



FACULTAD DE  
INGENIERÍA Y CIENCIAS

UAI  
UNIVERSIDAD ADOLFO IBÁÑEZ

# Reducción de costos de Infraestructura TI de web scrapers

**Empresa:** Tembi

**Área:** Tembi Tech Department

**Alumno:** Matthias Müller

**Profesor:** Fernando Vásquez Acuña

**Fecha:** 2do semestre 2023

## **Resumen ejecutivo**

El área de tecnología de la empresa Tembi posee una robusta infraestructura de tecnologías de la información (TI) donde se utilizan múltiples *web scrapers*, los cuales son bots especializados en la extracción de información de diversas páginas web de comercio electrónico. El funcionamiento de estos es bien simple: Acceden directamente a la base de datos para obtener sus tareas asignadas, donde cada tarea corresponde a un dominio específico, para luego extraer la información de la nube y finalmente almacenarlo en la base de datos.

Este flujo de extracción de información ha funcionado correctamente desde un principio, pero a medida que va aumentando la cantidad de clientes también aumenta la cantidad de información que se debe extraer y almacenar. Dada la existente arquitectura el costo asociado al crecimiento no es lineal: Actualmente se quiere expandir a nuevos mercados como Alemania y España, lo que implica que se debe extraer tres veces más dominios, lo que incurriría en costos asociados que aumentan 5,925 veces.

De esto se puede observar que no es sostenible expandirse indefinidamente, es necesario evaluar alternativas para poder crecer como empresa manteniendo un flujo positivo. De esta manera se planteó el objetivo de **reducir los costos asociados de la infraestructura de Tecnologías de la Información (TI) de los web scrapers en un 15% y de esta forma adaptar la arquitectura para garantizar una escalabilidad eficiente para poder enfrentarse a nuevos mercados.**

En primer lugar, se mejoró los web scrapers de manera interna para minimizar el tiempo de ejecución, se implementaron tecnologías para poder reducir el bloqueo de páginas web por detección de bots y se implementaron Logs para el almacenamiento en tiempo real de errores sucedidos internamente. En segundo lugar, se diseñó una arquitectura diferente: Entre la base de datos y los web scrapers se creó un intermediario, el cual se preocupa de asignar las tareas, almacenar la información obtenida para luego almacenarla en la base de datos en menos interacciones.

A través de esta propuesta los costos asociados a la infraestructura TI actual se reducirán 4,5 veces y expandirse a los mismos mercados implica un aumento de costos de 3,0004 veces, lo cual es cercano a representar una relación lineal respecto a costos y cantidad de dominios a extraer información.

## **Abstract**

The Tembi Tech department of Tembi company has a complete information technology (IT) infrastructure where multiple web scrapers are used. These are bots specialized in extracting information from various e-commerce websites. Their operation is simple: They directly access the database to get their assigned tasks, where each task corresponds to a specific domain, and then extract information from the cloud before finally storing it in the database.

This information extraction flow has worked correctly from the beginning, but as the number of customers increases, so does the amount of information that needs to be extracted and stored. Given the existing architecture, the cost associated with growth is not linear: There are plans to expand into new markets such as Germany and Spain, which implies that three times more domains need to be extracted, incurring associated costs that increase by 5,925 times.

From this, it is evident that expanding indefinitely is not sustainable; it is necessary to evaluate alternatives to grow as a company while maintaining a positive flow. Therefore, the goal was set to **reduce the associated costs of the IT infrastructure of the web scrapers by 15%, this way adapting the architecture to ensure efficient scalability to face new markets.**

Firstly, the web scrapers code were internally improved to minimize execution time, technologies were implemented to reduce web page blocking due to bot detection, and Logs were implemented for real-time storage of internal errors. Secondly, a different architecture was designed: An intermediary was created between the database and the web scrapers, which is responsible for assigning tasks, storing the obtained information, and then storing it in the database in fewer interactions.

Through this proposal, the costs associated with the IT infrastructure will be reduced by 4.5 times, and expanding to the same markets will imply a cost increase of 3.0004 times, which is close to representing a linear relationship regarding costs and the number of domains from which to extract information.

## Tabla de contenidos

<b>INTRODUCCIÓN .....</b>	<b>6</b>
CONTEXTO .....	6
<b>DESCRIPCIÓN DEL PROBLEMA.....</b>	<b>7</b>
BASE DE DATOS.....	8
WEB SCRAPERS.....	9
CONSECUENCIAS EN LA ESCALABILIDAD A NUEVOS MERCADOS.....	9
<b>OBJETIVOS SMART .....</b>	<b>13</b>
OBJETIVO GENERAL .....	13
OBJETIVOS ESPECÍFICOS .....	13
<b>MÉTRICAS DE DESEMPEÑO .....</b>	<b>13</b>
<b>ESTADO DEL ARTE .....</b>	<b>15</b>
LITERATURA.....	16
<i>Base de datos</i> .....	16
<i>Indexación de la base de datos y Optimización de querys</i> .....	16
<i>Data sharding y Functional partitioning</i> .....	16
<i>Tecnologías web scrapers</i> .....	17
<i>Tecnologías anti crawling</i> .....	18
<i>Code profilers, Logging y Clean Code</i> .....	19
CASOS DE ÉXITO .....	19
<i>Caso 1</i> .....	19
<i>Caso 2</i> .....	20
<b>PROPUESTAS DE SOLUCIÓN.....</b>	<b>21</b>
LÓGICA INTERNA DE SCRAPERS .....	21
ARQUITECTURA TI .....	22
<b>SOLUCIÓN ESCOGIDA .....</b>	<b>22</b>
<b>ANÁLISIS DE RIESGO.....</b>	<b>23</b>

<b>METODOLOGÍA Y PLAN DE IMPLEMENTACIÓN .....</b>	<b>25</b>
<b>DESARROLLO E IMPLEMENTACIÓN .....</b>	<b>27</b>
CONFIGURACIÓN DE PROYECTO SCRAPERS PARA LA RECOPILACIÓN DE INFORMACIÓN PARA REGRESIÓN LINEAL .....	27
IMPLEMENTACIÓN ROBUSTA DE LOGGING.....	28
IMPLEMENTACIÓN DE TRABAJO EN LOTES Y PROXIES.....	30
<i>Proxies.....</i>	30
<i>Implementación de trabajo por lotes .....</i>	30
EVALUACIÓN DE CAMBIOS .....	33
REDIS PARA LA ASIGNACIÓN DE TAREAS Y MEJORA DE ÍNDICES Y CONSULTAS.....	34
<i>1.Máquina Docker REDIS.....</i>	35
ESTABLECER LOS PARÁMETROS DE LA HERRAMIENTA SCRAPY .....	36
<i>Regresión lineal tiempo de ejecución .....</i>	37
<i>Regresión lineal error .....</i>	37
<b>RESULTADOS .....</b>	<b>39</b>
<b>EVALUACIÓN ECONÓMICA .....</b>	<b>41</b>
<b>CONCLUSIONES.....</b>	<b>45</b>
<b>RECOMENDACIONES .....</b>	<b>45</b>
<b>BIBLIOGRAFÍA .....</b>	<b>47</b>
<b>ANEXOS.....</b>	<b>48</b>
1. DESARROLLO TIEMPO DE EJECUCIÓN DE CONSULTADAS EN FUNCIÓN DEL TAMAÑO DE LA BASE DE DATOS .....	48
2. CONFIGURACIÓN DE PROYECTO SCRAPERS PARA LA RECOPILACIÓN DE INFORMACIÓN .....	54
3. IMPLEMENTACIÓN ROBUSTA DE LOGGING .....	58
3. IMPLEMENTACIÓN DE TRABAJO EN LOTES Y PROXIES.....	59
4. EVALUACIÓN DE CAMBIOS .....	61
5. REDIS PARA LA ASIGNACIÓN DE TAREAS Y MEJORA DE ÍNDICES Y CONSULTAS.....	61
<i>1.Máquina Docker REDIS.....</i>	62
<i>2.Creador de tareas .....</i>	63
<i>3.Meta scraper .....</i>	64

4. Procesador de ítems .....	65
6. ESTABLECER LOS PARÁMETROS DE LA HERRAMIENTA SCRAPY.....	66
7. RELACIÓN NÚMERO DE DOMINIOS CON EL TAMAÑO DEL MERCADO .....	71

## Introducción

### Contexto

Tembi es una startup danés ubicada en Copenhague, la cual fue fundada el año 2022 como spin off de DAMVAD Analytics. La empresa proporciona un servicio de inteligencia prescriptiva, lo que significa que recopila datos públicos de diversas fuentes, los organiza de manera estructurada y luego los presenta a los clientes para facilitar una toma de decisiones más informada. Esta solución se ofrece con el modelo Software as a Service (SaaS), es decir, los usuarios finales acceden al servicio a través de página web y están sujetos a una tarifa de suscripción mensual.

En la solución SaaS, Tembi ofrece los siguientes productos:

- **Real estate (Bienes raíces):** Se recopila información detallada de diversas empresas y, mediante el análisis de estos datos, se calcula la probabilidad de que una compañía se traslade de una ubicación X a una ubicación Y. Los profesionales del sector inmobiliario pueden utilizar esta valiosa información para establecer contacto con las empresas y ofrecerles espacios comerciales adecuados a sus necesidades.
- **E-commerce Intelligence Platform:** Se recopila información detallada del proceso de pago (check-out) en plataformas de comercio electrónico, incluyendo opciones de envío, precios y plazos de entrega. Esta información se ofrece a empresas de servicios de entrega para que puedan tomar decisiones informadas considerando la competencia y clientes.

Es importante mencionar que, para recopilar información, Tembi emplea robots de software conocidos como "*web scrapers*", que navegan por diversas páginas web con el objetivo de extraer los datos.

En el último año, la empresa emergente ha experimentado un crecimiento significativo, expandiendo su equipo de 6 a 16 empleados y actualmente tiene abiertas varias posiciones para seguir fortaleciendo su presencia. Tembi tiene operaciones en países nórdicos como Noruega, Holanda, Finlandia, Dinamarca y Suecia, recientemente ha iniciado operaciones en Lituania, Letonia y Estonia. Entre sus clientes destacados se encuentran PostNord, DHL, Bring, Posti y GLS. Dado su alto crecimiento, la empresa está en proceso de exploración para expandirse a nuevos mercados.

## Descripción del problema

Para tener un correcto entendimiento del problema es necesario entender la estructura de la arquitectura TI de los web scrapers, actualmente, se cuenta con una base de datos MYSQL 8.10 que incluye réplicas de lectura. Esto implica que las consultas de lectura pueden distribuirse entre múltiples servidores, mejorando así la eficiencia y reduciendo los tiempos de espera asociados con la recuperación de datos.

Respecto a los scrapers, la empresa posee diferentes bots que extraen información de diferentes fuentes, este proyecto se enfocará específicamente en el *meta\_scaper*, el cual se especializa en recopilar información general de páginas de comercio electrónico, proporcionando un panorama completo del sitio como lo son título, descripción, productos, categorías, correo de contacto entre otros componentes.

El problema se puede analizar en dos diferentes aspectos, todos relacionados directamente con la Infraestructura TI, la cual es la siguiente:

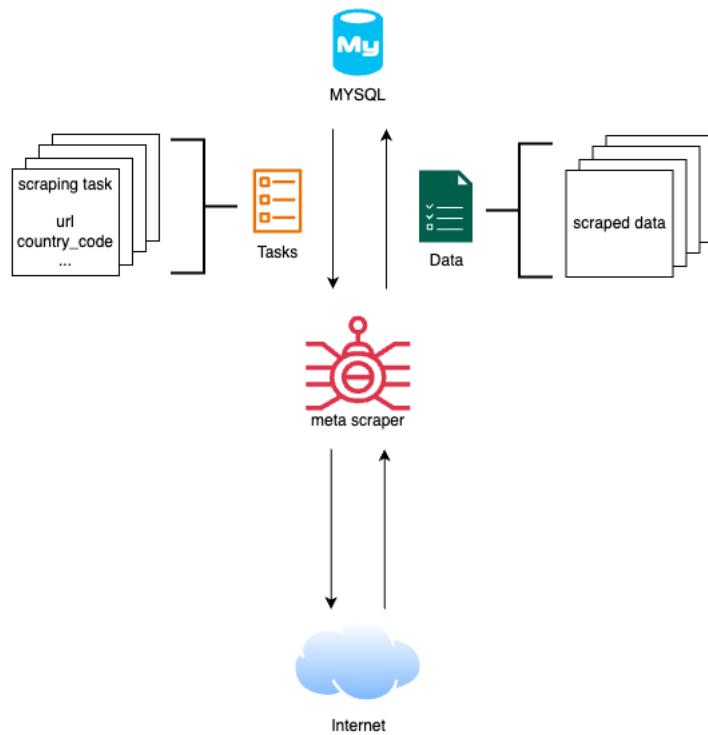


Ilustración 1: Arquitectura de meta scraper.<sup>1</sup>

De esta forma, el *meta\_scraper* consulta a la base de datos a que dominios realizar web scraping, los cuales llamaremos **tareas**. Este procede a obtener la información de la nube de los dominios y una vez hecho esto, los almacena en la misma base de datos.

## Base de datos

La base de datos representa un 32% de los costos asociados de la infraestructura TI, en sólo los últimos seis meses casi se ha triplicado (2,766 veces) la cantidad de datos históricos de páginas web, lo que ha impactado en la ejecución de consultas. Actualmente la tabla con más datos posee 13,422,064 filas. Las consultas más comunes toman 10 en promedio segundos en ejecutarse.

---

<sup>1</sup> Elaboración propia.

```
scrapers_NEW> select * from meta_scraper_view  
    order by rand()  
    limit 1  
[2023-12-04 15:00:08] 0 rows retrieved in 31 s 511 ms (execution: 31 s 477 ms, fetching: 34 ms)
```

Ilustración 2: Ejemplo tiempo de respuesta de 31,51 segundos de una solicitud a la base de datos.<sup>2</sup>

## Web Scrapers

Los Web Scrapers representan un 45% de los costos asociados de la infraestructura TI. Son los responsables de obtener la información crucial directamente de las páginas web. Son fundamentales para la empresa, dado que su modelo de negocio se enfoca en la extracción y análisis de datos, solo en los últimos 6 meses se han extraído 8,25 veces más datos de la web.

Consideraciones importantes:

- Se busca información múltiples veces para la misma página en vez de realizarlo una única vez.
- Los web scrapers solicitan información a la base de datos de forma secuencial, es decir uno a uno en vez de solicitar por grupos (lo que llamaremos **chunking**), esta metodología, combinada con la lenta velocidad de ejecución de las consultas, resulta en un total de 540,723 horas dedicadas exclusivamente a la espera de respuestas de la base de datos.
- La obtención de información de cada página web también se hace de forma secuencial, en consecuencia, el tiempo de respuesta de cada solicitud se vuelve un cuello de botella.
- En promedio, se utilizan 40.6223 segundos en obtener la información de un dominio para el **meta\_scraper**.

## Consecuencias en la escalabilidad a nuevos mercados

Para comprender adecuadamente el problema, es crucial entender lo que implica la expansión a un nuevo mercado para Tembi. Actualmente, la empresa brinda sus servicios en seis mercados distintos, cada uno con una variedad de clientes.

---

<sup>2</sup> Elaboración propia.

En el proceso de entrada a un nuevo mercado es esencial extraer una cantidad cercana al total de información para poder ofrecer una solución que entregue valor a sus clientes.

Si se considera el supuesto que hay una correlación positiva de 0,845<sup>3</sup> entre el número de páginas e-commerce y el tamaño de mercado, se puede evaluar la dimensión de recopilación de datos de un país en particular.

