



Desarrollo de una aplicación distribuida para la extracción, almacenamiento y procesamiento del historial de artículos wiki basados en Mediawiki

Marvin E. Bernal P.
C.I: 18.154.154

Francisco J. Delgado M.
C.I: 19.608.720

Tutor:
Prof. Eugenio Scalise

Agenda

- ◎ Contexto y problema
- ◎ Objetivos
- ◎ Justificación
- ◎ Arquitectura de la solución
- ◎ Metodología
- ◎ Componentes de la arquitectura
- ◎ Demostración
- ◎ Conclusiones y recomendaciones

Contexto y Problema

- Wikipedia es una de las herramientas más importantes para la búsqueda de información
- Miles de artículos son actualizados diariamente
- Estos cambios son almacenados en forma de historiales, también llamados revisiones

Crimea: Revision history

[?](#) Help

[View logs for this page](#)

Search for revisions

From year (and earlier): 2018

From month (and earlier): all

Tag filter:

Show

For any version listed below, click on its date to view it.

For more help, see [Help:Page history](#) and [Help:Edit summary](#).

External tools: [Revision history statistics](#) • [Revision history search](#) • [Edits by user](#) • [Number of watchers](#) • [Page view statistics](#) • [Fix dead links](#)

(cur) = difference from current version, (prev) = difference from preceding version,

m = minor edit, **→** = section edit, **←** = automatic edit summary

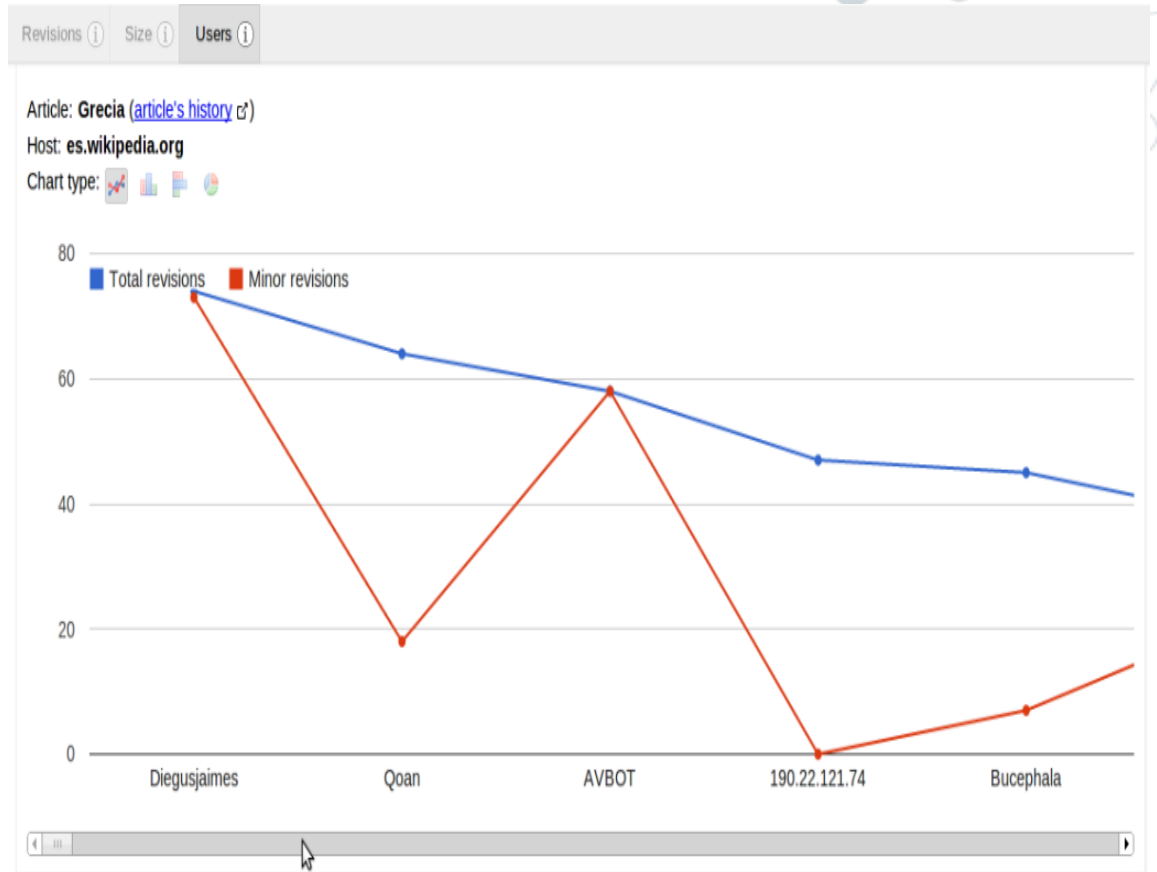
(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- [\(cur | prev\)](#) [14:43, 14 May 2018](#) [DeprecatedFixerBot](#) ([talk](#) | [contribs](#)) **m** . . (105,766 bytes) **(+12)** . . *(Removed deprecated parameter(s) from [Template:Columns-list](#) using [DeprecatedFixerBot](#). Questions? See [Template:Div col#Usage of "cols" parameter or msg TSD!](#) (please mention that this is task #2!))* ([undo](#))
- [\(cur | prev\)](#) [18:48, 10 May 2018](#) [Arjayay](#) ([talk](#) | [contribs](#)) **m** . . (105,754 bytes) **(+4)** . . *(Reverted edits by [104.153.241.74](#) ([talk](#)) to last version by [Dawnseeker2000](#))* ([undo](#)) ([Tag: Rollback](#))

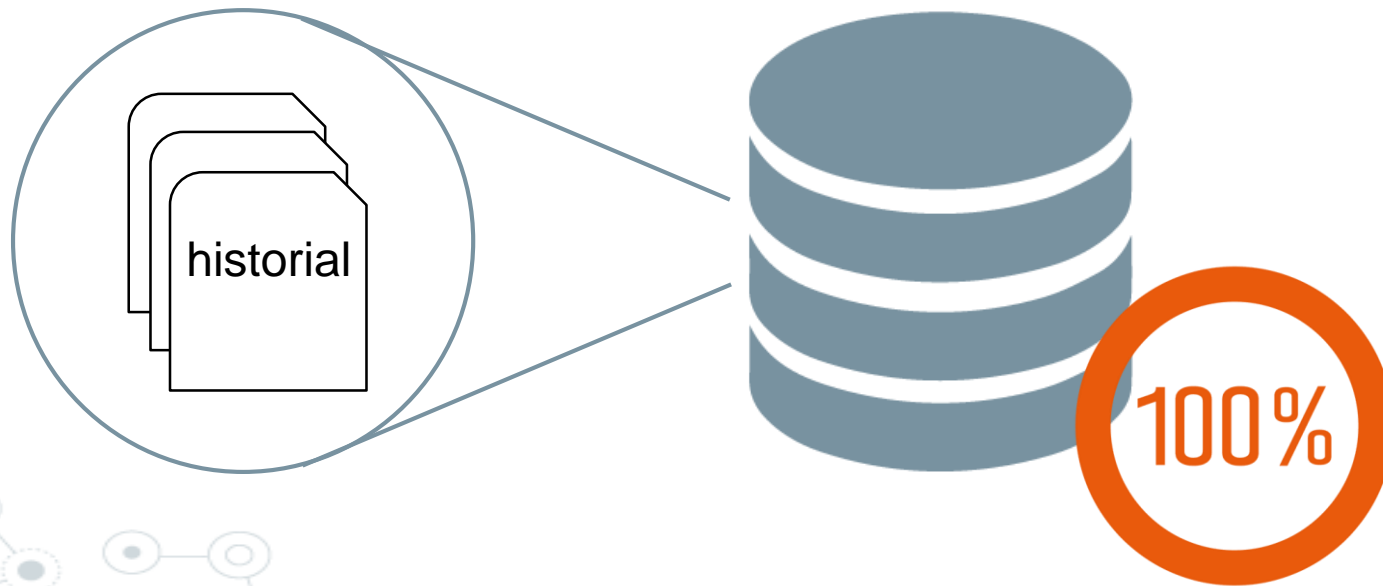
Contexto y Problema

- El estudio de los historiales permite visualizar métricas como: tendencia histórica o la ubicación geográfica de los autores
- Wiki-Metrics-UCV es una solución centralizada para el estudio de estos historiales



