ic4302"
                - name: KEY
                  value: "2023395931" # Ruta dentro del bucket donde están los
archivos
                - name: ACCESS_KEY
                  value: AKIAQ2VOGXQDVCA5KS4V
                - name: SECRET_KEY
                  value: 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
                - name: RABBITMQ
                  value: "databases-rabbitmq" # IP o URL de RabbitMQ
                - name: RABBITMQ_QUEUE
                  value: "MessageDocuments" # Nombre de la cola en RabbitMQ
                - name: MARIADB
                  value: "databases-mariadb" # IP o URL del servidor MariaDB
                - name: MARIADB_USER
                  value: "root" # Usuario de MariaDB
                - name: MARIADB_PASS
                  value: "WgEORHJJyI" # Cambiar por la contraseña que se encuentra en
Lens en secrets databases-mariadb
                - name: MARIADB_DB
                  value: "control" # Nombre de la base de datos
                - name: MARIADB_TABLE
                  value: "objects" # Nombre de la tabla donde guarda los archivos
                - name: RABBITMQ_USER
                  value: "user"
                - name: RABBITMQ_PASS
                  value: "tOHVluskizEC8LYP" # Cambiar por la contraseña que se
encuentra en Lens en secrets databases-rabbitmq
              restartPolicy: OnFailure # Puedes cambiarlo a Never si no quieres
reintentos

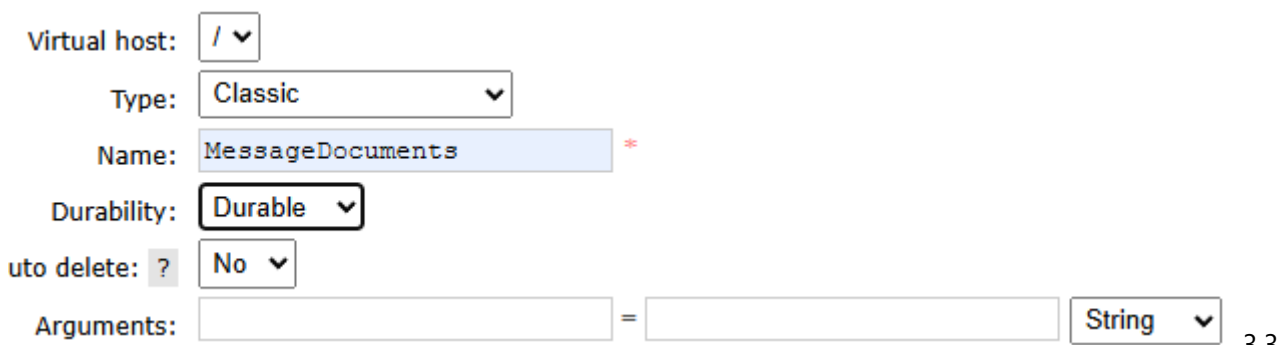
```

13. 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.

Creación de la cola para mensajes

3.1. Debemos seleccionar *Queues and Streams* > *Add a new queue* 3.2. Colocar las siguientes configuraciones:



Virtual host: /

Type: Classic

Name: MessageDocuments *

Durability: Durable

Auto delete: ? No

Arguments: = String

3.3.

Seleccionar *Add queue*

Quedaría listo RabbitMQ para ejecución

14. Instalar S3 Browser para una mejor visualización de los documentos que se descargan de la tienda.

1. ir a la dirección <https://s3browser.com/>, descargar la aplicación e instalar las recomendaciones.
2. Conectar al Bucket con las siguientes credenciales: 2.1. Bucket: 2025-01-ic4302 2.2. Access Key: AKIAQ2VOGXQDVCA5KS4V 2.3. Secret Key: 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
3. Ir a la Bucket 2025-01-ic4302 > Carpeta 2023395931/, Ahí se encontrarán los documentos que descargue el *Scraper*

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 boto3
```

```
pip install selenium
```

```
pip install web-driver
```

- Luego de nuevo se debe ejecutar desde el *GitBash* el script **build.sh** seguido de el usuario de *Docker*, en la carpeta `..\2025-01-IC4302-PO\PO\docker`:

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

16. Ejecutar el Scraper

1. Una vez realizados por completos y sin errores los pasos anteriores, ya se puede ejecutar el archivo *app.py* que se encuentra en `..\PO\docker\scraper\app.py`, según el editor de texto que se usa, se recomienda usar *Visual Studio Code*,
2. El Scraper 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 scraper.