Como se puede ver en los siguientes gráficos, el mercado actual contempla un monto de 80,08 mil millones de euros en el año 2021, mientras que mercados de interés de expansión para la empresa como Alemania y España componen un monto equivalente o incluso mayor a la suma de todos estos.

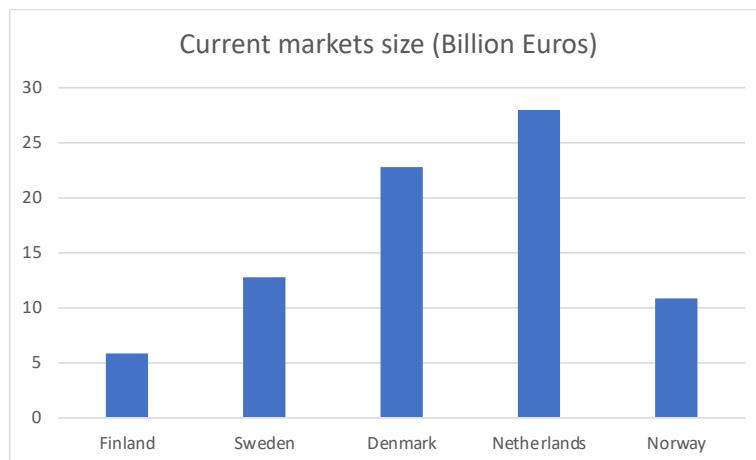
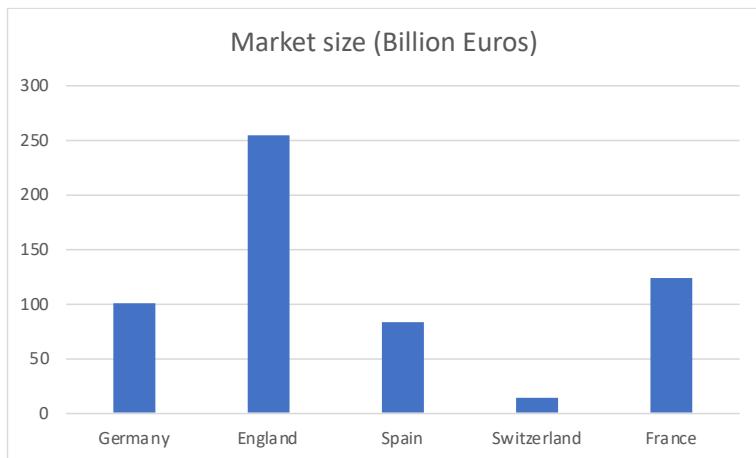


Gráfico 1: Tamaño en mil millones de euros de mercados actuales de Tembi.<sup>4</sup>

<sup>3</sup> Tabla en anexo 7. Relación número de dominios con el tamaño del mercado.

<sup>4</sup>Fuente: Europe: e-commerce market size, by Country 2021 | Statista. (2023, 30 agosto). Statista. <https://www.statista.com/statistics/1113005/market-size-of-e-commerce-in-europe-by-country/>



*Gráfico 2: Tamaño en mil millones de euros de mercados deseados a expansión en un plazo de un año o menos.<sup>5</sup>*

Para poder entender correctamente la relación del tamaño de base de datos con el impacto en el rendimiento de esta, se creó una base de datos idéntica en estructura a la original, se llenó con datos aleatorios y se midió el rendimiento de las dos consultas más comunes y luego se estimó su tiempo de respuesta a través de una regresión lineal.

---

<sup>5</sup> Fuente: *Europe: e-commerce market size, by Country 2021 / Statista.* (2023, 30 agosto). Statista. <https://www.statista.com/statistics/1113005/market-size-of-e-commerce-in-europe-by-country/>

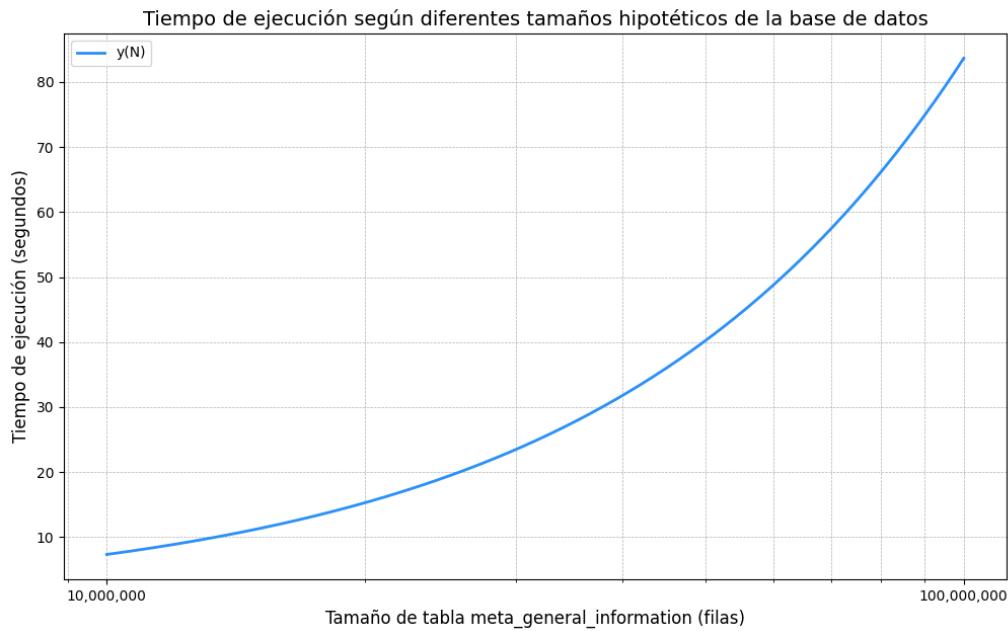


Gráfico 3: Tiempo de ejecución de consultas de lectura según diferentes tamaños hipotéticos de la base de datos.<sup>6</sup>

De esta relación se puede observar que, si se duplica la cantidad de dominios, el tiempo de consulta aumenta de 14.06 segundos a 29.41 segundos. Si se triplica aumenta a 45.138 segundos. Hay que considerar que solo para expandirse a Alemania habría que duplicar la cantidad de dominios.

Por otro lado, evaluando los costos asociados a realizar web scraping, si se quiere triplicar la cantidad de dominios, los costos asociados a obtener tal información aumentan 5,925 veces, lo que impacta significativamente en la rentabilidad del negocio.

De esto se puede observar que, si Tembi se quiere expandir a nuevos Mercado, primero necesitará adaptar su arquitectura TI para poder enfrentarse a estas demandas.

De esto surge la problemática: La existente arquitectura TI es ineficiente y posee altos costos, lo que representa una barrera significativa para la escalabilidad del negocio, dificultando el proceso de expansión a nuevos mercados para el crecimiento de la empresa.

---

<sup>6</sup> Elaboración propia. Desarrollo en Anexo 1.

## Objetivos Smart

### Objetivo General

En un plazo de cuatro meses, reducir los costos asociados de la infraestructura de Tecnologías de la Información (TI) de los web scrapers en un 15% y de esta forma adaptar la arquitectura para garantizar una escalabilidad eficiente para poder enfrentarse a nuevos mercados.

### Objetivos específicos

1. Reducir el tiempo de ejecución de scrapers en un 30%.
2. Optimizar las transacciones y la gestión de almacenamiento de la base de datos para conseguir una reducción del 20% en los costos operativos relacionados a los web scrapers.
3. Implementar monitoreo y manejo de errores para los scrapers en su totalidad.

El monitoreo es esencial en un sistema escalable, se necesita tener información lo más robusta y completa posible para la toma rápida de decisiones.

## Métricas de desempeño

Las métricas de desempeño que se emplearán en el proyecto son las siguientes:

- a. Tiempo medio de extracción de ítem

Donde un ítem representa un dominio e-commerce. Es importante mencionar que un dominio puede poseer varias páginas a las que hay que acceder.

$$\text{Tiempo Medio de Extracción (segundos)} = \frac{\text{Tiempo Total de Extracción}}{\text{Número de ítems Extraídos}}$$

- b. Tasa de errores en extracción de ítem

$$\text{Tasa de Errores} = \frac{\text{Número de ítem incompletos}}{\text{Número de ítems}} \times 100$$

La extracción de un ítem posee un error cuando no se extrae la totalidad de su información de un dominio. Se quiere minimizar, ya que la extracción incompleta de información implica volver a ejecutar tal tarea.

- c. Tiempo de consulta a la base de datos por ítem en segundos

El tiempo de consulta considera: Obtener, insertar y actualizar datos de la base de datos.

$$\text{Tiempo de Consulta por Ítem} = \frac{\text{Tiempo Total de Consultas}}{\text{Número de Ítems Consultados}}$$

- d. Tiempo promedio de ejecución de las diez consultas más comunes del *meta\_scrapers*.

$$\text{Tiempo Promedio de Consultas} = \frac{\text{Tiempo Total de Consultas}}{\text{Número Total de Consultas}}$$

- e. Costo mensual de la base de datos.

Este valor se obtiene directamente de los servicios de hosting.

$$\text{Costo Mensual de la DB} = \text{Costo Base en AWS}$$

- f. Tasa de consultas fallidas o abandonadas

Una consulta fallida o abandonada se traduce en un volver a ejecutarla, por lo que duplica los costos asociados, lo cual se busca minimizar.

$$\text{Tasa de Consultas Fallidas o Abandonadas} = \frac{\text{Número de Consultas Fallidas o Abandonadas}}{\text{Número Total de Consultas}} \times 100$$

- g. Porcentaje de Funciones con Manejo de Errores Integrado.

Si ocurre un error, el código no debe detenerse y continuar con las implicancias necesarias.

*Porcentaje de funciones con manejo de errores integrado*

$$= \frac{\text{Número de funciones de } \textit{meta_scrapers} \text{ con manejo de error}}{\text{Total de funciones}}$$

h. Porcentaje de Manejos de Errores con registro almacenado

Si ocurre un error, esto debe estar registrado en algún lado.

Porcentaje de Manejos de Errores con registro almacenado

$$= \frac{\text{Número de funciones de } \text{meta\_scraper} \text{ con registro almacenado}}{\text{Total de funciones}}$$

Las métricas poseen los siguientes valores actuales y esperados:

KPI	Unidad	Valor Inicial	Valor esperado
Tiempo medio de extracción	Segundos	41,982	29,3874
Tasa de errores de extracción de dominio	%	5,87%	5%
Tiempo de consulta a la base de datos por ítem en segundos	Segundos	16,1	10
Tiempo promedio de ejecución de las diez consultas más comunes	Segundos	10,3	5
Costo mensual de la base de datos	Unidades monetarias	19,841	15,8728
Tasa de consultas fallidas o abandonadas	%	4,63%	0%
Porcentaje de Funciones con Manejo de Errores Integrado	%	85%	100%
Porcentaje de Manejos de Errores con Registro almacenado	%	73,20%	100%

Tabla 1: Valores actuales y esperados de métricas de desempeño.<sup>7</sup>

## Estado del Arte

Para el correcto entendimiento del problema, se realizó una evaluación de la literatura existente al respecto. Dada la abrumante cantidad de información que se puede encontrar al respecto se decidió utilizar la siguiente estrategia de búsqueda para la literatura:

1. Fuentes confiables como libros, papers.
2. Este debe contener información directamente relacionada con:
  - a. Bases de datos MYSQL.
  - b. Web scraping a través de Scrapy.
  - c. Estrategias generales de web scraping.

---

<sup>7</sup> Elaboración propia.

## Literatura

Se realizó una evaluación independiente de cada aspecto respecto a los diferentes desafíos presentados.

### Base de datos

Las bases de datos son un tópico delicado en una empresa, a fin de cuentas, la información es lo que más valor tiene dentro de una empresa y estas son las encargadas de almacenarla y entregarla.

Para una comprensión profunda del estado actual de la problemática, se consultó el libro *High Performance MySQL: Proven Strategies for Operating at Scale* de Silvia Botros y Jeremy Tinley (Botros & Tinley, 2022). Este recurso resulta especialmente relevante, ya que se alinea con la infraestructura existente en Tembi: una base de datos MySQL que opera con el motor InnoDB y que se encuentra en la versión 8.0.

De forma resumida, los tópicos más relevantes que explica el libro son los siguientes:

#### *Indexación de la base de datos y Optimización de querys*

“Before you dive into scalability bottlenecks, make sure you’ve optimized your queries, checked your indexes, and have a solid configuration for MySQL.” (Botros & Tinley, 2022, l. 6572-6573)

Los índices en la base de datos cumplen la misma función que las de un libro: A través de ellas puedo ubicar rápidamente la información, lo cual hace la escritura y lectura más rápida a la base de datos, a pesar de ello es importante seleccionar correctamente la cantidad de índices, ya que se deben actualizar cada vez que se actualiza la información.

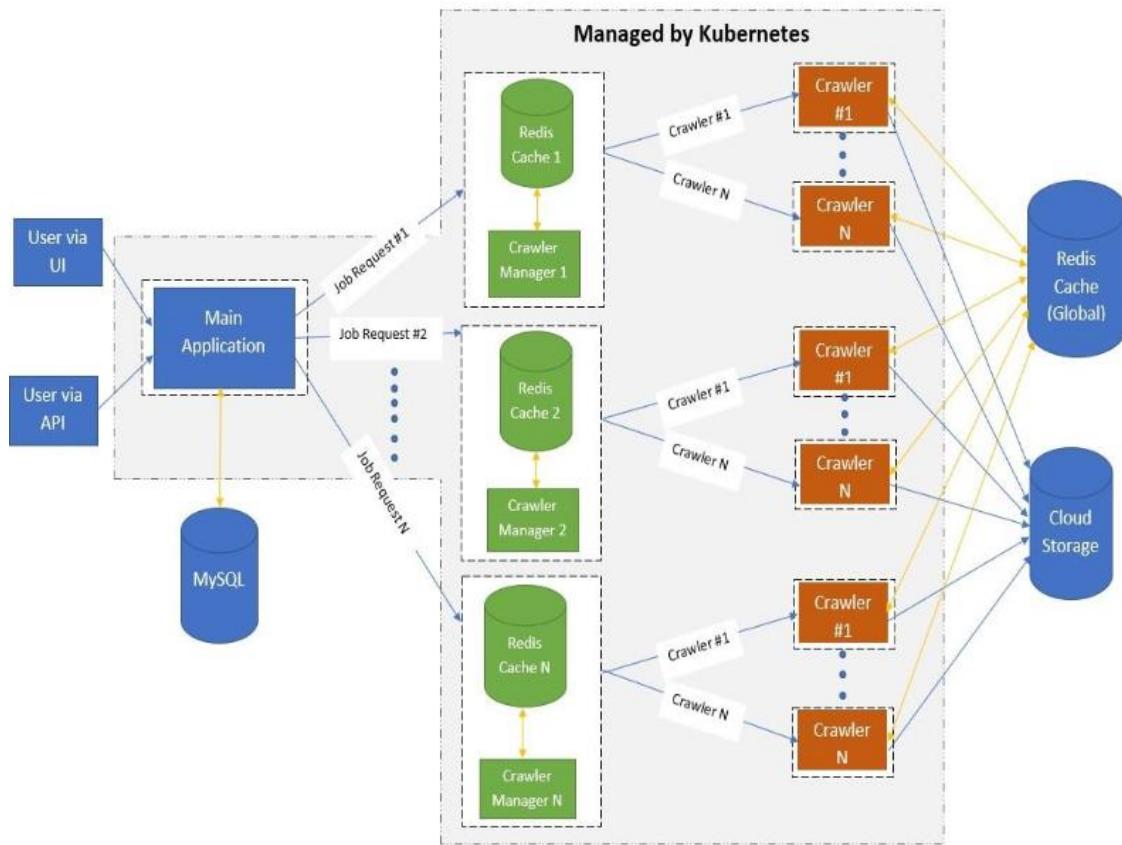
#### *Data sharding y Functional partitioning*

“Functional partitioning, or division of duties, means dedicating different nodes to different tasks. An example of this might be putting user records on one cluster and their billing on a different cluster.” (Botros & Tinley, 2022, l. 6350-6352)

Ambas son soluciones que subdividen la base de datos en conjuntos más pequeños, de esta forma es más fácil encontrar y actualizar información. Esto trae grandes desafíos, como mantener la consistencia y restricciones entre los diferentes conjuntos y realizar consultas reduciendo el número de grupos al que se solicita información.