Contexto y Problema

- ◎ Un sistema centralizado está atado a limitaciones en el almacenamiento





Contexto y Problema

◎ Una base de datos centralizada es vulnerable a fallas

◎ Recuperarse de fallos suele ser complicado

◎ Cuellos de botella



Objetivos

Desarrollar una aplicación distribuida para la extracción, almacenamiento y procesamiento del historial de los artículos wiki basados en MediaWiki a través de técnicas de web scraping y el uso de MediaWiki API

Objetivos

- ⦿ Desarrollar un módulo para la recolección y consulta de los datos sobre el servicio API de MediaWiki
- ⦿ Diseñar un modelo de datos para el almacenamiento de los historiales de los artículos wiki
- ⦿ Implementar y configurar la topología de sharding de MongoDB para el sistema distribuido

Objetivos

- ⦿ Configurar los nodos de MongoDB dentro del sistema distribuido para la implementación de réplicas
- ⦿ Adaptar los algoritmos de revisita sobre los artículos wiki basados en la solución brindada por Wiki-Metrics-UCV



Objetivos

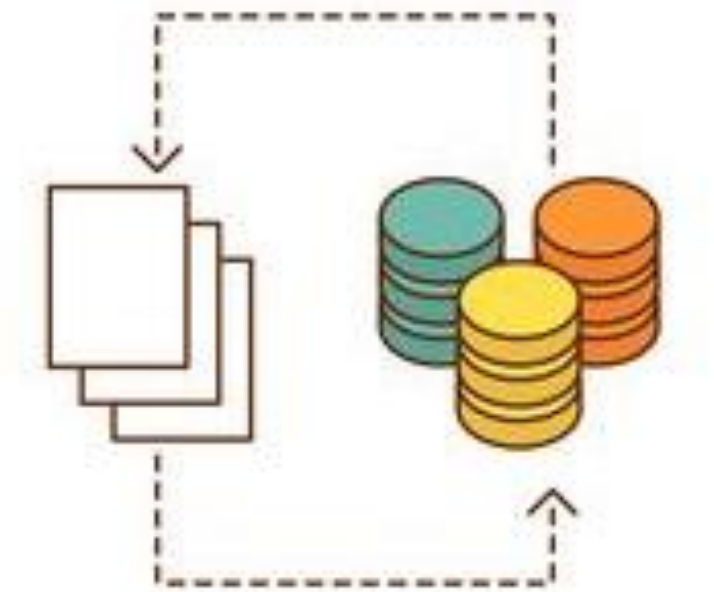
- ① Diseñar y configurar los métodos de asignación de tareas para el balance de carga
- ① Desarrollar los algoritmos de procesamiento para la visualización de estadísticas

Objetivos

- ◎ Implementar un conjunto de pruebas sobre los módulos de extracción de datos, almacenamiento, replicación y procesamiento de los mismos
- ◎ Desarrollar un API que permita a aplicaciones de terceros consultar los historiales de los artículos wiki

Justificación

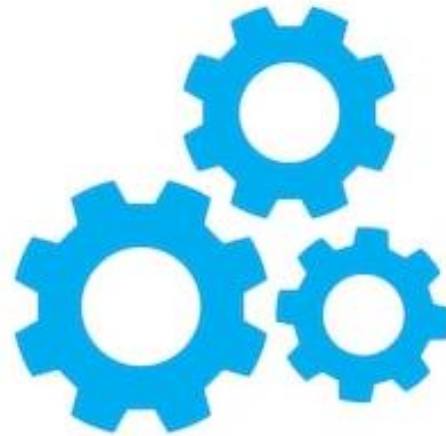
- ⦿ Escalamiento horizontal
- ⦿ Compartir carga de trabajo
- ⦿ Mayor capacidad de almacenamiento