17. Crear el CronJob para el S3-Spider

18. Verificar que el S3-Spider está descargando los archivos del Bucket en MariaDB

- Opcional: Instalar DBBeaver y realizar la conexión con la base de datos.
2. El Spider se ejecuta de manera automática gracias al *Cron Job* creado en el paso 16, 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 MariaDB y refrescas los datos de la tabla *objects* 2.3. Tercero, llendo a RabbitMQ y ver los ultimos mensajes de la cola *MessageDocuments*
 3. Verificar los pasos anteriores.

19. Crear la cola ProcessDocuments.

1. Navegar al apartado Queues and Streams.
2. Iniciar el proceso de creacion de uan cola nueva.
3. Cambiar el tipo a *Classuic*.
4. Colocar el nombre *ProcessDocuments*.
5. Crear la cola.

20. Iniciar sesión en Kibana - Elasticsearch.

1. En Lens, navegar a *Network > Services > ic4302-es-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*

21. Crear los índices en Elasticsearch

1. Crear los siguientes índices *Side menu > Management > Dev Tools*
2. Crear el índice *"processed_documents"*

```
PUT /processed_documents
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "title": {
        "type": "text"
      },
      "product_name": {
        "type": "text"
      },
      "price": {
        "type": "keyword"
      },
      "estado": {
        "type": "text"
      },
      "specifications": {
        "type": "nested",
        "properties": {
          "name": { "type": "text" },
          "value": { "type": "text" }
        }
      },
      "images": {
        "type": "keyword"
      }
    }
  }
}
```

3. Crear el índice *"documents"*

```
PUT /documents
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "title": {
        "type": "text"
      },
    },
  }
}
```

```

    "content": {
      "type": "text"
    }
  }
}

```

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

22. Crear el Deployment del Downloader.

1. Cambiar el nombre de usuario de la imagen.
2. Cambiar RABBITMQ_PASS y MARIADB_PASS por sus contraseñas.
3. Verificar que otras variables de entorno sean correctas. ej: RABBITMQ_QUEUE.

```

apiVersion: batch/v1
kind: CronJob
metadata:
  name: s3-spider
  namespace: default
  uid: 0db867a9-13bf-4217-a6ee-5e459bbb14a
  resourceVersion: '29712'
  generation: 2
  creationTimestamp: '2025-03-21T19:33:15Z'
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: >
      {"apiVersion":"batch/v1","kind":"CronJob","metadata":{"annotations":
      {},"creationTimestamp":"2025-03-21T04:50:56Z","generation":2,"name":"s3-
      spider","namespace":"default","selfLink":"/apis/batch/v1/namespaces/default/cronjo
      bs/s3-spider","uid":"24f47293-e44c-4f10-8e3c-9fccbc41be57"},"spec":
      {"concurrencyPolicy":"Allow","failedJobsHistoryLimit":1,"jobTemplate":{"metadata":
      {"creationTimestamp":null},"spec":{"template":{"metadata":
      {"creationTimestamp":null},"spec":{"containers":[{"env":
      [{"name":"BUCKET","value":"2025-01-ic4302"},{"name":"KEY","value":"2023395931"},
      {"name":"ACCESS_KEY","value":"AKIAQ2VOGXQDVCA5KS4V"},
      {"name":"SECRET_KEY","value":"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"},
      {"name":"RABBITMQ","value":"databases-rabbitmq"},
      {"name":"RABBITMQ_QUEUE","value":"MessageDocuments"},
      {"name":"MARIADB","value":"databases-mariadb"},
      {"name":"MARIADB_USER","value":"root"},
      {"name":"MARIADB_PASS","value":"yyxhqe1f9F"},
      {"name":"MARIADB_DB","value":"control"},
      {"name":"MARIADB_TABLE","value":"objects"},
      {"name":"RABBITMQ_USER","value":"user"},
      {"name":"RABBITMQ_PASS","value":"rTndqcRxMYPHZxYn"}],"image":"technowaffles/s3-
      spider:latest","imagePullPolicy":"Always","name":"s3-spider","resources":
      {},"terminationMessagePath":"/dev/termination-
      log","terminationMessagePolicy":"File"}],"dnsPolicy":"ClusterFirst","restartPolicy
      ":"OnFailure","schedulerName":"default-scheduler","securityContext":
      {},"terminationGracePeriodSeconds":30}}}},{"schedule":"*/2