## Tecnologías web scrapers

La gran mayoría de las investigaciones en este ámbito sugieren un adaptarse a la escalabilidad horizontal, es decir, si se quiere aumentar la páginas a realizar web scraping, hay que utilizar más máquinas que realicen esta tarea. En la gran mayoría de documentación disponible respecto a escalabilidad, se sugiere implementar un modelo en el cual un coordinador central, denominado "maestro", es responsable de asignar tareas específicas a nodos subordinados. Estos nodos se especializan en la extracción de datos de sitios web, permitiendo así una división eficiente del trabajo y optimizando el proceso de Web Scraping.



*Ilustración 3: Modelo de maestro subordinado para web scraping distribuido a través de Scrapy.<sup>8</sup>*

En la teoría se habla de utilizar tecnologías como Docker y Kubernetes para tener varios contextos de coordinador-subordinado (Prusty et al., 2020), y también se han propuesto soluciones para una asignación óptima de trabajos a los subordinados “*Combining dynamic load balancing techniques with consistent hashing algorithm, we designed a slice-based consistent hashing dynamic load balancing strategy for REDIS distributed storage system.*” (Ying et al., 2018).

#### Tecnologías anti crawling

Las páginas web implementan diferentes soluciones para evitar que bots puedan acceder a sus páginas, tales como CAPTCHAs, restricciones de IP, análisis de comportamiento de usuarios, y limitación de tasas de solicitud.

Es por esto por lo que se recomienda agregarle tecnologías anti-scraping para reducir la cantidad de errores y así aumentar la velocidad. En la teoría se destacan existen tres grandes soluciones:

1. VPN, donde se asigna una IP diferente al bot.
2. Proxies, donde cada solicitud que se realice a una página tendrá asignado una IP diferente con un historial positivo.
3. Challenge-solvers en los casos de las páginas más exigentes que exponen desafíos a sus solicitantes, como captchas, renderizar JavaScript o esperar una cierta cantidad de segundos.

Implementar estas soluciones trae consigo gran complejidad, pero a la vez una mejora significativa en el rendimiento; “The experiment results show that after adopting the distributed and anti-crawler prevent technology, the number of crawlers to the target increases by 10%, and the time is shortened by 70%.” (Han & Zheng, 2020).

---

<sup>8</sup> Fuente: Prusty, A., Mejia, O., Shah, A., Kancherlapalli, P., Suresh, A., & Schiebel, R. (2020). Horizontally Scalable Web Crawler using Containerization and a Graphical user Interface. International Journal of Engineering Research & Technology, 9(05), 22780181.

## Code profilers, Logging y Clean Code

"It's always a good idea to make sure an algorithm really is a bottleneck before investing your precious time trying to improve it." (Hunt & Thomas, 1999)

Si bien mejorar la arquitectura TI puede tener un impacto significativo, es fundamental primero evaluar si el código presente es el cuello de botella en el proceso. Es por esto por lo que la literatura recomienda el uso de herramientas para evaluar el código y así encontrar de manera acertada que está sucediendo el código. La primera herramienta crítica para un proceso de reducción de tiempo de ejecución son los code profilers: A través de ellos se pueden obtener los tiempos de ejecución y cuantas veces se llama a una función. Por otro lado, las herramientas de registro ("logging") son indispensables para supervisar la actividad de los scrapers en el entorno de producción; con su ayuda, podemos observar lo que sucede en tiempo real en estas máquinas.

Es importante mencionar que la consistencia en el código es vital, ya que permite una colaboración más efectiva, facilita la comprensión de lo que está ocurriendo, lo que trae consigo una reducción de la deuda técnica a largo plazo. Es por esto por lo que, en el desarrollo de este proyecto se están consultando libros como "*Design Patterns: Elements of Reusable Object-Oriented Software*" (Gamma et al., 1994) y "*Clean Code: A Handbook of Agile Software Craftsmanship*" (Martin, 2008) para guiar la creación de un código robusto y mantenable.

## Casos de éxito

Debido a que esta problemática es muy específica a empresas en crecimiento, lo mejor es observar empresas donde haya surgido una necesidad de adaptar la infraestructura TI para nuevas demandas.

### Caso 1

Se realizó una reunión con Carlos Véliz, quien se desempeña actualmente como director técnico (CTO) de una empresa destacada en el rubro, donde posee más de ocho años de experiencia en desarrollo de Web Scrapers a escala.

#### *Problemática*

La empresa se encontraba con elevados costos de la Infraestructura TI, los cuales impactaban críticamente en la rentabilidad de su solución SaaS. Por otro lado, existían varios procesos donde el cliente tenía que

esperar un tiempo significativo para que se completara la solicitud, haciendo que la experiencia de usuario fuera mucho más engorrosa.

#### *Solución*

En la fase inicial del proyecto, el equipo llevó a cabo una evaluación detallada del código de los web scrapers, utilizando herramientas de depuración para identificar cuellos de botella. Posteriormente, se efectuó una revisión de la base de datos, considerando la adaptación y normalización de esta, como también la creación de índices, el uso de almacenamiento en caché y la optimización de consultas.

A pesar de estas mejoras, se concluyó que la capacidad de la infraestructura existente era insuficiente para manejar el volumen de datos. Como solución, se migró la base de datos de MySQL a Amazon Redshift, un servicio especializado para manejar grandes volúmenes de datos.

En la etapa final del proyecto, se hizo un cambio significativo en la implementación de los web scrapers. En lugar de operar dentro de un contenedor Docker, se utilizó una arquitectura serverless de Amazon llamada lambda, traduciéndose en un sistema más escalable y eficiente. Por otro lado, la asignación de tareas se realizó con el modelo maestro-subordinado. Con este cambio se logró reducir significativamente la interacción con la base de datos principal. Los cambios proporcionaron una reducción de costos asociados a los web scraper en aproximadamente en un 75%.

#### Caso 2

En el evento “DEMODEV Day 2023” de Platanus ventures Josefina Hidalgo, quien ejerce como Gerente de Ingeniería en Buk, realizó una presentación respecto a un interesante desafío.

#### *Problemática*

Actualmente, la startup chilena Buk sobresale gracias a su crecimiento en los últimos años, y su solución SaaS se ha convertido en una herramienta esencial para la gestión de personal en distintas empresas. En esta área hay un comportamiento interesante: a fin de cada mes el uso interno del software entre las empresas aumenta drásticamente, se realizan el procesamiento de nóminas, cierre de periodos contables y generación de reportes. Todo el procesamiento de todos estos eventos en un pequeño lapso genera que el sistema se ralentice significativamente, por lo que se tuvo que realizar una acción para poder mejorar la experiencia de usuario en las tareas de fin de mes.

## *Solución*

Se analizaron los diferentes aspectos que impactan en la realización de estos procesos y se identificó que el principal cuello de botella es la escritura y lectura en la base de datos. Se realizó sharding sobre la base de datos: se separó la base de datos en base a reglas que se definió internamente en la empresa. De esta forma, los eventos de escritura y lectura solo afectan únicamente a su shard y no a toda la base de datos. Gracias a esta solución la base de datos pudo soportar a fin de mes la alta demanda. Es importante mencionar que a medida que la demanda va creciendo, la cantidad de shards hay que ir adaptándolas dinámicamente, es decir, hay que evaluar cada ciertos periodos si es necesario volver a realizar separaciones significativas de la base de datos.

## Propuestas de solución

Dada la literatura y los casos de éxito observados, es relevante separar las propuestas de solución en dos aspectos: arquitectura y lógica interna de scrapers. En cada una de estas se presentarán diferentes soluciones. Es importante mencionar que la mayoría de las selecciones no son excluyentes, se pueden elegir múltiples.

### Lógica interna de scrapers

Si bien la arquitectura genera grandes cambios en la escalabilidad y rendimiento, realizar cambios internos en la lógica de código es crucial para poder asegurarse de que cambios estructurales son necesarios. Se asume que se utilizará herramientas de code profiling para este proceso.

- 1.1 Implementación robusta de “logging”.
- 1.2 Proxies: En caso de fallos en una página, se recurrirá a proxies para determinar si el problema es ajeno a un bloqueo por IP del scraper.
- 1.3 Challange-solvers: Al persistir errores aun con el uso de proxies, se verificará mediante solucionadores de desafíos como captchas, la ejecución de JavaScript y la espera de un tiempo predefinido.
- 1.4 Implementación de procesamiento paralelo de tareas: La capacidad de realizar web scraping a varios dominios paralelamente.

- 1.5 Ajustar parámetros de la herramienta Scrapy: Se está utilizando el framework “Scrapy” de Python, el cual posee una serie de parámetros que pueden afinar con el fin de minimizar el tiempo de ejecución. Se puede configurar la cantidad de páginas que se solicitan o procesan a la simultáneamente, la cantidad de hilos en funcionamiento y tiempo de espera de solicitudes entre otros parámetros.
- 1.6 Ejecución de consultas por lotes en la base de datos: En vez de guardar o solicitar un dato a la vez, realizarlo con múltiples.

## Arquitectura TI

Estos cambios tienen un impacto mayor, pero traen consigo una complejidad relevante al momento de considerar su implementación.

- 2.1 Database Sharding y/o partitioning: Separar la base de datos en subconjuntos según reglas definidas previamente.
- 2.2 Lambda functions: Ejecutar el código de los web scrapers a través de funciones lambda en Amazon AWS en vez de contenedores Docker.
- 2.3 REDIS para la asignación de tareas: Coordinador central, denominado "maestro", es responsable de asignar tareas específicas a nodos subordinados. En este caso el maestro utiliza una base de datos REDIS, la cual brinda velocidades de escritura-lectura muy altos. Los web scrapers son los subordinados que obtienen la información, la procesan y la devuelven al maestro para que la almacene.
- 2.4 Mejorar índices y consultas: Reducir el tiempo que los web scrapers utilizan en la comunicación con la base de datos.
- 2.5 Normalización de base de datos: Si bien ya está normalizada en varios aspectos, realizar una normalización con un foco en la comunicación con los web scrapers.
- 2.6 Migración a Amazon Redshift: Un servicio especializado para manejar grandes volúmenes de datos el cual reemplazaría la base de datos MySQL actual.

## Solución escogida

Para la selección de la solución se utilizará una escala numérica del 1 al 10 en los siguientes aspectos, cada uno con una ponderación independiente que se alinee con los objetivos del proyecto.

Criterio	Evaluación	Ponderación
Mantenibilidad	¿Qué tan fácil es mantener el sistema en el largo plazo? (1 muy difícil - 10 muy fácil)	20%
Costo	¿Cuál es el costo asociado a implementar tal solución? (1 muy costoso - 10 muy económico)	10%
Complejidad	¿Qué tan complejo es implementar la solución al sistema? (1 muy complejo - 10 muy simple)	20%
Rendimiento	¿Qué nivel de impacto tendrá en el tiempo de ejecución? (1 bajo impacto - 10 alto impacto)	25%
Escalabilidad	¿Qué tan bien puede el sistema manejar el crecimiento en la demanda? (1 no escalable - 10 altamente escalable)	25%

Tabla 2: Criterios, evaluaciones y ponderaciones de solución escogida.<sup>9</sup>

	Mantenibilidad	Costo	Complejidad	Rendimiento	Escalabilidad	2	Resultado
Solución	20%	10%	20%	25%	25%		
1.1	5	5	2	5	10	5,65	
1.2	9	1	9	5	10	7,45	
1.3	3	2	4	4	9	4,85	
1.4	7	2	3	8	8	6,2	
1.5	9	1	8	8	7	7,25	
1.6	5	7	3	7	6	5,55	
2.1	2	3	2	6	9	4,85	
2.2	5	3	3	5	7	4,9	
2.3	8	5	3	8	10	7,2	
2.4	3	2	8	6	7	5,65	
2.5	2	6	3	5	5	4,1	
2.6	5	1	1	5	9	4,8	

Tabla 3: Análisis soluciones<sup>10</sup>

Dados los resultados, se consideró aquellas soluciones que tengan un puntaje por sobre 5.

## Análisis de riesgo

Para un análisis robusto del riesgo, se utilizará una matriz de riesgo 5x5, donde el eje x representa el impacto de un evento y el eje y representa la probabilidad de un evento.

Probabilidad/Impacto	Insignificante	Menor	Moderado	Mayor	Catastrófico
----------------------	----------------	-------	----------	-------	--------------

<sup>9</sup> Elaboración propia.

<sup>10</sup> Elaboración propia.

Muy alta	5	10	15	20	25
Alta	4	8	12	16	20
Media	3	6	9	12	15
Baja	2	4	6	8	10
Muy baja	1	2	3	4	5

Tabla 4: Matriz riesgo 5x5.<sup>11</sup>

El resultado obtenido en cada cuadrante representa el producto entre el impacto y su probabilidad, el cual llamaremos nivel de riesgo.

En la siguiente tabla se pueden observar los diferentes eventos que presentan un riesgo significativo y su respectiva mitigación, la cuál será considerada al momento de realizar el proyecto.

---

<sup>11</sup> Elaboración propia.

Riesgo	Probabilidad	Impacto	Nivel Riesgo	Mitigación
Configuración errónea de recopilación de información de rendimiento de scrapers. Incosistencia en los resultados	4	3	12	Recopilación incluirá la versión de código y en que condiciones se encuentra. Evaluación semanal de resultados.
Excesiva dependencia de CloudWatch para Logging. Puede caerse el sistema o cambiar su política de precios.	3	2	6	implementación que no sea dependiente solo de CloudWatch, que tenga alta adaptabilidad frente a otras soluciones
Información sensible en logs	3	4	12	Antes de que se hagan cambios a producción, revisar correctamente que logs se están guardando. Investigar soluciones que realicen scan de los cambios y detecten posibles vulnerabilidades
Peor rendimiento en trabajo de lotes	4	4	16	Generar una comparativa de rendimiento con lotes. Evaluar si la base de datos tiene una presión significativa en estos eventos.
Fallo en cadena en el procesamiento en lotes, resultando en la falla completa del lote de trabajos.	5	5	25	Implementar funciones que reintenten en caso de errores con la comunicación con la base de datos. Cambiar querys gradualmente en vez de cambiarlas todas en una sola vez.
Falta de un claro plan de evaluación de cambios puede resultar en una evaluación incompleta	3	3	9	Generar un plan de evaluación detallado de lo que se espera y de lo que se debe revisar antes de proceder al siguiente paso.
Configuración de parámetros de scrapy no se adecuan al ambiente de producción	4	4	16	Solo guardar parámetros que tienen la misma configuración que producción, es decir, 2GB ram y 0,5 CPU. Para esto, utilizar docker para simular el ambiente.
Incorrecta asignación de tareas a través de Redis	2	4	8	Revisión de resultados esperados vs obtenidos. Si existe incertidumbre, reiterar.

Tabla 5: Nivel de riesgo para cada posible evento.<sup>12</sup>

## Metodología y plan de implementación

Dada la naturaleza del rubro de la informática se decidió trabajar con la metodología SCRUM con el uso constante de JIRA para la asignación de tareas. De esta forma el equipo estaba actualizado respecto mi avance, prioridades y necesidades.