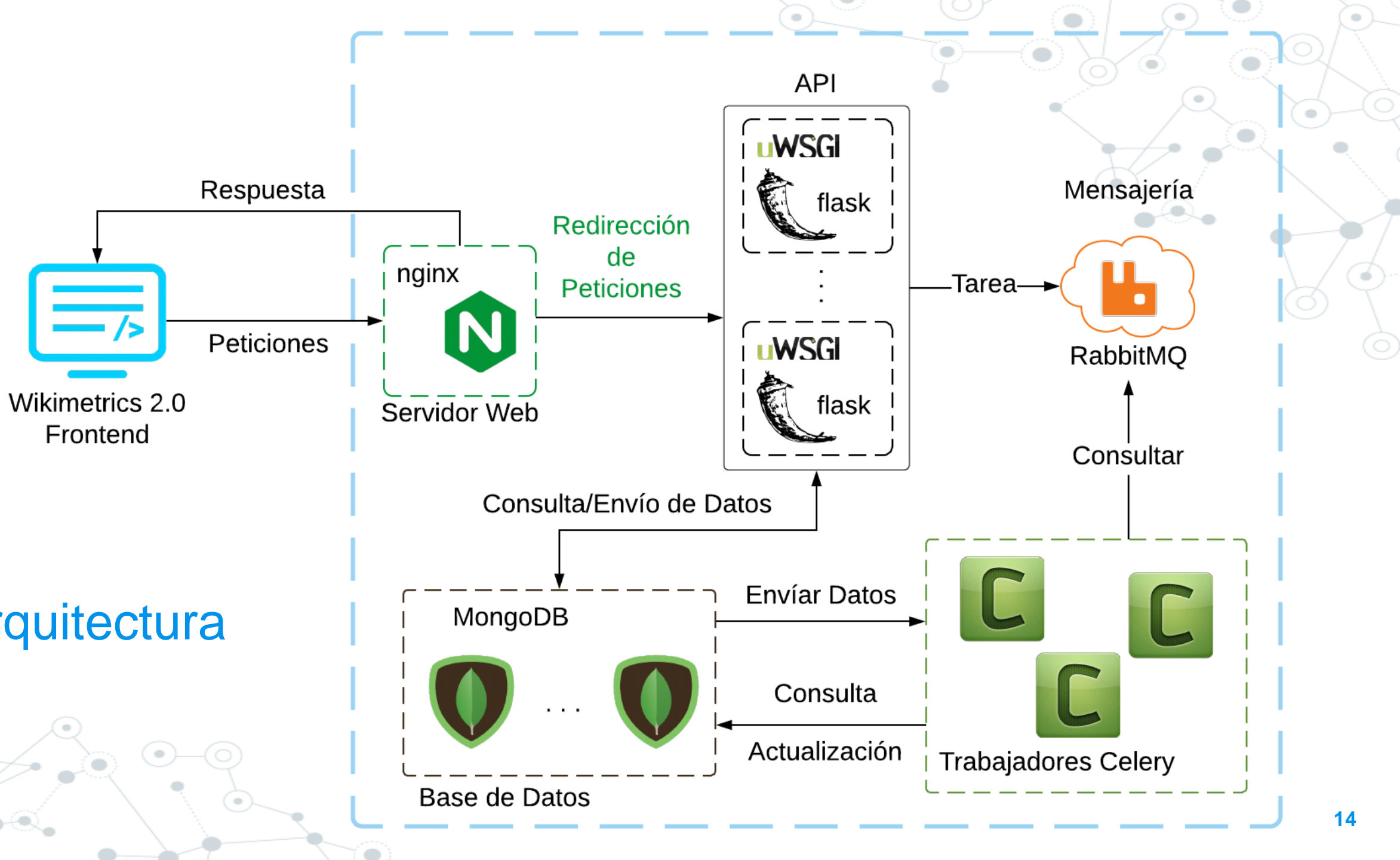
Arquitectura

⦿ Arquitectura de la solución

⦿ Tecnologías

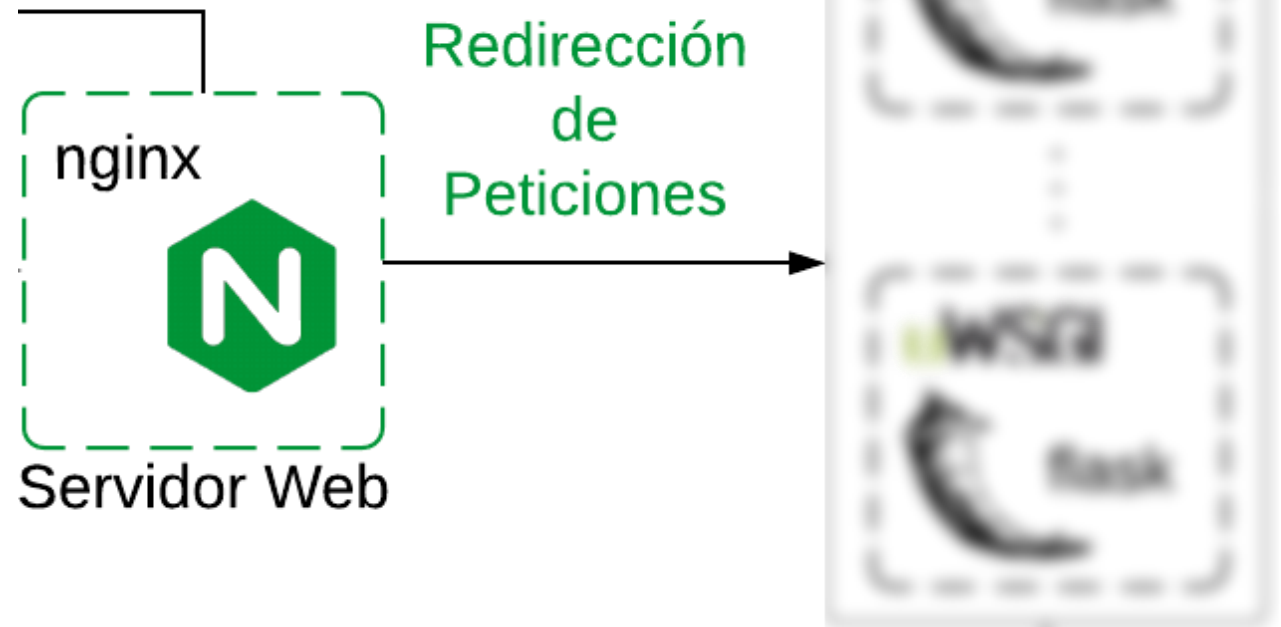


Arquitectura



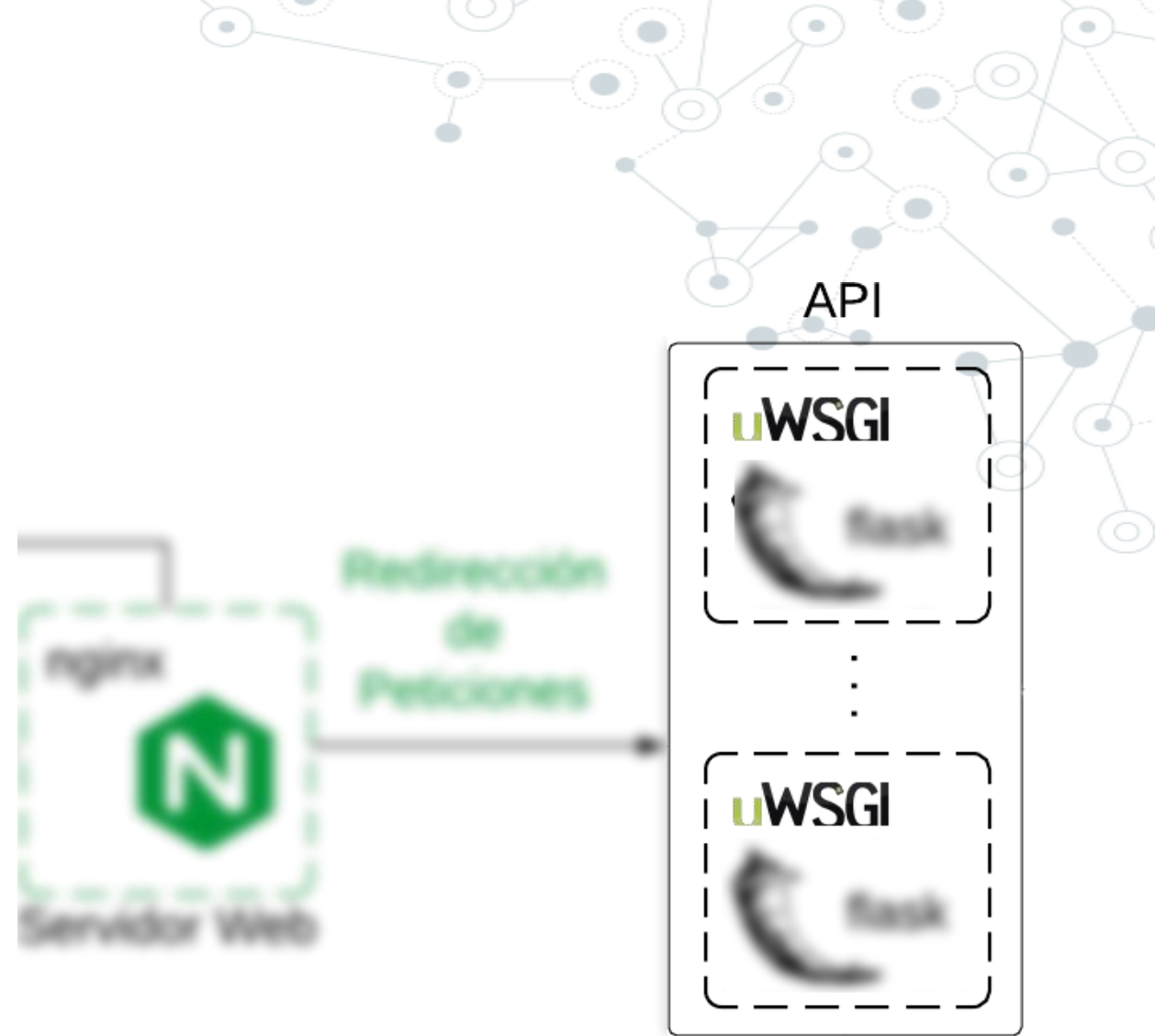
Tecnologías - Nginx

- ⦿ Servidor HTTP que también funciona como un proxy reverso
- ⦿ Balance de carga