```



```

    * * * *", "successfulJobsHistoryLimit": 3, "suspend": true}}
  selfLink: /apis/batch/v1/namespaces/default/cronjobs/s3-spider
status:
  active:
    - kind: Job
      namespace: default
      name: s3-spider-29043268
      uid: d6bc2a0d-32bf-4257-a140-add30c2e448d
      apiVersion: batch/v1
      resourceVersion: '29711'
  lastScheduleTime: '2025-03-21T22:28:00Z'
  lastSuccessfulTime: '2025-03-21T22:26:07Z'
spec:
  schedule: '*/2 * * * *'
  concurrencyPolicy: Allow
  suspend: false
  jobTemplate:
    metadata:
      creationTimestamp: null
    spec:
      template:
        metadata:
          creationTimestamp: null
        spec:
          containers:
            - name: s3-spider
              image: technowaffles/s3-spider:latest
              env:
                - name: BUCKET
                  value: 2025-01-ic4302
                - name: KEY
                  value: '2023395931'
                - name: ACCESS_KEY
                  value: AKIAQ2VOGXQDVCA5KS4V
                - name: SECRET_KEY
                  value: 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
                - name: RABBITMQ
                  value: databases-rabbitmq
                - name: RABBITMQ_QUEUE
                  value: MessageDocuments
                - name: MARIADB
                  value: databases-mariadb
                - name: MARIADB_USER
                  value: root
                - name: MARIADB_PASS
                  value: yyxhq1f9F
                - name: MARIADB_DB
                  value: control
                - name: MARIADB_TABLE
                  value: objects
                - name: RABBITMQ_USER
                  value: user
                - name: RABBITMQ_PASS
                  value: rTndqcRxMYPHZxYn

```

```

resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
imagePullPolicy: Always
restartPolicy: OnFailure
terminationGracePeriodSeconds: 30
dnsPolicy: ClusterFirst
securityContext: {}
schedulerName: default-scheduler
successfulJobsHistoryLimit: 3
failedJobsHistoryLimit: 1

```

4. Verificar en los logs que esté consumiendo los mensajes de la cola de MessageDocuments RabbitMQ

23. Crear el Deployment del Procesor.

1. Cambiar el nombre de usuario de la imagen.
2. Cambiar RABBITMQ_PASS, MARIADB_PASS, ELASTICSEARCH_INDEX, ELASTICSEARCH_INDEX_DST por sus contraseñas y datos.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: procesor
  namespace: default
  uid: a0a716ec-1119-4f5d-aed5-de1d5fee5e24
  resourceVersion: '15151'
  generation: 2
  creationTimestamp: '2025-03-21T19:54:39Z'
  labels:
    app: procesor
    app.kubernetes.io/managed-by: Helm
    k8slens-edit-resource-version: v1
  annotations:
    deployment.kubernetes.io/revision: '2'
    kubectl.kubernetes.io/last-applied-configuration: >
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":
      {},"labels":{"app":"procesor","app.kubernetes.io/managed-
      by":"Helm"},"name":"procesor","namespace":"default"},"spec":
      {"progressDeadlineSeconds":600,"replicas":1,"revisionHistoryLimit":10,"selector":
      {"matchLabels":{"app":"procesor"},"strategy":{"rollingUpdate":
      {"maxSurge":"25%","maxUnavailable":"25%"},"type":"RollingUpdate"},"template":
      {"metadata":{"labels":{"app":"procesor"},"spec":{"containers":[{"env":
      [{"name":"BUCKET","value":"2025-01-ic4302"},{"name":"KEY","value":"2023395931"},
      {"name":"ACCESS_KEY","value":"AKIAQ2VOGXQDVCA5KS4V"},
      {"name":"SECRET_KEY","value":"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"},
      {"name":"RABBITMQ","value":"databases-rabbitmq"},
      {"name":"RABBITMQ_QUEUE","value":"ProcessedDocuments"},
      {"name":"RABBITMQ_USER","value":"user"},
      {"name":"RABBITMQ_PASS","value":"rTndqcRxMYPHZxYn"},
      {"name":"MARIADB","value":"databases-mariadb"},

```