---

<sup>12</sup> Elaboración propia. Sus mitigaciones directamente en el proyecto explicadas y elaboradas.

El plan de implementación tendrá dos fases, primero se implementarán los cambios asociados a la lógica interna de scrapers y luego se realizarán los cambios de arquitectura TI. De esta forma, se podrá entender correctamente cada aspecto de los scrapers antes de proceder a cambios más amplios.

1. Configuración de proyecto scrapers para la recopilación de información.

Para poder medir correctamente el impacto de todos los cambios realizados, es necesario recopilar toda la información posible respecto al rendimiento de los web scrapers. Afortunadamente la librería Scrapy posee un módulo que almacena diferentes métricas en el formato deseado. De esta forma, cada vez que se ejecuten los scrapers de forma local, se recopilará información automáticamente.

2. Implementación robusta de logging.

Es indispensable tener una visión completa de lo que sucede tanto en producción como en desarrollo local. Es por esto por lo que se utilizará la herramienta CloudWatch de Amazon, la cual almacena logs de forma secuencial y cronológica.

3. Implementación de trabajo en lotes y Proxies.

Se implementará la función de procesar páginas en lotes, lo que se traduce en procesamiento paralelo de tareas y ejecución de consultas por lotes en la base de datos.

4. Evaluación de cambios.

Antes de realizar cambios de gran impacto como lo son de infraestructura, se procederá a evaluar el rendimiento de los cambios ya realizados. De esta forma se tendrá completa seguridad que los cambios están totalmente preparados para la siguiente fase.

5. REDIS para la asignación de tareas y mejora de índices y consultas.

Se iniciará creando un ambiente local a través de Docker, donde se replicará el mismo ambiente que producción. Luego se crearán las instancias de REDIS, las cuales se encargarán de asignar las tareas a los subordinados.

6. Establecer los parámetros de la herramienta Scrapy.

Como se comentó anteriormente, Scrapy posee una serie de parámetros que se pueden recopilar. Con el gran volumen de información recopilado en el paso 1, se procederá a utilizar una regresión lineal para estimar los parámetros óptimos para los scrapers.

	Octubre				Noviembre			
	1	2	3	4	1	2	3	4
Configuración de proyecto scrapers para la recopilación de información								
Implementación robusta de logging								
Implementación de trabajo en lotes y Proxies								
Evaluación de cambios								
REDIS para la asignación de tareas y mejora de índices y consultas								
Establecer los parámetros de la herramienta Scrapy								

Ilustración 4: Planificación puesta en marcha proyecto desde el primer día de ejecución.<sup>13</sup>

## Desarrollo e Implementación

### Configuración de proyecto scrapers para la recopilación de información para regresión lineal

La información recopilada debe ser lo más cercana posible al ambiente de producción, para lograr esto, se utilizaron máquinas de Docker con variables que configuradas idénticas a producción: 0,5 GB RAM y 0,25 núcleos.

Esta instancia tendrá una funcionalidad especial: Al momento de inicializarse, tomará una configuración de parámetros aleatoria e irá guardando los resultados relevantes, como número de páginas obtenidas, información obtenida, número de errores y tiempo de ejecución.

Las variables independientes que cambiarán en cada ejecución son las siguientes:

Variable	Descripción	Tipo de número
Download delay	Mínimo de tiempo que se espera entre dos solicitudes.	Flotante
Download timeout	Tiempo máximo de espera por respuesta de solicitud. Si supera el tiempo, se asume que hay un problema con ese dominio.	Flotante
Concurrent requests	Máximo de solicitudes que pueden ocurrir de manera simultánea.	Entero
Concurrent requests per domain	Máximo de solicitudes simultáneas a un dominio.	Entero
Reactor Threadpool Maxsizes	Número máximo de hilos asíncronicos que pueden ocurrir simultáneamente	Entero

<sup>13</sup> Elaboración propia.

Tabla 6: Variables que cambiarán de manera aleatoria en cada ejecución de un meta scraper.<sup>14</sup>

Es importante mencionar que el web scraper trabajará con chunks de 20 ítems, es decir, recibe 20 dominios, los procesa y luego los almacena.

## Implementación robusta de logging

Dado que se está usando Amazon AWS se decidió utilizar las herramientas que ofrece para lo que es logging: Amazon CloudWatch. A través de esto, todos los elementos que se impriman en la consola serán almacenados en tiempo real. Para poder integrar correctamente este servicio, se integró la siguiente configuración en la tarea de meta\_scraper:

The screenshot shows the 'Log collection' configuration section. It includes a dropdown menu set to 'Amazon CloudWatch', a checked checkbox for 'Use log collection', and three key-value pairs: 'awslogs-group' (Value type: Value, Value: \*\*\*\*\*), 'awslogs-region' (Value type: Value, Value: \*\*\*\*\*), and 'awslogs-stream-prefix' (Value type: Value, Value: \*\*\*\*\*). There is also a button labeled 'Add log configuration option'.

Ilustración 5: Configuración logging para meta scraper.<sup>15</sup>

Para no almacenar una cantidad gigantesca de logs, se configuró el nivel de severidad de logs que se almacenarán de manera de solo mostrar las advertencias de error.

---

<sup>14</sup>Elaboración propia. Desarrollo completo de código en anexo 2.

<sup>15</sup> Elaboración propia.

```

def meta_scrape_domains(
    country_code: str,
    domain_ids_and_urls: List[Tuple[str, str]],
    debug=False
) -> None:
    log_file_location = f"scraped_data/logs/meta_scraped.log"
    crawler_settings = {
        # Configuración del meta scraper
    }

    if not debug:
        crawler_settings['LOG_LEVEL'] = 'WARNING'
    else:
        crawler_settings['LOG_FILE'] = log_file_location
        crawler_settings['DOWNLOADER_STATS'] = True

```

Ilustración 6: Configuración nivel de severidad de Logs en producción.<sup>16</sup>

Por último, se implementó logs para la totalidad de manejo de errores: Si ocurre un error, generamos un Log de advertencia y el código procede. De esta forma se tendrá constancia después de que ocurrió en el momento y continua sin interrumpir el funcionamiento.

```

try:
    addresses = self.return_address_extractor.extract_from_text(full_text)

    if len(addresses) > 0:
        return {"subdirectory": url_subdirectory, 'addresses': addresses}

except Exception as e:
    self.logger.error(f"Error in return address extraction for {response.url}: {e}")

return "None"

```

Ilustración 7: Ejemplo de código a manejo de error con Logs. Si el código en el bloque “try” tiene algún fallo, se procederá con el bloque ubicado en “except”, impidiendo que el código termine por un fallo.<sup>17</sup>

---

<sup>16</sup> Elaboración propia.

<sup>17</sup> Elaboración propia.

Una gran ventaja de la integración de Amazon Cloudwatch es las herramientas en la consola para encontrar errores específicos a través de comandos, por ejemplo, a través del siguiente comando se pueden obtener los errores ocurridos después de la fecha y hora 2023-11-18 00:00:00, en los meta scrapers con la palabra “call\_worker” incluida en ella.

```
(meta-scraper-py3.10) ➔ meta_scraper git:(mmu_branch) ✘ aws logs filter-log-events \
    --log-group-name "/ecs/meta-scraper" \
    --filter-pattern "call_worker" \
    --start-time 1700276400000
```

Ilustración 8: Ejemplo búsqueda de Logs a través de todos los meta scraper en ejecución y ejecutados.<sup>18</sup>

## Implementación de trabajo en lotes y Proxies

### Proxies

Para su correcta implementación se usaron los Middlewares de la librería Scrapy, estos interceptan solicitudes y respuestas de los servidores. (Spider Middleware — Scrapy 2.11.0 documentación, s. f.). En este caso, se creó la clase ProxyMiddleware<sup>19</sup>, el cuál posee la siguiente lógica:

1. uso\_de\_proxy\_por\_dominio: Lista la cual almacena los dominios que usarán proxy.
2. Intercepción de solicitudes: Si el dominio se encuentra en “uso\_de\_proxy\_por\_dominio”, usar proxy en la solicitud.
3. Intercepción de respuestas: Si la respuesta del servidor tiene un código de estado 403, el dominio se agrega a “uso\_de\_proxy\_por\_dominio” y se reintentará. Solo se realizará una vez por dominio.

### Implementación de trabajo por lotes

El desarrollo se puede separar en diferentes partes:

1. Consultas a la base de datos por lotes

Se adaptaron la gran mayoría de las lecturas y escrituras para que pudieran trabajar por lotes.

---

<sup>18</sup> Elaboración propia.

<sup>19</sup> Elaboración de clase en anexo 3. Implementación de trabajo en lotes y Proxies.

```

UPDATE meta_scrapes
SET
    {timestamp_column} = NOW()
WHERE
    `meta_scraping_id` = %(meta_scraping_id)s
    AND
    `domain_id` IN %(domain_ids)s

```

Ilustración 9: Código de escritura en chunks a la base de datos. Esta consulta en específico actualiza el tiempo de comienzo o término de un chunk.<sup>20</sup>

```

SELECT
    *
FROM
    meta_scraper_view
WHERE
    country_code = %(country_code)s AND
    meta_scraping_id = %(meta_scraping_id)s
ORDER BY LIMIT %(limit)s;

```

Ilustración 10: Código lectura de base de datos en chunks. Este código en específico solicita un grupo de dominios a procesar.<sup>21</sup>

## 2. Trabajo de tareas en paralelo con Scrapy

Se adaptó gran parte del código para trabajar con chunks de 20 tareas de forma paralela. Dado que se utiliza la Librería Scrapy, no fue difícil su implementación.

## 3. Uso de profiling para identificación de cuellos de botella.

Se utilizó la herramienta pyinstrument para poder realizar un análisis del código:

---

<sup>20</sup> Elaboración propia.

<sup>21</sup> Elaboración propia.

```

07.143 cmdmodules test_sitemap.py1
└ 56.601 meta_scrape_domains meta_scrapers/utils/processor_functions.py7
  └ 56.382 CrawlerProcess.start scrapy/crawler.py:391
    └ 56.382 SelectReactor.run twisted/internet/base.py:1316
      └ 56.382 SelectReactor.mainloop twisted/internet/base.py:1370
        └ 35.151 SelectReactor.runUntilCurrent twisted/internet/base.py:955
          └ 55.092 Cooperator._tick twisted/internet/task.py:674
            └ 35.092 CooperativeTask._onWorkUnit twisted/internet/task.py:519
              └ 35.088 <genexpr> scrapy/utils/defer.py:74
                └ 35.088 iter_ercallback scrapy/utils/defer.py:113
                  └ 35.082 MutableChain._next_ scrapy/utils/python.py:352
                    └ 35.082 SpiderMiddlewareManager._evaluate_iterable scrapy/core/spidermw.py:54
                      └ 35.082 OffsiteMiddlewareManager._process_spider_output scrapy/spidermiddlewares/offsite.py:28
                        └ 35.081 SpiderMiddlewareManager._evaluate_iterable scrapy/core/spidermw.py:54
                          └ 35.081 <genexpr> scrapy/spidermiddlewares/referer.py:342
                            └ 35.081 SpiderMiddlewareManager._evaluate_iterable scrapy/core/spidermw.py:54
                              └ 35.081 <genexpr> scrapy/spidermiddlewares/utlength.py:48
                                └ 35.081 SpiderMiddlewareManager._evaluate_iterable scrapy/core/spidermw.py:54
                                  └ 35.081 <genexpr> scrapy/spidermiddlewares/depth.py:58
                                    └ 35.081 SpiderMiddlewareManager._evaluate_iterable scrapy/core/spidermw.py:54
                                      └ 34.628 MetaSpider._parse meta_scrapers/scrapers/metaspider.py:112
                                        └ 34.628 MetaSpider.find_sitemap_info meta_scrapers/scrapers/metaspider.py:433
                                          └ 34.626 get requests/api.py:16
                                            └ 34.626 request requests/api.py:16
                                              └ 34.627 Session.request requests/sessions.py:457
                                                └ 34.625 Session.send requests/sessions.py:613
                                                  └ 28.955 HTTPAdapter.send requests/adapters.py:395
                                                    └ 28.935 HTTPSConnectionPool.urlopen urllib3/connectionpool.py:522
                                                      └ 28.928 HTTPSConnectionPool._make_request urllib3/connectionpool.py:361
                                                        └ 22.297 HTTPSConnection.getresponse http/client.py:1338
                                                          └ 22.296 HTTPResponse.begin http/client.py:311
                                                            └ 22.296 HTTPResponse._read_status http/client.py:278
                                                              └ 22.289 SocketIO.readinto socket.py:979
                                                                └ 22.288 SSLocket.recv_into ssl.py:1263
                                                                  └ 22.288 SSLocket.read ssl.py:1221
                                                                    └ 22.288 _SSLocket.read None
          └ 6.619 HTTPSConnectionPool._validate_conn urllib3/connectionpool.py:1034
            └ 6.619 HTTPSConnection.connect urllib3/connection.py:361
              └ 3.489 HTTPSConnection._new_conn urllib3/connection.py:161
                └ 3.489 create_connection urllib3/util/connection.py:37
                  └ 3.376 socket.connect None
                    └ 3.193 ssl_wrap_socket urllib3/util/ssl_.py:355
                      └ 3.059 _ssl_wrap_socket_imp urllib3/util/ssl_.py:481
                        └ 3.059 SSLSocket._wrap_socket ssl.py:507
                          └ 3.059 SSLSocket._create ssl.py:1014
                            └ 3.055 SSLSocket._do_handshake ssl.py:1335
                              └ 3.055 _SSLSocket._do_handshake None
          └ 5.040 Response.content requests/models.py:825
            └ 5.039 generate requests/models.py:756
              └ 5.039 HTTPResponse.stream urllib3/response.py:607
                └ 5.039 HTTPResponse.read_chunked urllib3/response.py:789
                  └ 5.036 HTTPResponse._handle_chunk urllib3/response.py:767
                    └ 5.036 HTTPResponse._safe_read http/client.py:623
                      └ 5.036 SocketIO.readinto socket.py:691
                        └ 5.036 SSLocket.recv_into ssl.py:1263
                          └ 5.036 SSLocket.read ssl.py:1121
                            └ 5.036 _SSLocket.read None
      └ 21.251 SelectReactor.doSelect twisted/internet/selectreactor.py:92
        └ 21.045 select None

```

Ilustración 11: Ejemplo de métricas de tiempo de ejecución a través de profiling de `meta_scaper` sin interacciones con la base de datos.<sup>22</sup>

Se puede observar que el tiempo de ejecución es de 57,143 segundos. Después de varias iteraciones de adaptación de código, se logró reducir el tiempo de ejecución significativamente a 31,367.

Los cambios más relevantes realizados son los siguientes:

Función	Descripción	Cambio	Impacto
find_sitemap_info	Obtiene productos y categorías de la página web	Sé implementó una función filtro para no repetir solicitudes que ya se realizaron.	Reducción de solicitudes
parse_page	Obtiene información de página y crea solicitudes para páginas siguientes para continuar.	Selección y filtro robusta para la selección de siguientes páginas para continuar.	Reducción de solicitudes
parse_html	Obtención de código html	En vez de ejecutarlo de forma independiente por función, pasar resultado como variable.	Reducción de ejecución redundante de funciones.

<sup>22</sup> Elaboración propia.