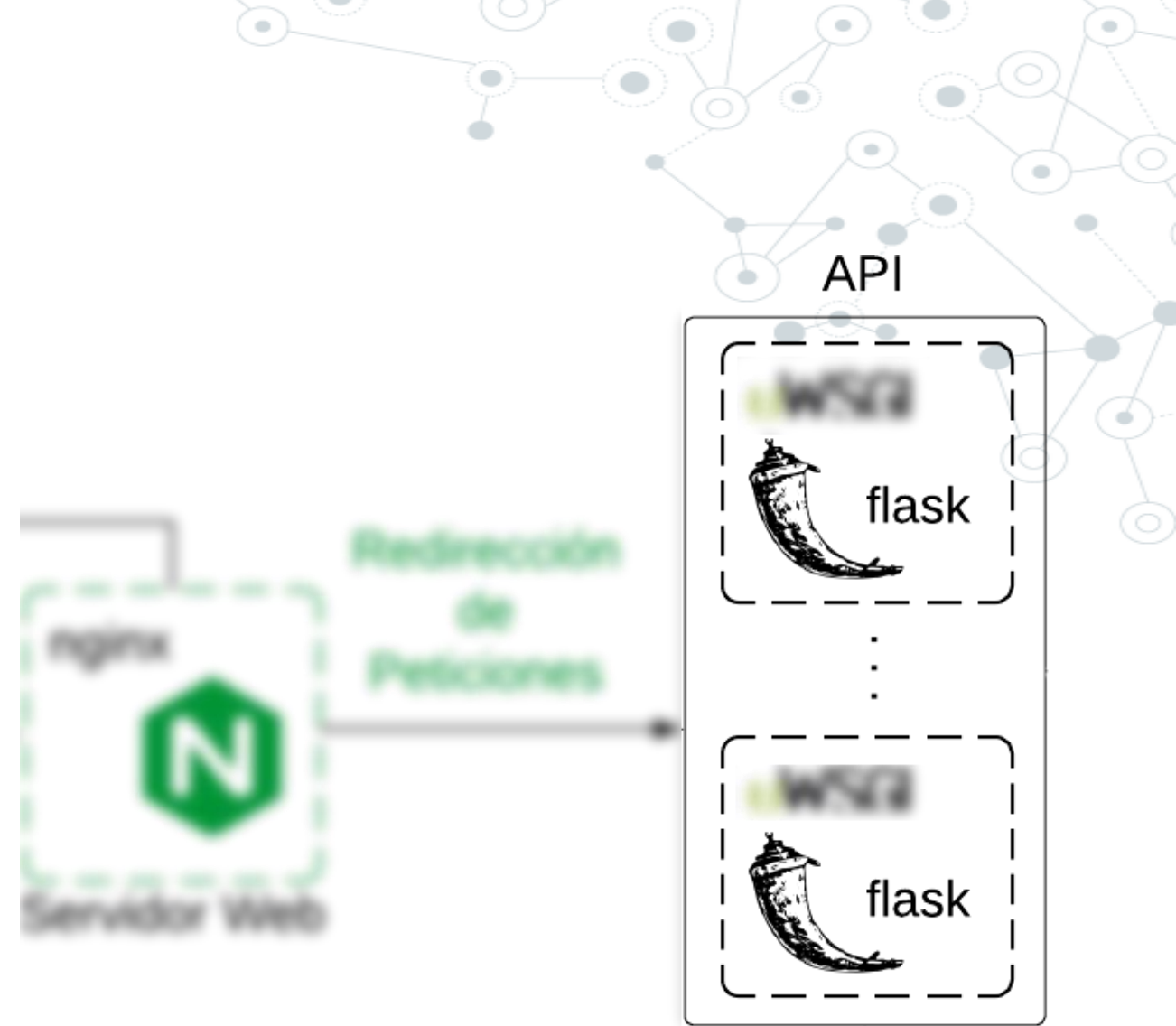
Tecnologías – uWSGI

- ⦿ Aplicación que provee herramientas para el desarrollo y funcionamiento de aplicaciones y servicios web
- ⦿ Se comunica con las aplicaciones haciendo uso de la interfaz WSGI



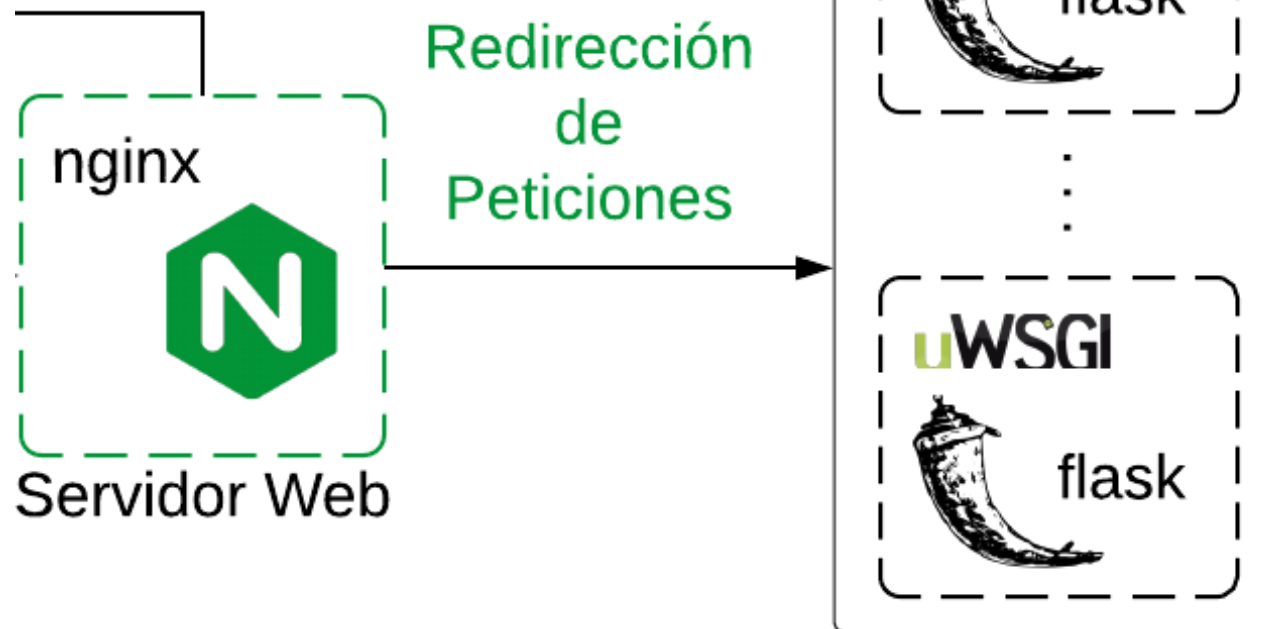
Tecnologías - Flask

- ⦿ Framework para el desarrollo de aplicaciones web
- ⦿ Se concentra en mantener el núcleo de la aplicación simple