```

{"name": "MARIADB_USER", "value": "root"},
{"name": "MARIADB_PASS", "value": "yyxhqe1f9F"},
{"name": "MARIADB_DB", "value": "control"},
{"name": "MARIADB_TABLE", "value": "objects"},
{"name": "ELASTICSEARCH", "value": "http://ic4302-es-http:9200"},
{"name": "ELASTICSEARCH_USER", "value": "elastic"},
{"name": "ELASTICSEARCH_PASS", "value": "0iUz3wJce164973J4DZ2j4PF"},
{"name": "ELASTICSEARCH_INDEX", "value": "documents"},
{"name": "ELASTICSEARCH_INDEX_DST", "value": "processed_documents"}], "image": "technowaffles/processor", "imagePullPolicy": "Always", "name": "procesor", "resources": {}, "terminationMessagePath": "/dev/termination-log", "terminationMessagePolicy": "File"}], "dnsPolicy": "ClusterFirst", "restartPolicy": "Always", "schedulerName": "default-scheduler", "securityContext": {}, "terminationGracePeriodSeconds": 30}}}}
  selfLink: /apis/apps/v1/namespaces/default/deployments/procesor
status:
  observedGeneration: 2
  replicas: 1
  updatedReplicas: 1
  readyReplicas: 1
  availableReplicas: 1
  conditions:
    - type: Available
      status: 'True'
      lastUpdateTime: '2025-03-21T19:54:43Z'
      lastTransitionTime: '2025-03-21T19:54:43Z'
      reason: MinimumReplicasAvailable
      message: Deployment has minimum availability.
    - type: Progressing
      status: 'True'
      lastUpdateTime: '2025-03-21T20:30:20Z'
      lastTransitionTime: '2025-03-21T19:54:39Z'
      reason: NewReplicaSetAvailable
      message: ReplicaSet "procesor-6449c9bfcf" has successfully progressed.
spec:
  replicas: 1
  selector:
    matchLabels:
      app: procesor
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: procesor
    spec:
      containers:
        - name: procesor
          image: technowaffles/processor
          env:
            - name: BUCKET
              value: 2025-01-ic4302
            - name: KEY
              value: '2023395931'
            - name: ACCESS_KEY

```