Tabla 7: Cambios con mayor impacto en el código interno del meta\_scraping.<sup>23</sup>

## Evaluación de cambios

Primero se procedió a subir los cambios a producción y evaluar los resultados obtenidos. Se comparó la información obtenida antes y después de los cambios. Por un lado, se evaluó el tiempo de ejecución, el cuál disminuyó significativamente:

```
select meta_scraping_id,
       count(*)                                              as count,
       AVG(TIMESTAMPDIFF(SECOND, meta_scraping_start, meta_scraping_end)) as avg_time
  from meta_scrapes
 where meta_scraping_id = 'OLD_META_SCRAPING_ID'
   or meta_scraping_id = 'NEW_META_SCRAPING_ID'
 group by meta_scraping_id;
```

Ilustración 12: Consulta a la base de datos para obtener el tiempo de ejecución promedio por ítem.<sup>24</sup>

meta_scraping_id	count	avg_time	Chunk size	Average per domain
OLD_META_SCRAPING_ID	223449	41,982	1	41,982
NEW_META_SCRAPING_ID	381714	345,3671	20	17,268355

Tabla 8: Resultados de consulta.<sup>25</sup>

Se puede observar que el tiempo promedio de ejecución por dominio disminuyó significativamente, de 41,982 segundos a 17,268 segundos.

Luego se evaluó a información obtenida, esta fue exactamente la misma en ambos casos<sup>26</sup>, con una diferencia significativa: En ciertos dominios se obtenía una respuesta 403 ya que el dominio detectó que quien realiza la solicitud es un bot. En la nueva versión, por el uso de proxies se pudo obtener la información completa de 4,87% de dominios que antes no se obtenían.

---

<sup>23</sup> Elaboración propia.

<sup>24</sup> Elaboración propia.

<sup>25</sup> Elaboración propia.

<sup>26</sup> Desarrollo en anexo 4. Evaluación de cambios.

meta_scraping_id	proxy_used	count	% of domains with new information
NEW_META_SCRAPING_ID		18477	378676 4,879%

Tabla 9: Resultados consulta de impacto de uso de proxies.<sup>27</sup>

REDIS para la asignación de tareas y mejora de índices y consultas.

Para el uso de REDIS se incorporaron diferentes componentes al meta scraper.

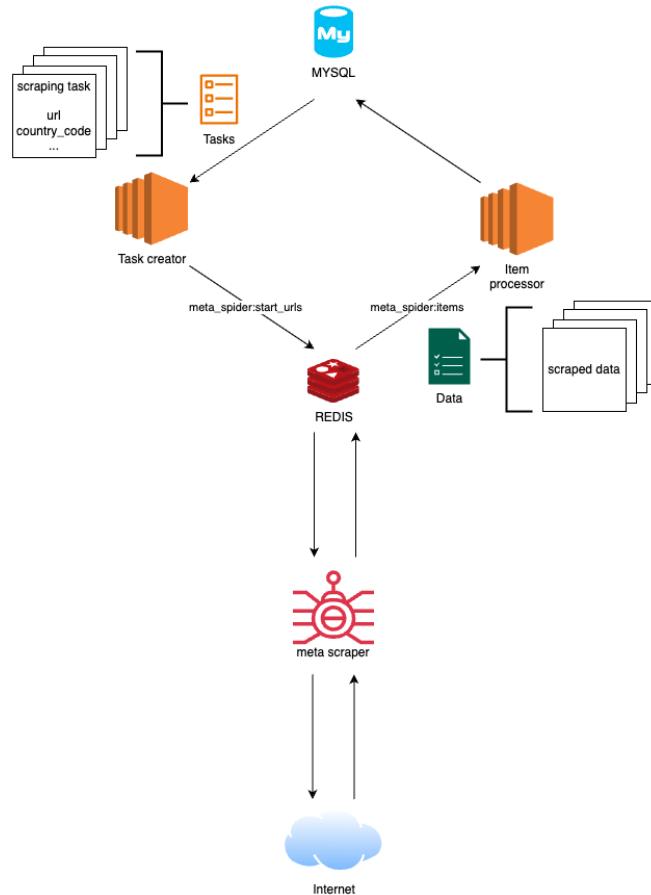


Ilustración 13: Asignación de tareas a través de REDIS.<sup>28</sup>

---

<sup>27</sup> Elaboración propia.

<sup>28</sup> Elaboración propia.

## 1. Máquina Docker REDIS.

A través de Docker se configuró una máquina REDIS, la cuál será un intermediario entre la base de datos y los scrapers.<sup>29</sup>

La máquina de REDIS posee dos llaves para almacenar la información:

1. “meta\_spider:start\_urls”: Tareas para realizar.
2. “**meta\_spider:items**”: Información de tareas completas.

## 2. Creador de tareas

Se preocupa de inicializarse, conectarse a la base de datos y crear las tareas.<sup>30</sup>

## 3. Meta scraper

El código posee la siguiente lógica<sup>31</sup>:

1. Si no hay dominios para trabajar, obtener más. Si no existen dominios, terminar.
2. Realizar web scraping a los dominios obtenidos.
3. Una vez terminado insertar ítems a REDIS.
4. Repetir.

## 4. Procesador de ítems

Los ítems almacenados en REDIS se almacenan en la base de datos, trabajando en chunks más grandes, reduciendo la cantidad de consultadas y tiempo de consulta promedio por ítem.<sup>32</sup>

---

<sup>29</sup> Código máquina Docker en anexo 5. REDIS para la asignación de tareas y mejora de índices y queries.

<sup>30</sup> Desarrollo en anexo 5. REDIS para la asignación de tareas y mejora de índices y queries.

<sup>31</sup> Desarrollo en anexo 5. REDIS para la asignación de tareas y mejora de índices y queries.

<sup>32</sup> Desarrollo en anexo 5. REDIS para la asignación de tareas y mejora de índices y queries.

## Establecer los parámetros de la herramienta Scrapy

Una vez recopilada toda la información a lo largo de todo el desarrollo del proyecto, se procedió a realizar una regresión lineal<sup>33</sup> con diferentes parámetros para así obtener una ecuación que estime el comportamiento de los web scrapers con el objetivo de minimizar el tiempo de ejecución por ítem, sujeto a un error de extracción máximo el cual también será estimado.

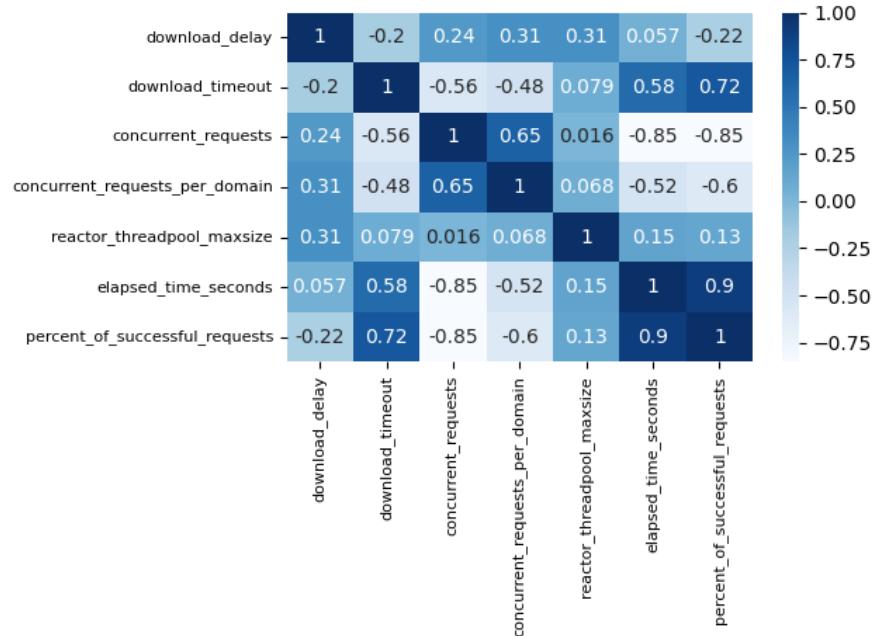


Ilustración 14: Correlaciones variables independientes.<sup>34</sup>

Se puede observar que gran parte de las variables menos download\_delay poseen una alta correlación con la variable tiempo de ejecución. Por otro lado, download timeout y elapsed time seconds poseen una alta correlación con el error del proceso de web scraping.

<sup>33</sup> Desarrollo de código en anexo 6. Establecer los parámetros de la herramienta Scrapy.

<sup>34</sup> Elaboración propia.

### Regresión lineal tiempo de ejecución

En la primera iteración se identificó que concurrent\_requests\_per\_domain y reactor\_threadpool\_maxsize no poseen una significancia para el modelo, por lo que se decidió eliminarlos. En la segunda iteración se obtuvo un modelo robusto: todas las variables son significativas (considerando un 5% de significancia), no existe multicolinealidad y el modelo es estadísticamente significativo. Por último, a través del modelo se explica un 84,9% de la variación de tiempo de ejecución.

### Regresión lineal error

Para poder dar un estimado al número de páginas que se pudo obtener respecto al total, se realizó una simple división entre el número de páginas total<sup>35</sup> contra el número de páginas obtenidas. De esta forma, se pudo calcular que proporción del total de páginas se obtuvo con la configuración propuesta.

Tras la primera regresión se decidió eliminar las variables download\_delay y concurrent\_requests\_per\_domain ya que no presentan significancia. En la segunda regresión se pueden obtuvieron resultados con una robustez parecida a la regresión lineal del tiempo de ejecución, pero con un mayor R-cuadrado de 93,41%.

Las ecuaciones resultantes son las siguientes:

$$\begin{aligned} \text{error} = & 0.8438 + \text{download timeout} * 0.0019 + \text{concurrent requests} * -0.0117 \\ & + \text{reactor threadpool maxsize} * 0.0034 \end{aligned}$$

$$\begin{aligned} y = & \text{download delay} * 14.53 + \text{download timeout} * 0.46 + \text{concurrent requests} * -7.2622 \\ & + 400.0861 \end{aligned}$$

Donde **y** representa el tiempo de ejecución y **error** representa el error máximo deseado.

Dados estos resultados se buscó el tiempo mínimo de ejecución sujeto al error, es decir:

---

<sup>35</sup> El número total de páginas fue estimado a través del máximo de páginas obtenido en todos los scrapers. Los dominios y el número de dominios fue exactamente el mismo para todas las combinaciones de parámetros.

$$\min_{y>0} y$$

Sujeto a:

$$error < (1 - error\ threshold)$$

De esta forma se procedió a escribir esta ecuación en código para encontrar el óptimo en cada caso de error mínimo.

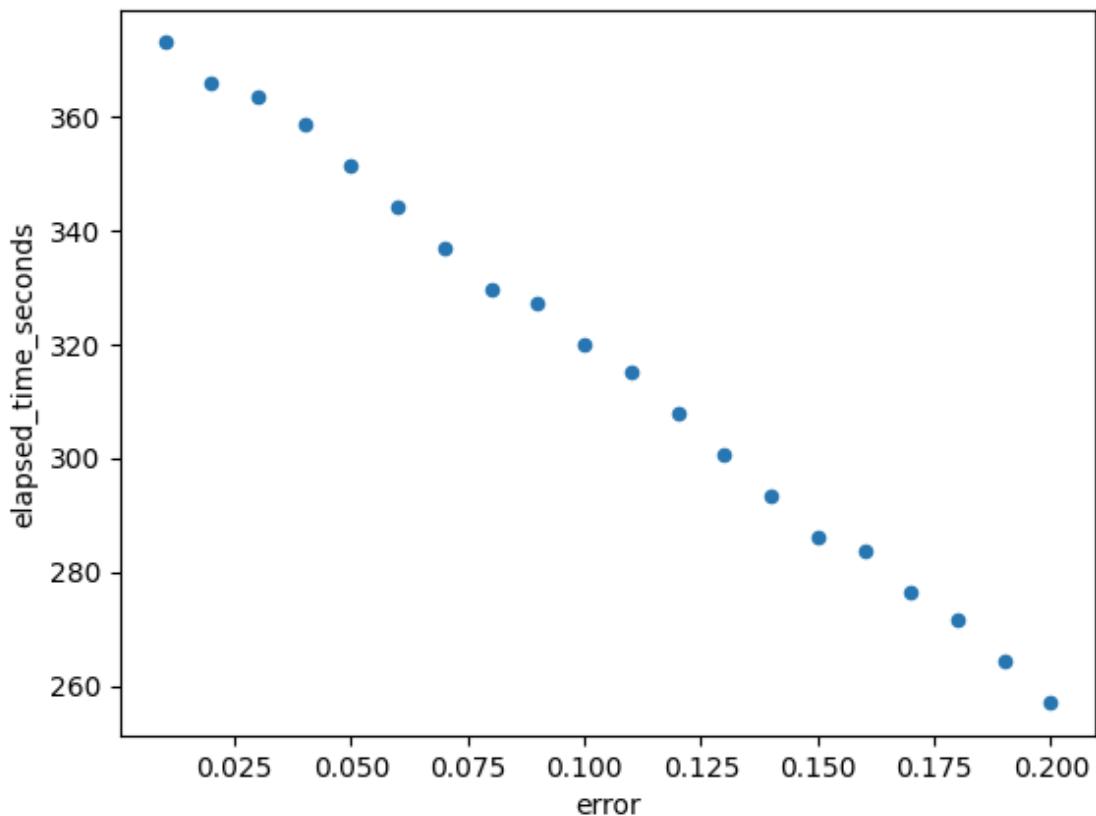


Ilustración 15: Relación entre error y tiempo de ejecución.<sup>36</sup>

<sup>36</sup> Elaboración propia.

Dados los resultados y los objetivos específicos, se busca un 5% de error, lo cual se traduce en 351.3499 segundos de ejecución, considerando que son 20 ítems por chunk, cada ítem toma 17,56 segundos en ser obtenido, con las siguientes variables:

Variable	Descripción
Download delay	1
Download timeout	115
Concurrent requests	16
Concurrent requests per domain	1
Reactor Threadpool Maxsizes	23

Tabla 10: Parámetros Scrapy que minimizan el tiempo de ejecución por ítem sujeto a un error máximo del 5%.<sup>37</sup>

## Resultados

Se realizó una comparativa, tiempo de ejecución al principio del proyecto, con implementación con chunks y finalmente con REDIS.

Es importante mencionar que REDIS no fue implementado en producción, el equipo de informática se encuentra actualmente migrando la infraestructura TI a Azure, en este proceso de migración se desea agregar REDIS a la solución. Es por esto mismo que los valores obtenidos se calcularon en el ambiente de desarrollo y se estimaron para producción.

---

<sup>37</sup> Elaboración propia.

	Non-chunk	Chunk	Chunk-REDIS
Chunk size obtain tasks	1	20	114084
Query to obtain tasks (seconds)	10,5	10,50	7
Chunk size scrape	1	20	20
Average time to complete (seconds)	41,982	345,36	345,36
Chunk save items	1	20	1000
Query to save items (seconds)	5,6	5,8	5,9
Average time per task (seconds)	58,082	18,083	17,27396136
Number of domains (Denmark)	114084	114084	114084
Total time (seconds)	6626226,9	2062981,0	1970682,6
<b>Total time (days)</b>	<b>76,7</b>	<b>23,9</b>	<b>22,8</b>

Tabla 11: Tiempo de ejecución de tareas dependiendo de la arquitectura del meta scraper.<sup>38</sup>

Para poder evaluar el impacto de la solución en la escalabilidad, se evaluó el tiempo de ejecución para diferentes tamaños de la base de datos:

Increase size by n times	Total time (days)		
	Non-chunk	Chunk	Chunk-REDIS
1	76,7	23,9	22,8
3	122,4	26,2	22,8
4	143,8	27,2	22,8
5	165,3	28,3	22,8
10	275,4	33,8	22,8
25	619,8	51,0	22,8

Tabla 12: Tiempo de ejecución del meta scraper en días dependiendo de cuantas veces se aumenta el tamaño de la base de datos.<sup>39</sup>

Se puede observar que la implementación de REDIS brinda una ventaja única: El impacto del tamaño de la base de datos no es significativo en el tiempo de ejecución.