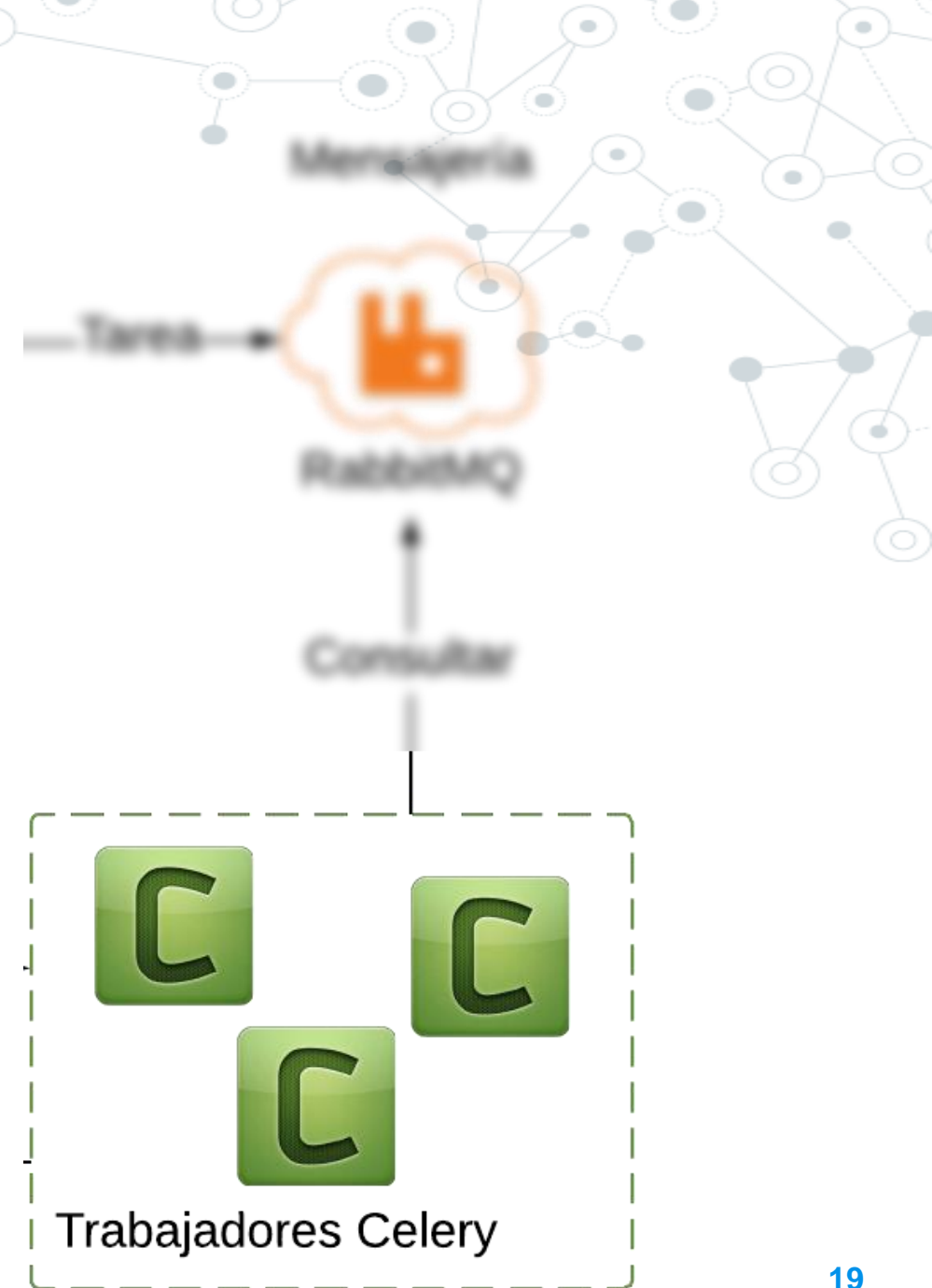
Tecnologías - Atención de Peticiones

- Flask ofrece herramientas de trabajo
- Nginx ofrece balance de carga
- uWSGI permite servir paginas dinámicas implementadas con Flask



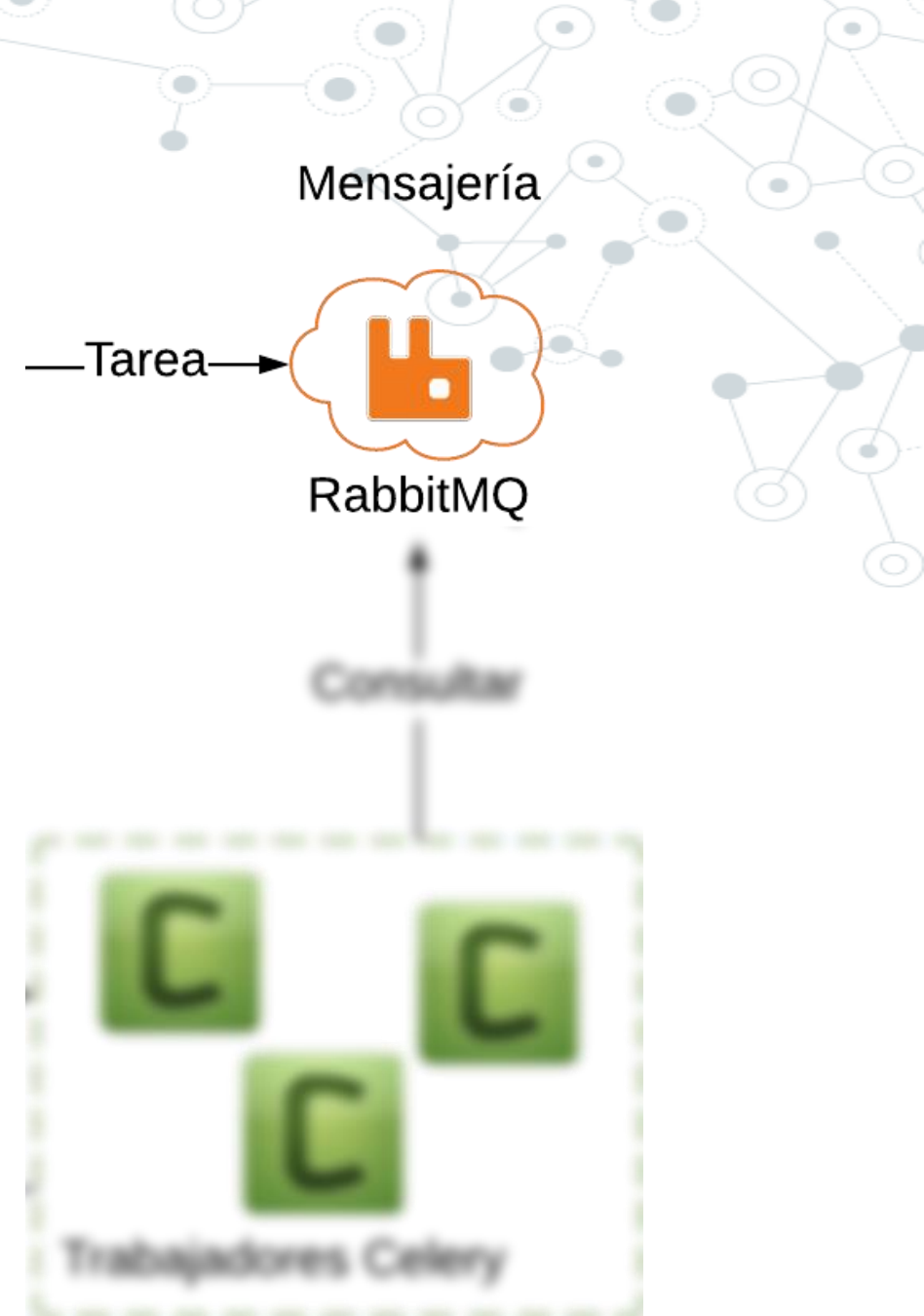
Tecnologías - Celery

- ⦿ Permite la creación de colas y tareas de ejecución asíncronas
- ⦿ Las colas de tareas son usadas para la distribución de trabajo entre múltiples hilos o máquinas
- ⦿ Hace uso de la transmisión de mensajes en un ambiente distribuido