```

    value: AKIAQ2VOGXQDVCA5KS4V
  - name: SECRET_KEY
    value: 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
  - name: RABBITMQ
    value: databases-rabbitmq
  - name: RABBITMQ_QUEUE
    value: ProcessedDocuments
  - name: RABBITMQ_USER
    value: user
  - name: RABBITMQ_PASS
    value: rTNdqcRxMYPHZxYn
  - name: MARIADB
    value: databases-mariadb
  - name: MARIADB_USER
    value: root
  - name: MARIADB_PASS
    value: yyxhqe1f9F
  - name: MARIADB_DB
    value: control
  - name: MARIADB_TABLE
    value: objects
  - name: ELASTICSEARCH
    value: http://ic4302-es-http:9200
  - name: ELASTICSEARCH_USER
    value: elastic
  - name: ELASTICSEARCH_PASS
    value: ILN7b13V2kAzjcZL07009rZ4
  - name: ELASTICSEARCH_INDEX
    value: documents
  - name: ELASTICSEARCH_INDEX_DST
    value: processed_documents
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
imagePullPolicy: Always
restartPolicy: Always
terminationGracePeriodSeconds: 30
dnsPolicy: ClusterFirst
securityContext: {}
schedulerName: default-scheduler
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 25%
    maxSurge: 25%
revisionHistoryLimit: 10
progressDeadlineSeconds: 600

```

24. Link de acceso a pruebas unitarias

- [Pruebas Realizadas](#)

25. Conclusiones y recomendaciones.

Conclusiones

1. El sistema cumple con el principio básico de lo solicitado
2. El uso de herramientas como dbeaver y lens es de suma utilidad para ver de manera gráfica diversos procesos
3. Antes de la implementación de cada código es eficiente debatir su funcionamiento entre el grupo
4. El uso de docker estandariza y hace más fácil el traslado del código a otros miembros del grupo
5. La integración de Selenium, RabbitMQ y Elasticsearch permite un mejor manejo de datos
6. El uso de MariaDB hace más fácil el seguimiento del estado de los documentos
7. Las pruebas unitarias son una gran ayuda para verificar el estado de cada parte por separado
8. La gestión de RabbitMQ es de suma importancia para un buen flujo y procesamiento de los documentos
9. La automatización garantiza un manejo fluido de los documentos
10. Son herramientas muy potentes y muy aplicables a la realidad actual con la cantidad de datos que se pueden manejar

Recomendaciones

1. Realizar un mejor manejo del git y git hub para optimizar el manejo de versiones
2. Tener una mejor comunicación y organización con el equipo para no hacer procesos repetidos y no malgastar el tiempo
3. Empezar con más antelación para no tener que hacer procesos atropellados
4. Leer más detenidamente la documentación antes de empezar el proyecto para saber previamente los alcances de las herramientas
5. Tener nombres significativos en los documentos desde el principio para no tener que modificar después la estructura del código
6. Tener un mejor manejo de los recursos que utiliza kubernetes para que todos los miembros puedan ejecutarlos con fluidez
7. Trabajar con mayor cooperatividad para no tener sesgos en el progreso del trabajo individual
8. Realizar revisiones periódicas de código y funcionalidad para facilitar el avance del trabajo
9. Corroborar el funcionamiento de los contenedores de docker antes de pasarlos a los demás compañeros
10. Tener mayor cuidado con el manejo de passwords para no tener confusiones con las de los demás compañeros

26. Referencias

1. Python Software Foundation, "os — Miscellaneous operating system interfaces," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/os.html>
2. Python Software Foundation, "json — JSON encoder and decoder," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/json.html>
3. Python Software Foundation, "logging — Logging facility for Python," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3/howto/logging.html>
4. Amazon Web Services, Inc., "Boto3 documentation," *AWS SDK for Python (Boto3)*, 2025. [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

5. L. Richardson, "Beautiful Soup documentation," *Beautiful Soup 4*, 2025. [Online]. Available: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
6. Pika Developers, "Pika documentation," *Pika – RabbitMQ Client Library for Python*, 2025. [Online]. Available: <https://pika.readthedocs.io/en/stable/>
7. PyMySQL Developers, "PyMySQL documentation," *PyMySQL – MySQL Client for Python*, 2025. [Online]. Available: <https://pymysql.readthedocs.io/en/latest/>
8. Elasticsearch BV, "Elasticsearch documentation," *Elasticsearch Reference*, 2025. [Online]. Available: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>
9. Elastic, "Kibana documentation," *Kibana Guide*, 2025. [Online]. Available: <https://www.elastic.co/guide/en/kibana/current/index.html>
10. MariaDB Foundation, "MariaDB documentation," *MariaDB Knowledge Base*, 2025. [Online]. Available: <https://mariadb.com/kb/en/documentation/>
11. RabbitMQ Team, "RabbitMQ documentation," *RabbitMQ Messaging Broker*, 2025. [Online]. Available: <https://www.rabbitmq.com/documentation.html>
12. K. Reitz, "Requests: HTTP for humans," *Requests Documentation*, 2025. [Online]. Available: <https://docs.python-requests.org/en/latest/>
13. Python Software Foundation, "tempfile — Generate temporary files and directories," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/tempfile.html>
14. Python Software Foundation, "hashlib — Secure hashes and message digests," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/hashlib.html>
15. Python Software Foundation, "sys — System-specific parameters and functions," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/sys.html>
16. Python Software Foundation, "time — Time access and conversions," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/time.html>
17. Python Software Foundation, "random — Generate pseudo-random numbers," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/random.html>
18. Selenium Project, "WebDriver documentation," *Selenium Documentation*, 2025. [Online]. Available: <https://www.selenium.dev/documentation/webdriver/>
19. Selenium Project, "Selenium documentation," *Selenium Documentation*, 2025. [Online]. Available: <https://www.selenium.dev/documentation/>
20. Python Software Foundation, "datetime — Basic date and time types," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/datetime.html>
21. Python Software Foundation, "unittest — Unit testing framework," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/unittest.html>
22. Python Software Foundation, "unittest.mock — Mocking and testing," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/unittest.mock.html>
23. Python Software Foundation, "io — Core tools for working with streams," *Python 3.10 Documentation*, 2025. [Online]. Available: <https://docs.python.org/es/3.10/library/io.html>