De esta forma, los valores finales en los KPI son los siguientes:

<sup>38</sup> Elaboración propia.

<sup>39</sup> Elaboración propia.

KPI	Unidad	Valor Inicial	Valor esperado	Valor Final
Tiempo medio de extracción	Segundos	41,982	29,3874	17,268
Tasa de errores de extracción de dominio	%	5,87%	5%	5%
Tiempo de consulta a la base de datos por ítem en segundos	Segundos	16,1	10	0,0059614
Tiempo promedio de ejecución de las diez consultas más comunes	Segundos	10,3	5	10,3
Costo mensual de la base de datos	Unidades monetarias	19,841	15,8728	14,625
Tasa de consultas fallidas o abandonadas	%	4,63%	0%	0%
Porcentaje de Funciones con Manejo de Errores Integrado	%	85%	100%	100%
Porcentaje de Manejos de Errores con Registro almacenado	%	73,20%	100%	100%

Tabla 13: Valores finales KPI proyecto.<sup>40</sup>

## Evaluación económica

Para una correcta evaluación económica se realizó un flujo de caja en dólares centrado en los costos y ahorros asociados directamente con el proyecto, es decir, un enfoque de flujo de caja marginal.

Es importante que en esta evaluación consideraremos como proyecto la implementación completa de REDIS con el uso de chunks.

Se tomaron en cuenta los siguientes costos variables:

Variables	Descripción
AWS cloudwatch	Almacenamiento de Logs en tiempo real.
Ejecución REDIS	Ejecución de máquina REDIS para la asignación de tareas.
Ejecución máquinas ECS	Ejecución de scrapers.
Proxies	Costo de Proxies.

Tabla 14: Costos variables proyecto.<sup>41</sup>

Por otro lado, los costos fijos:

Fijos	Descripción
Salarios	Salarios de empleados asociados al proyecto.

Tabla 15: Costos fijos proyecto.<sup>42</sup>

Se consideraron los siguientes supuestos:

---

<sup>40</sup> Elaboración propia.

<sup>41</sup> Elaboración propia.

<sup>42</sup> Elaboración propia.

1. Cambio de moneda de coronas danesas a dólares fijo a través de todos los periodos. (1 DKK = 0,1415 USD)<sup>43</sup>.
2. En cada mes se extraerá la información de cada dominio 3,2<sup>44</sup> veces.
3. La relación del costo de la base de datos respecto al tiempo de ejecución de consultas es lineal. Esta se approximó a través de la relación histórica de dominios y costo base de datos.
4. La base de datos mantendrá la misma configuración a lo largo del proyecto.
5. El sueldo promedio de un desarrollador de software con 3 años de experiencia es de 6483,87 dólares antes de impuestos. Se estimó un mes de trabajo como la *inversión* del proyecto.
6. Dado que no se solicitó un crédito, se utilizó una tasa de descuento de 10% la cual se estimó según los parámetros financieros de la empresa.
7. Los costos por uso de los servicios ofrecidos por Amazon AWS se mantendrán a lo largo del proyecto.

Por último para poder evaluar correctamente el crecimiento de la empresa, se consideró la siguiente expectativa de dominios, dando una comparativa del aumento de volumen respecto a agregar un país como nuevo cliente.

Año	Dominios totales	Países como referencia de aumento de volumen
1	555186	
2	2310486	Alemania y España
3	4817586	Inglaterra

Tabla 16: Crecimiento de mercados y dominios totales a través del proyecto considerando el supuesto de relación lineal entre tamaño mercado y cantidad de dominios.<sup>45</sup>

<sup>43</sup> Extraído de Google. (2023). Tasa de cambio de la Corona Danesa (DKK) a Dólar Estadounidense (USD) el 1 de noviembre de 2023. Google Finance. Recuperado el 1 de noviembre de 2023, de <https://g.co/finance/DKK-USD?window=6M>

<sup>44</sup> Valor obtenido a través del promedio de veces que se obtiene cada dominio por mes.

<sup>45</sup> Elaboración propia.

Dado este crecimiento se realizó el flujo de caja con y sin proyecto:

Año	1	2	3
Costos fijos			
Costos variables	-\$ 7.775	-\$ 46.068	-\$ 156.627
<b>Utilidad antes de impuestos</b>	<b>-\$ 7.775</b>	<b>-\$ 46.068</b>	<b>-\$ 156.627</b>
Impuestos(-)	-\$ 1.555	-\$ 9.214	-\$ 31.325
<b>Utilidad después de impuestos</b>	<b>-\$ 9.330</b>	<b>-\$ 55.282</b>	<b>-\$ 187.953</b>
<b>FLUJO OPERACIONAL</b>	<b>-\$ 9.330</b>	<b>-\$ 55.282</b>	<b>-\$ 187.953</b>

Tabla 17: Flujo de caja privado sin proyecto.<sup>46</sup>

Año	0	1	2	3
Costos fijos	-\$ 4.319			
Costos variables		-\$ 1.936	-\$ 5.809	-\$ 11.618
<b>Utilidad antes de impuestos</b>	<b>-\$ 4.319</b>	<b>-\$ 1.936</b>	<b>-\$ 5.809</b>	<b>-\$ 11.618</b>
Impuestos(-)	-\$ 2.176	-\$ 387	-\$ 1.162	-\$ 2.324
<b>Utilidad después de impuestos</b>	<b>-\$ 6.495</b>	<b>-\$ 2.323</b>	<b>-\$ 6.970</b>	<b>-\$ 13.941</b>
<b>FLUJO OPERACIONAL</b>	<b>-\$ 6.495</b>	<b>-\$ 2.323</b>	<b>-\$ 6.970</b>	<b>-\$ 13.941</b>

Tabla 18: Flujo de caja privado con proyecto.<sup>47</sup>

	0	1	2	3
Flujo de caja privado (sin proyecto)	\$ - -\$	10.466	-\$ 66.020	-\$ 230.218
Flujo de caja privado (con proyecto)	-\$ 6.495	-\$ 2.324	-\$ 6.972	-\$ 13.944
Delta	-\$ 6.495	\$ 8.142	\$ 59.049	\$ 216.274

Tabla 19: Delta flujo de caja situación con proyecto y sin proyecto.<sup>48</sup>

<b>TIR</b>	328%
<b>VAN</b>	\$ 170.539

Tabla 20: VAN y TIR resultante del proyecto.<sup>49</sup>

<sup>46</sup> Elaboración propia.

<sup>47</sup> Elaboración propia.

<sup>48</sup> Elaboración propia.

<sup>49</sup> Elaboración propia.

Observando la TIR y VAN, se puede concluir que el proyecto es económicamente rentable.

Es importante mencionar que su rentabilidad esta sujeta al supuesto de que habrá crecimiento de dominios a realizar web scraping en los años posteriores. Si se mantiene la misma cantidad de dominios en el mismo periodo se obtienen resultados interesantes: El proyecto sigue siendo rentable, pero disminuye considerablemente su rentabilidad.

TIR	93%
VAN	\$ 8.265

Tabla 21: VAN y TIR resultante del proyecto si se mantiene la misma cantidad de dominios.<sup>50</sup>

De esta forma, se puede concluir que existe una relación positiva entre la rentabilidad del proyecto y la cantidad de dominios.

Como la estimación del impacto del proyecto en los costos asociados a la base de datos fueron aproximados utilizando datos históricos, se realizó un análisis de sensibilidad respecto a este.

Costo de base de datos respecto al tiempo de ejecución de consultas	TIR	VAN
Reducción 5%	324%	\$ 166.373
Caso base	328%	\$ 170.539
Aumento 5%	332%	\$ 174.705

Tabla 22: Análisis sensibilidad de costos lineales asociados al tiempo de ejecución de consultas a la base de datos.<sup>51</sup>

Se puede observar que el proyecto posee una sensibilidad leve a esta variable, esto ocurre porque al implementar REDIS el uso de la base de datos se reduce significativamente, reduciendo su impacto al variar el costo asociado a este.

---

<sup>50</sup> Elaboración propia.

<sup>51</sup> Elaboración propia.

## Conclusiones

La adaptación de la Arquitectura TI del ***meta\_scaper*** brinda una solución robusta para Tembi, se puede observar que la relación entre el aumento de dominios a obtener información y el costo asociado es cercana a una relación lineal de coeficiente 1.0, brindando mayor seguridad en su escalabilidad y a la vez asegurando visibilidad y seguridad en caso de errores.

Observando las proyecciones iniciales respecto a los resultados se puede concluir que los cambios realizados tuvieron un impacto mucho mayor al esperado, lo cual se puede analizar en dos aspectos:

1. La deuda técnica en el código resultó ser mayor al esperado, presentando varias inefficiencias tanto en tiempo de ejecución como en su estructura. Se logró identificar que las solicitudes a páginas son el proceso que consume más tiempo.
2. Se cometió el error de asumir desde un principio que la optimización de la base de datos es la única forma de minimizar el tiempo de ejecución, a través de REDIS se logró descentralizar el papel predominante de la base de datos en el proceso reduciendo significativamente su uso.

Por otro lado, no se logró reducir el tiempo promedio por consulta a la base de datos, ya que se identificó que es más eficiente enfocarse en reducir el tiempo de consulta a la de base de datos por ítem que en reducir la consulta en su totalidad, por lo que se priorizó trabajar en grupos.

## Recomendaciones

1. Implementar la misma infraestructura maestro-subordinado en el resto de los web scrapers.  
Evaluar la posibilidad de usar una instancia REDIS para la asignación de todas las tareas.
2. Creación de alarmas para los ***meta\_scaper***, de manera que si ocurre un error se pueda actuar rápidamente.
3. En el proceso de migración investigar soluciones que se adapten a la infraestructura actual, el uso de Kubernetes puede ser esencial para esta tarea.
4. Dada la alta complejidad de la actual infraestructura, educar a los profesionales de todas las áreas de la empresa para que tengan un entendimiento general de su funcionamiento.

5. Reducir la deuda técnica del código a través de roles de control de calidad y DevOps para una mejora paulatina de los repositorios.

## Bibliografía

- Botros, S., & Tinley, J. (2022). High Performance MySQL, 4th Edition.
- Europe: e-commerce market size, by Country 2021 / Statista.* (2023, 30 agosto). Statista. <https://www.statista.com/statistics/1113005/market-size-of-e-commerce-in-europe-by-country/>
- Gamma, E., Helm, R., Johnson, R. E., & Vlissides, J. (1994). *Design patterns: elements of reusable Object-Oriented software.* <http://www.ulb.tu-darmstadt.de/tocs/59840579.pdf>
- Han, X., & Zheng, L. (2020). Design and implementation of firmware data acquisition system based on SCRAPY Framework. *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS).* <https://doi.org/10.1109/icpics50287.2020.9202251>
- Hunt, A. P., & Thomas, D. A. (1999). *The Pragmatic Programmer: from journeyman to master.* <http://ci.nii.ac.jp/ncid/BA51839218>
- Martin, R. (2008). *Clean Code: a handbook of agile software craftsmanship.* <http://ci.nii.ac.jp/ncid/BA87924669>
- Prusty, A., Mejia, O., Shah, A., Kancherlapalli, P., Suresh, A., & Schiebel, R. (2020). Horizontally Scalable Web Crawler using Containerization and a Graphical user Interface. *International Journal of Engineering Research & Technology, 9(05),* 22780181.
- Ying, Z., Zhang, F., & Fan, Q. (2018). Consistent Hashing Algorithm Based on Slice in Improving Scrapy-REDIS Distributed Crawler Efficiency. En *2018 IEEE International Conference on Computer and Communication Engineering Technology.* IEEE. <https://doi.org/10.1109/CCET.2018.8542217>

## Anexos

### 1. Desarrollo tiempo de ejecución de consultadas en función del tamaño de la base de datos

Las consultas que se evaluaron fueron las siguientes:

1. Lectura de una fila: Meta scraper, obtiene los siguientes n dominios a realizar web scraping, filtrado por país e id de tarea.
2. Escritura de una fila: Inserta los datos de obtenidos por meta\_scraping.

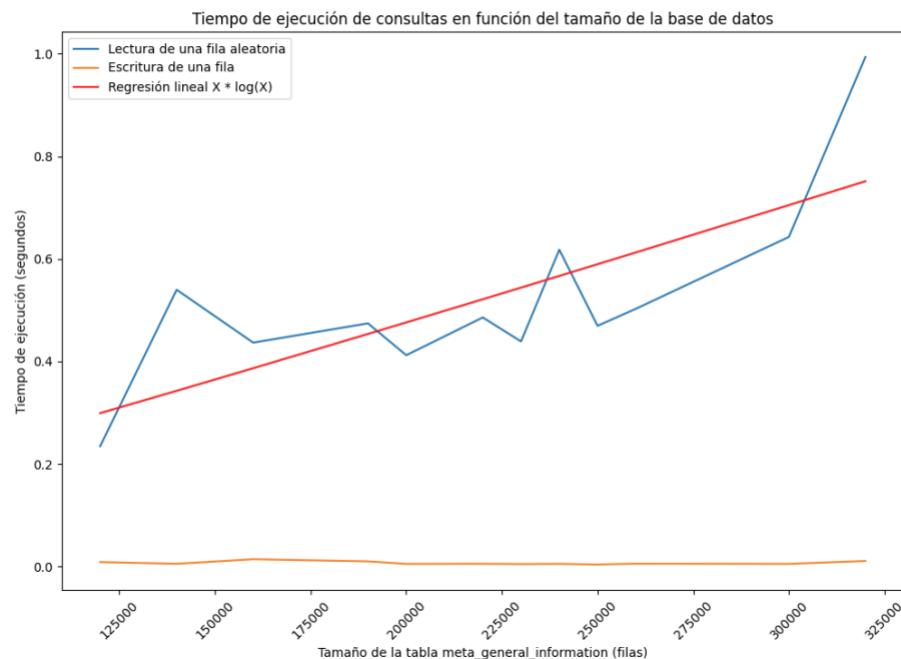


Gráfico 4: Tiempo de ejecución de consultas en función del tamaño de la base de datos.