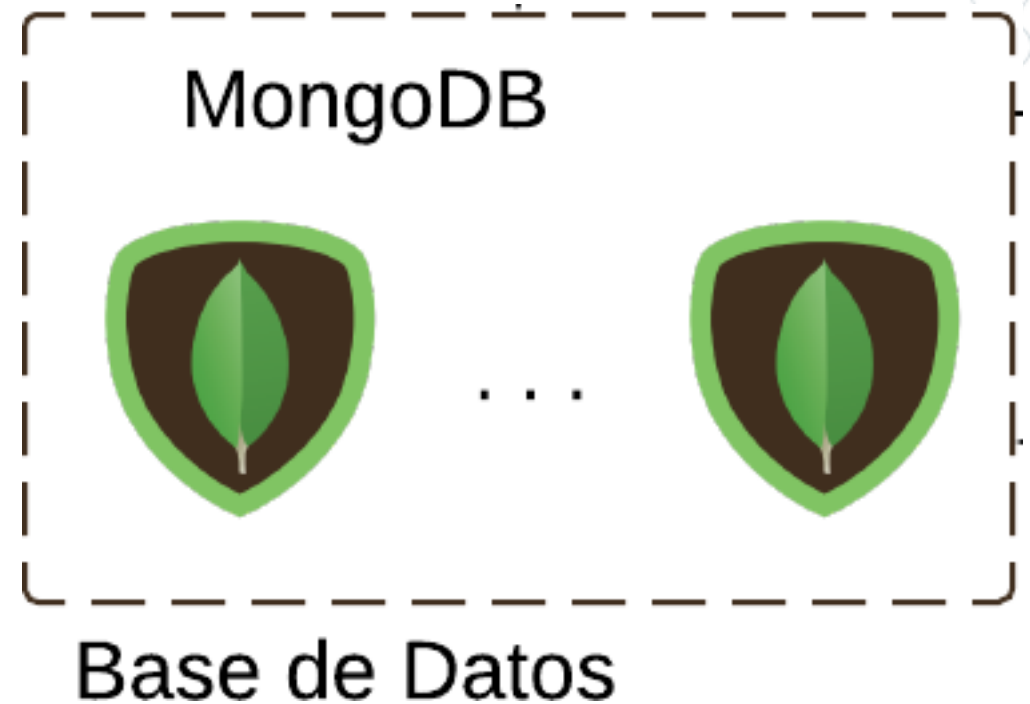
Tecnologías - RabbitMQ

- Es un intermediario para la transmisión de mensajes
- Su uso permite la comunicación de mensajes entre los múltiples componentes del sistema
- Un mensaje puede incluir diversos datos.



Tecnologías - MongoDB

- ◎ Base de datos NoSQL orientada a documentos
- ◎ Cada documento es un objeto JSON
- ◎ Estructura flexible
- ◎ Escalabilidad (Sharding)



Tecnologías - MongoDB

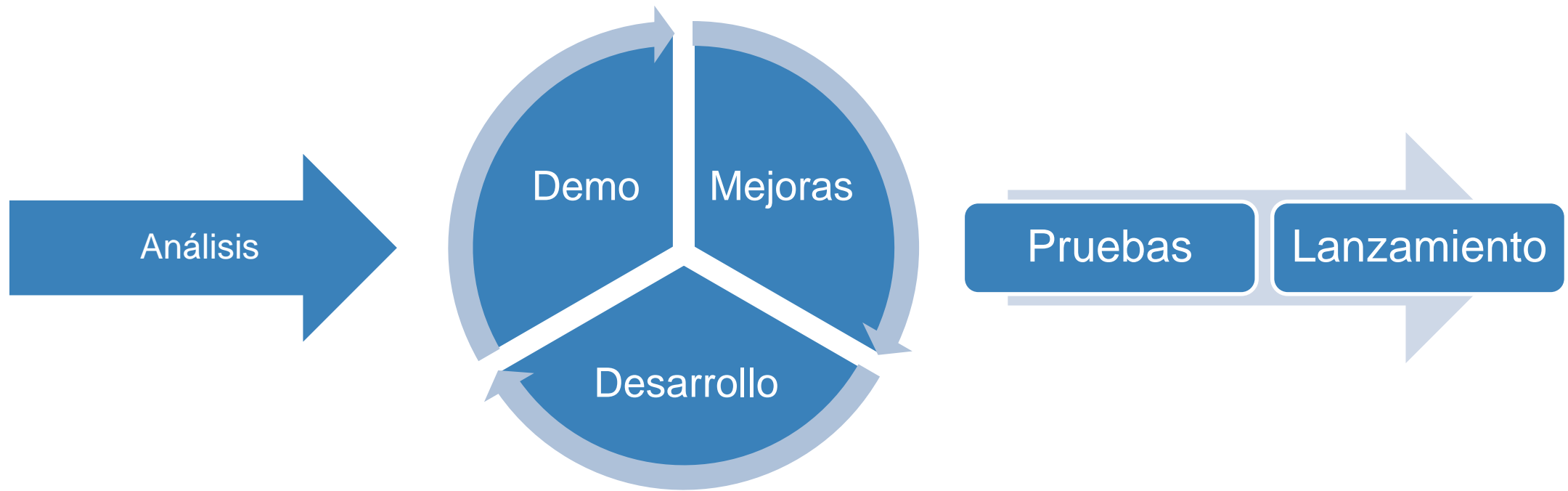
- ◎ Replicación Nativa
- ◎ Balance de Carga
- ◎ Funciones de Agregación
- ◎ MapReduce















Metodología – Rapid Application Development

- ⦿ La dinámica de trabajo se enfoca en la implementación de componentes pequeños en cortos períodos de tiempo
- ⦿ Su ciclo de vida se basa en el desarrollo de prototipos de forma iterativa
- ⦿ Retroalimentación con el cliente y revisiones constantes

Metodología – Rapid Application Development



Metodología – Rapid Application Development

	 Sorting versions of articles (revisions) by timestamp #39 by testica was closed on Sep 29, 2017		 4
	 Filter queries by date using YYYY-mm-dd format #38 by frankjdelgado was closed on Sep 27, 2017		 1
	 Improve queries.py to accept filters by whitelist enhancement #35 by korenerok was closed on Sep 14, 2017		 1

Metodología – Rapid Application Development

Missing requirements and troubles #21

 Closed

testica opened this issue on Aug 14, 2017 · 4 comments



testica commented on Aug 14, 2017 • edited ▾

Contributor



- Python `2.7.10` doesn't include `pip` according to [this](#), so it is an extra requirement
- Missing step `sudo pip install -r requirements.txt`

I got several issues installing requirements.txt :

1. Package `PAM == 0.4.2` doesn't exist, I had to change to `PAM==0.1.2`
2. Packages `Twisted-*` (* Conch, Core, Mail, Names, Web) don't exist, I found those packages are included in Twisted.
3. Package `adium-theme-ubuntu==0.3.4` doesn't exist (is necessary ?).