Para entender en profundidad la relación, se realizó una regresión lineal considerando una ecuación logarítmica<sup>52</sup>, obteniendo la siguiente ecuación:

$$y = 1.70405488 \times 10^{-7} * (N \log N) + 0.0600846$$

a. Creación de base de datos local a través de Docker:

```
version: '3'

services:

db:
    image: arm64v8/mysql:8.0.29
    ports:
        - "${MYSQL_PORT}:${MYSQL_PORT}"
    volumes:
        - scrapers-mysql-data-test:/var/lib/mysql:rw
        - ./mysql/conf.d:/etc/mysql/conf.d:ro
        - ./mysql/my.cnf:/etc/mysql/my.cnf:ro
        - ./sql-scripts:/docker-entrypoint-initdb.d:ro
    environment:
        MYSQL_USER: "${MYSQL_USER}"
        MYSQL_PASSWORD: "${MYSQL_PASS}"
        MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
        MYSQL_PORT: "${MYSQL_PORT}"
    networks:
        shared:
            aliases:
                - db
```

b. Creación de datos:

```

headers = ['amount_of_meta_scrapes', 'query_execution_time', 'query_kind']
metrics_results = pd.read_csv(filepath_or_buffer= "metrics_database_size.csv", header=0)

active_connection.close()
engine.dispose()

for amount_of_desired_data in amount_of_desired_data:

    while amount_of_meta_scrapes < amount_of_desired_data:
        engine = get_db_engine(**get_db_credentials(MYSQL_DB=SCRAPERS_SCHEMA, option="SELECT"))
        active_connection = engine.connect()
        size_to_insert = min(amount_of_desired_data - amount_of_meta_scrapes, insert_max_chunk_size)
        df_to_insert = df.sample(size_to_insert)
        meta_scraping_id = create_meta_scraping_id(df_to_insert)
        df["meta_scraping_id"] = meta_scraping_id
        df_to_insert.to_sql(name="meta_general_information",
                            con=active_connection,
                            if_exists="append",
                            chunksize=1000,
                            method="multi",
                            index=False)
        amount_of_meta_scrapes = pd.read_sql(AMOUNT_OF_META_GENERAL_INFO, active_connection).iloc[0, 0]
        active_connection.close()
        engine.dispose()

    engine = get_db_engine(**get_db_credentials(MYSQL_DB=SCRAPERS_SCHEMA, option="SELECT"))
    active_connection = engine.connect()

    number_of_runs = 10
    for i in range(number_of_runs):
        print(f"Run {i}")
        query_execution_time = profile_query_read(QUERY_RAND)
        metrics_results = metrics_results.append(
            {
                'amount_of_meta_scrapes': amount_of_desired_data,
                'query_execution_time': query_execution_time,
                'query_kind': 'read_random'
            }, ignore_index=True
        )
    metrics_results.to_csv("metrics_database_size.csv", index=False)

    for i in range(number_of_runs):
        print(f"Run {i}")
        one_item = df.sample(1)
        meta_scraping_id = create_meta_scraping_id(one_item)
        one_item["meta_scraping_id"] = meta_scraping_id
        query_execution_time = profile_query_write_with_pandas_df(one_item)
        metrics_results = metrics_results.append(
            {
                'amount_of_meta_scrapes': amount_of_desired_data,
                'query_execution_time': query_execution_time,
                'query_kind': 'write_one'
            }, ignore_index=True
        )
    metrics_results.to_csv("metrics_database_size.csv", index=False)

    active_connection.close()
    engine.dispose()

```

### c. Desarrollo de regresión logarítmica:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

df1 = pd.read_csv('metrics_database_size.csv')
df2 = pd.read_csv('metrics_database_size_c.csv')

df = df1.append(df2)

df_grouped = df.groupby(['amount_of_meta_scrapes', 'query_kind']).mean()

df = df_grouped.reset_index()
df = df.pivot(index='amount_of_meta_scrapes', columns='query_kind', values='query_execution_time')
df.plot(label=['read_random', 'write_one'])

df_grouped = df_grouped.reset_index()
df = df_grouped[df_grouped['query_kind'] == 'read_random']
X = np.array(df['amount_of_meta_scrapes']).reshape(-1, 1)
y = np.array(df['query_execution_time']).reshape(-1, 1)

X_transformed = X * np.log(X)

linear_reg = LinearRegression()
linear_reg.fit(X_transformed.reshape(-1, 1), y)
y_pred = linear_reg.predict(X_transformed.reshape(-1, 1))

plt.xticks(rotation=45)
plt.title('Tiempo de ejecución de consultas en función del tamaño de la base de datos')
plt.xlabel('Tamaño de la tabla meta_general_information (filas)')
plt.ylabel('Tiempo de ejecución (segundos)')
plt.plot(*args: X, y_pred, color='red', label='Regresión lineal X * log(X)')

handles, labels = plt.gca().get_legend_handles_labels()
labels[0] = 'Lectura de una fila aleatoria' # Change as per your data
labels[1] = 'Escritura de una fila' # Change as per your data
labels.append('Regresión polinomial')

plt.legend(*args: handles, labels)

plt.gcf().set_size_inches(w: 12, h: 8)

print('O(N log N) regression')
print(f'y = {linear_reg.coef_[0]} * (N log N) + {linear_reg.intercept_}')
print('r2 score: {}'.format(linear_reg.score(X_transformed.reshape(-1, 1), y))) # x = 13,422,064

N_predict = 13422064
N_predict_transformed = N_predict * np.log(N_predict)

predicted_time = linear_reg.predict([[N_predict_transformed]])[0]

print(f'Estimated time for {N_predict} rows: {predicted_time} seconds')
plt.show()
```

d. Resultados regresión logarítmica.

```
O(N log N) regression
y = [1.70405488e-07] * (N log N) + [0.06008467]
r2 score: 0.5747402027629962
Estimated time for 13422064 rows: [37.59844097] seconds
```

e. ¿Por qué se realizó una regresión logarítmica y no lineal?

Si se evalúa la complejidad de la consulta de lectura, esta se separa en dos partes:

```
SELECT
    *
FROM
    meta_scraping_view
WHERE
    country_code = %(country_code)s AND
        meta_scraping_id = %(meta_scraping_id)s
ORDER BY RAND() LIMIT %(limit)s;
```

1. Se revisa toda la tabla “*meta\_scraping\_view*”, lo cual se traduce en una operación O(N).
2. Rand() genera un número aleatorio para cada fila y luego ordena la tabla en base a esto.  
Esto tiene una complejidad de O(N log N).

Si se evalúa la consulta en su totalidad, se obtiene una complejidad de O(N log N) cuando N es un número grande.

f. Supuesto: Relación lineal entre potencia de máquina de base de datos y ecuación obtenida.

Las métricas obtenidas y en consecuencia la ecuación obtenida fue evaluadas localmente, es decir, existe una diferencia en la RAM, procesadores y tipo de almacenamiento de la base de datos. Dado que no se puede simular exactamente el mismo ambiente que producción, se calculará un estimado de la ecuación usando una relación lineal.

El tiempo de ejecución es de 13,13 segundos con un tamaño de 18.574.853 filas.

En nuestra ecuación el valor es: 53.037 segundos.

La relación obtenida es:  $53.037 / 13,13 = 3,75350318$

Es decir, la ecuación corregida es:

$$y = \frac{1.70405488 \times 10^{-7} * (N \log N) + 0.06008467}{3,75350318}$$

g. Desarrollo gráfico casos hipotéticos tamaño base de datos vs tiempo de ejecución.

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np

coefficient = 1.70405488e-07
intercept = 0.06008467
division = 3.75350318

N_values = np.linspace( start: 10000000, stop: 100000000, num: 1000)

y_values = (coefficient * (N_values * np.log(N_values)) + intercept) / division

plt.figure(figsize=(12, 7))
plt.plot(*args: N_values, y_values, label='y(N)', color='dodgerblue', linewidth=2)

plt.title(label: 'Gráfico de y = (1.70405488e-07 * (N * log(N)) + 0.06008467) / 3.75350318', fontsize=14)
plt.xlabel(xlabel: 'N', fontsize=12)
plt.ylabel(ylabel: 'y', fontsize=12)
plt.grid(visible: True, which='both', linestyle='--', linewidth=0.5)
plt.xscale('log')
plt.yscale('linear')

plt.gca().xaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: '{:.0f}'.format(x)))

plt.legend()
plt.show()
```

## 2. Configuración de proyecto scrapers para la recopilación de información

```
(meta-scraper-py3.10) + meta_scraper git:(mmu_branch) ✘ docker build -t scraper_metrics:latest .
[+] Building 33.6s (14/14) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 977B
=> [internal] load .dockerrcignore
=> => transferring context: 44B
=> [internal] load metadata for docker.io/library/python:3.10-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/8] FROM docker.io/library/python:3.10-slim@sha256:dc6e9c287b31327c98228c37c4d93941084ca4199a0e6ac8d4fda46dc6ae1473
=> [internal] load build context
=> => transferring context: 1.08MB
=> CACHED [2/8] RUN apt-get update && apt-get install build-essential -y
=> CACHED [3/8] RUN python3 -m pip install --upgrade pip
=> CACHED [4/8] RUN python3 -m pip install "poetry== 1.4.2"
=> CACHED [5/8] RUN poetry config virtualenvs.create false
=> CACHED [6/8] WORKDIR /app
=> [7/8] COPY . /app/
=> [8/8] RUN poetry install $(test "$YOUR_ENV" == production && echo "--no-dev") --no-interaction --no-ansi
=> exporting to image
=> => exporting layers
=> => writing image sha256:2b9be6e715be1f53fec632875bead539f506fb6eb79211ec83f9af020798f7f0
=> => naming to docker.io/library/scrapers_metrics:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
(meta-scraper-py3.10) + meta_scraper git:(mmu_branch) ✘ docker run -m 512m --cpus=0.25 -d -v "$(pwd)":/app scraper_metrics:latest
535d58839c6e95ef7668bf410982742d7a4974fdc1723bd2b8c88147d82e922f
(meta-scraper-py3.10) + meta_scraper git:(mmu_branch) ✘ []
```

Ilustración 16: Creación e inicialización de máquinas `meta_scraper` en Docker con parámetros idénticos a producción.<sup>53</sup>

---

<sup>53</sup> Elaboración propia.

```
download_delays = [1.5, 3, 5]
download_timeouts = [60, 120]
concurrent_requests = [8, 16]
concurrent_requests_per_domain = [1, 2]
reactor_threadpool_maxsizes = [12, 24]

combinations = list(itertools.product(
    download_delays,
    download_timeouts,
    concurrent_requests,
    concurrent_requests_per_domain,
    reactor_threadpool_maxsizes
))

df = pd.DataFrame(combinations, columns=[
    'download_delay',
    'download_timeout',
    'concurrent_requests',
    'concurrent_requests_per_domain',
    'reactor_threadpool_maxsize'
])
df['done'] = False

file_path = 'scrapy_settings_combinations.csv'

if os.path.isfile(file_path):
    print("File exists")
    exit()

df.to_csv(file_path, index=False)
```

Ilustración 17: Generación de combinaciones posibles para los parámetros a analizar.<sup>54</sup>

```
df = pd.read_csv(filepath_or_buffer= "metrics/domains.csv", header=0, names=["domain_id", "url"])

df = df.groupby(df.index // 20)

combinations = pd.read_csv("scrapy_settings_combinations.csv")

combinations = combinations[combinations["done"] == False]

country_code = "DK"

combinations = combinations.iloc[::-1]

for index, row in combinations.iterrows():
    download_delay = row["download_delay"]
    download_timeout = row["download_timeout"]
    concurrent_requests = row["concurrent_requests"]
    concurrent_requests_per_domain = row["concurrent_requests_per_domain"]
    reactor_threadpool_maxsize = row["reactor_threadpool_maxsize"]
    print(f'Combination {index}')

    for i, group in df:
        print(f"Group {i}")
        print(len(group))
        call_worker(group, country_code, download_delay, download_timeout,
                   concurrent_requests, concurrent_requests_per_domain, reactor_threadpool_maxsize, i)

combinations.loc[index, "done"] = True
combinations.to_csv("scrapy_settings_combinations.csv", index=False)
```

Ilustración 18: Código de inicialización de Docker.<sup>55</sup>

---

<sup>54</sup> Elaboración propia.

<sup>55</sup> Elaboración propia.

```

df = pd.read_csv( filepath_or_buffer: "metrics/domains.csv", header=0, names=["domain_id", "url"])

df = df.groupby(df.index // 20)

country_code = "DK"
while True:
    download_delay = [0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9][random.randint( a: 0, b: 9)]
    download_timeout = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50][random.randint( a: 0, b: 9)]
    concurrent_requests = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50][random.randint( a: 0, b: 9)]
    concurrent_requests_per_domain = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10][random.randint( a: 0, b: 9)]
    reactor_threadpool_maxsize = [1, 5, 10, 15, 20, 25, 30, 35, 40, 45][random.randint( a: 0, b: 9)]

    for i, group in df:
        print(f"Group {i}")
        print(len(group))
        call_worker(group, country_code, download_delay, download_timeout,
                    concurrent_requests, concurrent_requests_per_domain, reactor_threadpool_maxsize, i)

```

Ilustración 19: Código de inicialización de Docker con parámetros aleatorios.<sup>56</sup>

```

def close(spider, reason):
    stats = spider.crawler.stats.get_stats()
    stats['total_domains'] = len(spider.domain_ids_and_urls)
    stats['total_urls'] = len(spider.urls)
    stats['country_code'] = spider.country_code
    settings = spider.crawler.settings
    stats['download_delay'] = settings.get('DOWNLOAD_DELAY')
    stats['concurrent_requests_per_domain'] = settings.get('CONCURRENT_REQUESTS_PER_DOMAIN')
    stats['concurrent_requests'] = settings.get('CONCURRENT_REQUESTS')
    stats['reactor_threadpool_maxsize'] = settings.get('REACTOR_THREADPOOL_MAXSIZE')
    stats['depth_priority'] = settings.get('DEPTH_PRIORITY')
    stats['download_timeout'] = settings.get('DOWNLOAD_TIMEOUT')
    stats['retry_enabled'] = settings.get('RETRY_ENABLED')
    singleton_data: FoundAndNotFoundResponses = singleton_storage_client.get_and_remove_many_keys([domain_id for domain_id, _ in spider.domain_ids_and_urls])
    stats['found'] = len(singleton_data['response'])
    stats['not_found'] = len(singleton_data['failed_data'])
    stats['not_found_domains'] = len(singleton_data['not_found_keys'])
    stats['group_number'] = spider.group_number

    filename = 'stats_v2.csv'
    if os.path.isfile(filename):
        df = pd.read_csv(filename, index_col=0)
    else:
        df = pd.DataFrame()
    df = df.append(stats, ignore_index=True)
    df.to_csv(filename)

```

---

<sup>56</sup> Elaboración propia.

*Ilustración 20: Almacenamiento de resultados de configuración de parámetros en Scrapy.*<sup>57</sup>

### 3. Implementación robusta de logging

```
def meta_scrape_domains(
    country_code: str,
    domain_ids_and_urls: List[Tuple[str, str]],
    debug=False
) -> None:
    log_file_location = f"scraped_data/logs/meta_scraped.log"
    crawler_settings = {
        # Configuración del meta scraper
    }

    if not debug:
        crawler_settings['LOG_LEVEL'] = 'WARNING'
    else:
        crawler_settings['LOG_FILE'] = log_file_location
        crawler_settings['DOWNLOADER_STATS'] = True
```

*Ilustración 21: Configuración nivel de severidad de Logs en producción.*<sup>58</sup>

---

<sup>57</sup> Elaboración propia.

<sup>58</sup> Elaboración propia.

### 3. Implementación de trabajo en lotes y Proxies

```
class ProxyMiddleware:
    retry_http_codes = {403}

    @classmethod
    def from_crawler(cls, crawler):
        return cls(crawler.settings, crawler.spider.country_code)

    def __init__(self, settings, country_code):
        proxy_logger.info('USING PROXY MIDDLEWARE')
        dotenv.load_dotenv()
        proxy_username = os.environ["PROXY_USERNAME"]
        proxy_pass = os.environ["PROXY_PASS"]

        if not proxy_username or not proxy_pass:
            raise Exception("Proxy username or password not found in the environment variables")

        self.proxy_server = "SOMETHING SECRET!"
        self.use_proxy_by_domain = set()

    def process_response(self, request, response, spider):
        # never used proxies, we add them.
        if response.status in self.retry_http_codes and not self.is_domain_in_proxies(request) and not request.meta.get('proxy'):
            proxy_logger.info(f"adding {request.cb_kwargs.get('domain_id')} to proxy list")

            self.use_proxy_by_domain.add(request.cb_kwargs.get('domain_id'))
            return request

        # we are using proxies. If it fails, we will retry up to one more time.
        if response.status in self.retry_http_codes and self.is_domain_in_proxies(request):
            if request.meta.get('retry_proxy_times', 0) < 1:
                proxy_logger.info(f"retrying {request.url}")
                return request

            # we have retried once, we will not retry again.
            proxy_logger.error(f"not retrying proxies with {request.url}, domain_id: {request.cb_kwargs.get('domain_id')}")
            return response

    def process_request(self, request, spider):
        if self.is_domain_in_proxies(request) and not request.meta.get('proxy') == self.proxy_server:
            proxy_logger.info(f"using proxy for {request.url}")
            retry_times = request.meta.get('retry_proxy_times', 0)
            request.meta['proxy'] = self.proxy_server
            request.meta['retry_proxy_times'] = retry_times + 1
        return None

    def is_domain_in_proxies(self, request) -> bool:
        return request.cb_kwargs.get('domain_id') in self.use_proxy_by_domain
```

Ilustración 22: Código ProxyMiddleware.<sup>59</sup>

---

<sup>59</sup> Elaboración propia.