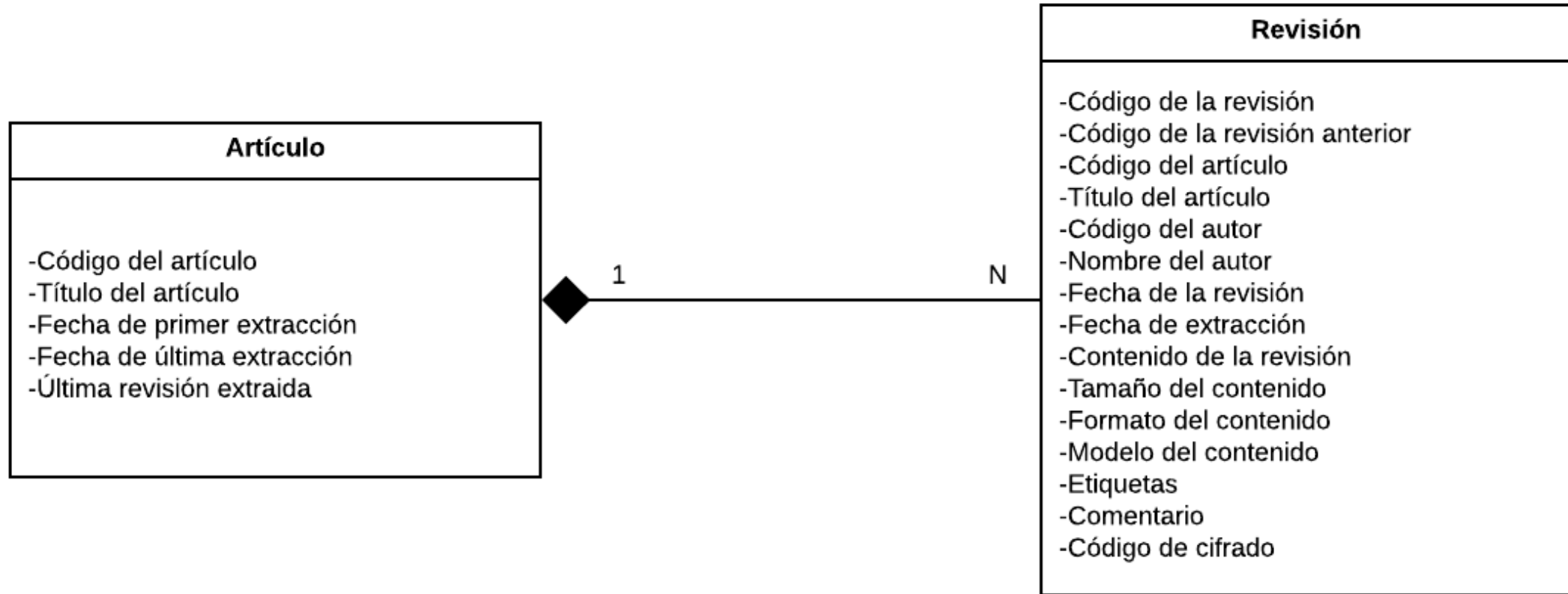
Componentes

- ◎ Base de Datos.
- ◎ API.
- ◎ Algoritmo de Revisita.
- ◎ Worker.
- ◎ Monitor.
- ◎ Docker.

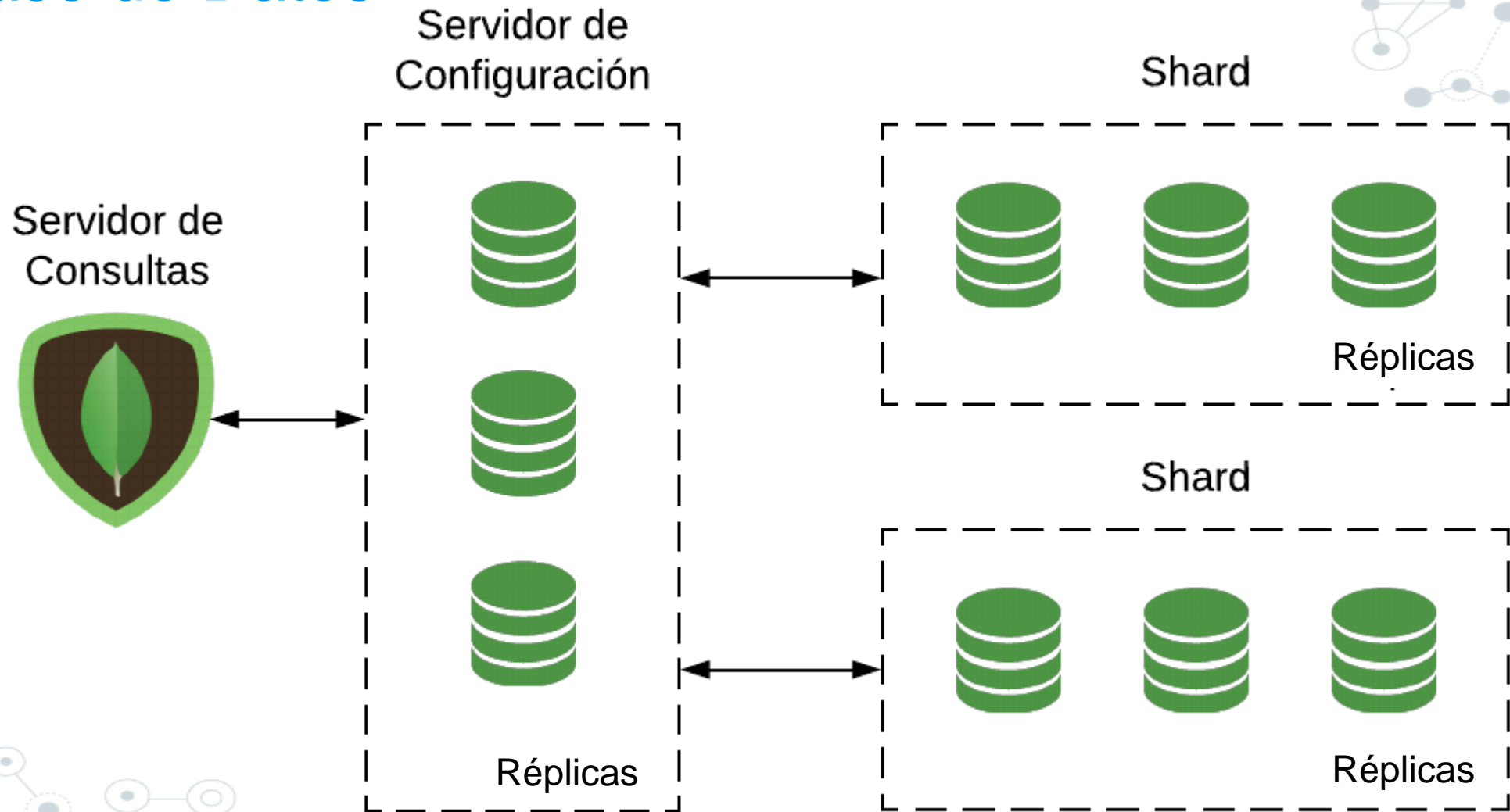
Base de Datos

- © Posee 2 colecciones principales: artículos y revisiones.
- © Implementa réplicas para brindar alta disponibilidad y tolerancia a fallos.
- © Se fragmentan los datos para distribuirlos sobre múltiples nodos.

Base de Datos



Base de Datos



API

- ⦿ Módulo encargado de realizar las consultas y extracción de revisiones.
- ⦿ Se proveen distintas rutas de acceso, para realizar las operaciones.
- ⦿ Los parámetros recibidos son utilizados para especificar, filtrar, ordenar y paginar.

API

- ◎ Las consultas son asignadas a una cola de tareas de Celery, que será utilizada por los trabajadores.
- ◎ Las respuestas de las consultas se presentan en formato JSON.
- ◎ Las rutas de acceso pueden separarse en 3 categorías:
 - Extracción de revisiones.
 - Consultas de datos.
 - Consulta de métricas.



API - Extracción de revisiones

- ⦿ Se recibe el título o URL de un artículo.
- ⦿ Se encola la tarea para la extracción. El estado de la misma puede verificarse.
- ⦿ El proceso se realiza desde la última revisión extraída.

API - Extracción de revisiones

- ⦿ Se puede utilizar el parámetro opcional para selección de idioma.
- ⦿ Los resultados son almacenados y actualizados en la base de datos.

API - Extracción de revisiones

⦿ Petición

`http://localhost/api/v1/extract?title=RabbitMQ&locale=es`
Título Idioma

`http://localhost/api/v1/extract?url=https://en.wikipedia.org/wiki/The_Lord_of_the_Rings`
Idioma Título

⦿ Respuesta

```
{  
  "Location": "http://localhost/api/v1/status/34842158-fb95-456b-9284-ada8c31b5fd4?name=extract_article"  
}
```

Progreso

```
{  
  "state": "IN PROGRESS",  
  "status": "100 revisions extracted"  
}
```



API - Consultas de datos

- ⦿ Pueden consultarse las revisiones y los artículos.
- ⦿ Se pueden utilizar parámetros adicionales para filtrar los resultados, paginarlos, ordenarlos y/o mostrarlos.

API - Consultas de datos

⦿ Petición

localhost/api/v1/revisions?pageid=146232

⦿ Respuesta

```
0:
  comment:      "Inicio Artículo"
  size:         155
  contentformat: "text/x-wiki"
  pageid:       146232
  tags:         []
  locale:       "es"
  timestamp:    "2005-06-13T20:24:49"
  ▼ *:         "El cerro El Ávila está localizado en la cordillera del litoral
               central, al norte de la ciudad de [[Caracas]], [[Venezuela]].\n
               \n{{Esbozo}}\n{{Endesarrollo}}"
  userid:       35895
  revid:        938069
  contentmodel: "wikitext"
  extraction_date: "2018-04-27T19:00:55.062000"
  parentid:     0
  title:        "Parque nacional El Ávila"
  ▼ _id:
    $oid:       "5ae37367d823524f1de97dbf"
  user:         "Venex"
```

API - Consultas de métricas

- ⦿ Se hacen operaciones para calcular métricas realizadas sobre las revisiones.
- ⦿ Se ofrecen métricas predeterminadas: cantidad, promedio y moda.
- ⦿ Se reciben parámetros que permiten filtrar los datos a operar.

API - Consultas de métricas

- ⦿ Se ofrece una ruta de acceso para crear operaciones mas complejas sobre la base de datos.

```
{
  "$match": {
    "extraction_date": "2017-09-28T02:36:33.452000",
    "pageid": 4606
  },
  {
    "$project" :{
      "pageid": 1 , "timestamp" : 1
    }
  },
  { "$limit" : 5 }
```

Algoritmo de Revisita

- Se utiliza como modelo probabilístico, la distribución exponencial. Se calcula la esperanza matemática en base a la información de los artículos.

$$E(x) = 1 / \lambda$$

$$\lambda = n / (t1 - t0)$$

- El artículo cuyo tiempo desde su última extracción supere su esperanza, es visitado.

Algoritmo de Revisita

- ⦿ Cada artículo tiene su propia esperanza basada en la frecuencia que es actualizado.
- ⦿ Se utiliza una tarea programada para verificar si los historiales de un artículo deben ser actualizados.

Trabajadores de Celery

- ⦿ Tienen la función de tomar tareas de la cola de Celery, y ejecutar el proceso tomado.
- ⦿ Múltiples trabajadores ejecutan sus tareas simultáneamente.

Docker

- ◎ Herramienta encargada de ejecutar las aplicaciones del sistema distribuido.
- ◎ Permite emular un servidor físico en un contenedor, en donde se almacena y ejecuta un servicio o aplicación.
- ◎ Múltiples servicios pueden ser ejecutados al mismo tiempo y comunicarse entre si.

NAME	CPU %	MEM USAGE / LIMIT
wikihistoryextractorapi_flask_2	0.95%	31.73MiB / 3.859GiB
wikihistoryextractorapi_flask_3	0.94%	30.03MiB / 3.859GiB
wikihistoryextractorapi_nginx_1	0.00%	1.484MiB / 3.859GiB
wikihistoryextractorapi_worker_3	0.13%	23.99MiB / 3.859GiB
wikihistoryextractorapi_worker_1	0.13%	24.64MiB / 3.859GiB
wikihistoryextractorapi_worker_2	0.00%	43.8MiB / 3.859GiB
wikihistoryextractorapi_adminmongo_1	0.00%	28.25MiB / 3.859GiB
wikihistoryextractorapi_flask_1	0.91%	31.65MiB / 3.859GiB
wikihistoryextractorapi_flower_1	0.03%	40.33MiB / 3.859GiB
mongo	0.41%	7.941MiB / 3.859GiB
wikihistoryextractorapi_rabbit_1	2.72%	53.56MiB / 3.859GiB
mongocfg3	1.10%	84.52MiB / 3.859GiB
mongors1n3	1.04%	48.57MiB / 3.859GiB
mongors1n1	0.88%	47MiB / 3.859GiB
mongocfg2	1.03%	83.13MiB / 3.859GiB
mongocfg1	0.76%	82.95MiB / 3.859GiB
mongors2n1	0.98%	43.86MiB / 3.859GiB
mongors1n2	1.25%	1.323GiB / 3.859GiB
mongors2n3	1.25%	46.92MiB / 3.859GiB
mongors2n2	0.87%	43.7MiB / 3.859GiB
wiki_history_api	0.00%	7.203MiB / 3.859GiB
wiki_history_db	0.76%	10.01MiB / 3.859GiB
wiki_history_client	0.00%	1.336MiB / 3.859GiB

Docker

- © La configuración de cada servicio puede ser llevada a cabo por medio de archivos YAML llamados *Compose*.

```
1  version: '3'
2
3  services:
4    rabbit:
5      image: rabbitmq:3.6.11
6      restart: always
7      hostname: rabbit
8      environment:
9        - RABBITMQ_DEFAULT_USER=wiki
10       - RABBITMQ_DEFAULT_PASS=wiki123
11      ports:
12        - "5673:5672"
13      volumes:
14        - 'wiki_rabbit:/data'
15      networks:
16        - wiki_network
17
```

Monitor

- 🕒 Flower: consultar el estado de las tareas de Celery y nodos.
- 🕒 Mongo Express: consultar la base de datos.



RabbitMQ



python



Demostración

Conclusiones

- ⦿ Gracias al escalamiento horizontal, se redujeron las limitaciones existentes.
- ⦿ El uso de herramientas facilitó la implementación del sistema distribuido y la comunicación de sus componentes.

Conclusiones

- ◎ El uso de una metodología ágil facilitó el desarrollo de la aplicación, ofreciendo retroalimentación.
- ◎ La recopilación de información y el cálculo de métricas facilitará el trabajo de expertos en la búsqueda de patrones de los datos de los historiales.

Trabajos Futuros y Recomendaciones

- ◎ Implementar base de datos caché.
- ◎ Distribuir el servicio RabbitMQ para la tolerancia a fallos y alta disponibilidad.
- ◎ Ajustar tarea del algoritmo de revisita.
- ◎ Incremento de Máquinas Virtuales en ambiente de trabajo.



¿Preguntas?