Ilustración 23: Métricas de tiempo de ejecución a través de profiling de meta\_scrapers sin interacciones con la base de datos.<sup>60</sup>

<sup>60</sup> Elaboración propia.

```

for link_text in subsites_to_explore:
    # languagexpath
    follow_links += response.xpath(
        f'//a[contains(translate(@href, "ABCDEFGHIJKLMNOPQRSTUVWXYZ", "abcdefghijklmnopqrstuvwxyz"), "{link_text}") or contains(translate(text(), "ABCDEFGHIJKLMNOPQRSTUVWXYZ", "abcdefghijklmnopqrstuvwxyz"), "{link_text}")]//@href').getall()

# languagepython regex
ignore_pages_regex = [
    # non html files. Some sites have args in the url, so we need to check if is the end of string or a question mark.
    r'^\.(css|js|ico|png)(?:$|\?)',
    # references to javascript,mailto, tel
    r'^\!(?:(javascript|mailto|tel))',
    # mails
    r'^\!mailto:\!\@\!(\w+\.)?\w+\!\@?',
    # WordPress management and content pages
    r'^\!(?:(wp-json|wp-content|wp-includes))',
    # Used by Windows Live Writer to offer editing and publishing functionality. Not relevant. (BUT can be useful to find pages that are not in the main menu)
    r'^\!manifest\.xml',
    # wp-login is the login page for admin users.
    r'^\!wp-login\.png',
    # declares endpoints or 'structures' in https://wp?rsd. Can be useful to find more pages.
    r'^\!https?\.png',
]

ignore_pages_regex = [re.compile(_.regex) for _regex in ignore_pages_regex]
follow_links = set([str(link).lower() for link in follow_links if not any([_.regex.search(link) for _regex in ignore_pages_regex]) or original_url == link])

```

Ilustración 24: Filtración de dominios en función parse\_page.

#### 4. Evaluación de cambios

```

select *
from meta_scrapes
where meta_scraping_id = 'OLD_META_SCRAPING_ID'
    or meta_scraping_id = 'NEW_META_SCRAPING_ID'
group by meta_scraping_id;

```

Ilustración 25: Consulta para obtención de información para la comparación de cambios.

```

select meta_scrapes.meta_scraping_id,
       sum(proxy_used) as proxy_used,
       count(*)         as count
  from meta_scrapes
 inner join meta_general_information mgi
    |          on meta_scrapes.meta_scraping_id = mgi.meta_scraping_id and mgi.domain_id = meta_scrapes.domain_id
 where meta_scrapes.meta_scraping_id = 'NEW_META_SCRAPING_ID'
group by meta_scraping_id;

```

Ilustración 26: Consulta para obtención de información para la comparación de cambios a nivel de uso de proxies.

#### 5. REDIS para la asignación de tareas y mejora de índices y consultas.

Para el uso de REDIS se incorporaron diferentes componentes al meta scraper.

## 1. Máquina Docker REDIS.

```
FROM redis:latest

COPY redis.conf /usr/local/etc/redis/redis.conf
COPY users.acl /etc/redis/users.acl

EXPOSE 6379

# Command to run when starting the container
CMD ["redis-server", "/usr/local/etc/redis/redis.conf"]
```

Ilustración 27: Configuración de máquina Docker REDIS en el puerto 6379

## 2.Creador de tareas

```
redisClient = redis.from_url('redis://@localhost:6379')

sites_per_query = 20

allowed_countries = ["DK", "SE", "NO", "FI", "NL", "EE", "LT", "LV"]
country_code = get_country_ip()

key_to_eval = f'{country_code}:meta_spider:start_urls'

available_meta_scraping_ids = get_current_scraping_ids()

for meta_scraper_id in available_meta_scraping_ids['meta_scraping_id'].to_list():
    print(f'Meta scraper id: {meta_scraper_id}')
    sites_to_be_meta_scraped = get_x_sites_to_be_meta_scraped(country_code, limit=sites_per_query,
                                                               meta_scraper_id=meta_scraper_id)
    while not sites_to_be_meta_scraped.empty():
        print("Worker scraping chunk")
        dict_of_domain_ids_and_urls = [
            {'domain_id': domain['domain_id'],
             'url': domain['url']}
            for domain in sites_to_be_meta_scraped.to_dict('records')
        ]
        engine_update = get_db_engine(**get_db_credentials(MYSQL_DB=SCRAPERS_SCHEMA, option="UPDATE"))
        active_connection_update = engine_update.connect()

        log_meta_scraping_times_for_one_domain(
            timestamp_column="meta_scraping_start",
            meta_scraping_id=meta_scraper_id,
            domain_ids=sites_to_be_meta_scraped['domain_id'].to_list(),
            active_connection_update=active_connection_update
        )
        active_connection_update.close()
        engine_update.dispose()

        scraping_information = {
            'meta_scraping_id': meta_scraper_id,
            'domain_ids_and_urls': dict_of_domain_ids_and_urls
        }
        serialized_domain = json.dumps(scraping_information)
        redisClient.rpush(key_to_eval, serialized_domain)
        print(f'Added {len(dict_of_domain_ids_and_urls)} domains to queue')

        sites_to_be_meta_scraped = get_x_sites_to_be_meta_scraped(country_code, limit=sites_per_query,
                                                               meta_scraper_id=meta_scraper_id)
```

Ilustración 28: Código de máquina creadora de tareas.<sup>61</sup>

### 3. Meta scraper

```
def next_requests(self):
    """Returns a request to be scheduled or none."""
    self.company_numbers_found = {}
    found = 0
    datas = self.fetch_data(self.redis_key, self.redis_batch_size)
    print(f'Processing {len(datas)} domains')
    self.logger.debug(f'Fetched {len(datas)} datas from '{self.redis_key}')
    meta_scraping_start = time.strftime('%Y-%m-%d %H:%M:%S')
    for data in datas:
        scraping_info = self.retrieve_data(data)
        print(f'scraping info: {scraping_info}')
        reqs = self.make_request_from_data(scraping_info)
        for req in reqs:
            yield req
            found += 1
            self.logger.info(f'start req url:{req.url}')

    if found:
        self.logger.debug(f'Read {found} requests from '{self.redis_key}')

    meta_scraping_end = time.strftime('%Y-%m-%d %H:%M:%S')

    items = singleton_storage_client.get_and_remove_many_keys([item['domain_id'] for item in scraping_info['domain_ids_and_urls']], meta_scraping_start, meta_scraping_end)
    self._add_items_to_redis_queue(items, meta_scraping_id=scraping_info['meta_scraping_id'])

def _add_items_to_redis_queue(self, items: dict, meta_scraping_id: str):
    self.server.rpush(self.redis_items_key.format(meta_scraping_id=meta_scraping_id), json.dumps(items))

def make_request_from_data(self, data: dict):
    for domain_data in data['domain_ids_and_urls']:
        self.requested_sitemaps_by_domain_id[domain_data['domain_id']] = set()

    return [scrapy.Request(
        url=prefix + domain_data['url'],
        callback=self.parse,
        dont_filter=True,
        headers=self.headers,
        errback=self.parse_error,
        cb_kwargs=dict(domain_id=domain_data['domain_id'], twisted_errors=0),
    ) for prefix in ["https://", "http://", "http://www."] for domain_data in data['domain_ids_and_urls']]
```

Ilustración 29: Código Meta scraper para la obtención de tareas almacenadas en REDIS.<sup>62</sup>

---

<sup>61</sup> Elaboración propia.

<sup>62</sup> Elaboración propia.

#### 4. Procesador de ítems

```
redisClient = redis.from_url('redis://@localhost:6379')

key_to_eval = "meta_spider:items:{meta_scraping_id}"

meta_scraping_id = 'META_SCRAPING_ID'
country_code = 'DK'

key_to_eval = key_to_eval.format(meta_scraping_id=meta_scraping_id)

while not redisClient.llen(key_to_eval):

    json_data = redisClient.lpop(key_to_eval)

    json_data: FoundAndNotFoundResponse = json.loads(json_data)

    domain_ids = json_data['domain_ids']
    meta_scraping_start = json_data['meta_scraping_start']
    meta_scraping_end = json_data['meta_scraping_end']

    engine_update = get_db_engine(**get_db_credentials(MYSQL_DB=SCRAPERS_SCHEMA, option="UPDATE"))
    active_connection_update = engine_update.connect()

    log_meta_scraping_times(
        meta_scraping_id=meta_scraping_id,
        domain_ids=domain_ids,
        active_connection_update=active_connection_update,
        meta_scraping_start=meta_scraping_start,
        meta_scraping_end=meta_scraping_end
    )

    print("Worker processing json data file and adding to database")
    process_json_file_and_add_meta_data_to_db(meta_scraping_id=meta_scraping_id,
                                                country_code=country_code,
                                                domain_ids=domain_ids,
                                                singleton_data=json_data)
    active_connection_update.close()
    engine_update.dispose()
```

Ilustración 30: Código procesador de ítems.<sup>63</sup>

---

<sup>63</sup> Elaboración propia.

## 6. Establecer los parámetros de la herramienta Scrapy

```
df = df.groupby(['download_delay', 'download_timeout', 'concurrent_requests', 'concurrent_requests_per_domain', 'reactor_threadpool_maxsize']).mean()
df = df.reset_index()

df.to_csv('summary.csv', index=False)

X = df[['download_delay', 'download_timeout', 'concurrent_requests']]
y = df['elapsed_time_seconds']

corr_matrix = df[['download_delay', 'download_timeout', 'concurrent_requests', 'concurrent_requests_per_domain', 'reactor_threadpool_maxsize', 'elapsed_time_seconds']].corr()

sns.heatmap(corr_matrix, annot=True, cmap='Blues')
plt.rcParams['figure.figsize'] = [15, 12]
plt.xticks(fontsize=8)
plt.yticks(fontsize=8, rotation=0)
# make the space of the labels smaller
plt.tight_layout()

plt.show()

X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2)
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

y_pred = linear_model.predict(X_test)

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

X_train_sm = sm.add_constant(X_train)
model_sm = sm.OLS(y_train, X_train_sm).fit()

print(model_sm.summary())

vif_data = pd.DataFrame()
vif_data["feature"] = X_train.columns
vif_data["VIF"] = [variance_inflation_factor(X_train.values, i) for i in range(len(X_train.columns))]

linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

y_pred = linear_model.predict(X_test)

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"R2: {r2}")
```

Ilustración 31: Código regresión lineal para tiempo de ejecución.<sup>64</sup>

---

<sup>64</sup> Elaboración propia.

```

=====
Dep. Variable:      elapsed_time_seconds R-squared:          0.814
Model:                          OLS Adj. R-squared:         0.792
Method:                     Least Squares F-statistic:    36.72
Date:           Sun, 26 Nov 2023 Prob (F-statistic): 2.74e-14
Time:            19:29:36 Log-Likelihood:             -250.38
No. Observations:                  48 AIC:                   512.8
Df Residuals:                      42 BIC:                   524.0
Df Model:                           5
Covariance Type:            nonrobust
=====
coef std err t P>|t|    [ 0.025   0.975]
-----
const 380.2309 30.104 12.630 0.000 319.478 440.984
download_delay 12.2347 3.667 3.336 0.002 4.834 19.636
download_timeout 0.5460 0.218 2.499 0.010 0.105 0.987
concurrent_requests -7.5841 0.912 -8.317 0.000 -9.424 -5.744
concurrent_requests_per_domain 3.0788 5.014 0.614 0.543 -7.942 13.198
reactor_threadpool_maxsize 1.0782 0.864 1.248 0.219 -0.665 2.822
=====
Omnibus:                 1.083 Durbin-Watson:          1.558
Prob(Omnibus):           0.582 Jarque-Bera (JB):  0.530
Skew:                    -0.238 Prob(JB):            0.767
Kurtosis:                3.196 Cond. No. 376.
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
feature VIF
0 download_delay 4.663202
1 download_timeout 2.907818
2 concurrent_requests 5.310623
3 concurrent_requests_per_domain 5.048221
4 reactor_threadpool_maxsize 5.549053
RMSE: 36.592940115018266
R^2: 0.8412162561305507

```

*Ilustración 32: Primera regresión lineal para tiempo de ejecución.*<sup>65</sup>

---

<sup>65</sup> Elaboración propia.

```

OLS Regression Results

=====
Dep. Variable: elapsed_time_seconds R-squared:      0.808
Model:           OLS Adj. R-squared:      0.795
Method:          Least Squares F-statistic:     61.89
Date: Sun, 26 Nov 2023 Prob (F-statistic):    7.91e-16
Time: 19:32:50 Log-Likelihood:             -251.27
No. Observations:      48 AIC:                  510.5
Df Residuals:         44 BIC:                  518.0
Df Model:            3
Covariance Type:    nonrobust
=====
coef std err t P>|t|      [ 0.025      0.975 ]
-----
const 400.0861 28.324 14.125 0.000 343.003 457.169
download_delay 14.5360 3.697 3.932 0.000 7.086 21.986
download_timeout 0.4602 0.213 2.156 0.037 0.030 0.890
concurrent_requests -7.2622 0.726 -9.998 0.000 -8.726 -5.798
=====
Omnibus:            2.633 Durbin-Watson:        2.338
Prob(Omnibus):      0.268 Jarque-Bera (JB):   1.649
Skew:               -0.373 Prob(JB):          0.438
Kurtosis:           3.518 Cond. No. 356.
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified
feature VIF
0 download_delay 3.810161
1 download_timeout 1.992287
2 concurrent_requests 2.750509
RMSE: 34.692149519803294
R^2: 0.8498189582263032

```

Ilustración 33: Segunda regresión lineal para tiempo de ejecución.<sup>66</sup>

```

X = df[['download_timeout', 'concurrent_requests', 'reactor_threadpool_maxsize']]

y = df['downloader/response_status_count/200'].apply(lambda x: min(x / 82, 1))

print(max(y))
print(min(y))

y = y + 1 - max(y)

```

---

<sup>66</sup> Elaboración propia.

Ilustración 34: Código extra para realizar la regresión lineal del error.<sup>67</sup>

```
OLS Regression Results
=====
Dep. Variable: downloader/response_status_count/200 R-squared:      0.789
Model:           OLS Adj. R-squared:      0.764
Method:          Least Squares F-statistic:     31.50
Date:        Tue, 28 Nov 2023 Prob (F-statistic): 3.47e-13
Time:        20:03:51 Log-Likelihood:      52.946
No. Observations:      48 AIC:            -95.89
Df Residuals:       42 BIC:            -82.66
Df Model:             5
Covariance Type:    nonrobust
=====
coef std err t P>|t|
[0.025   0.975]
-----
const 0.9789 0.061 16.152 0.000 0.857 1.101
download_delay -0.0054 0.007 -0.774 0.443 -0.019 0.009
download_timeout 0.0012 0.000 2.954 0.005 0.000 0.002
concurrent_requests -0.0113 0.002 -7.355 0.000 -0.014 -0.008
concurrent_requests_per_domain -0.0077 0.010 -0.789 0.435 -0.027 0.012
reactor_threadpool_maxsize 0.0013 0.002 0.730 0.469 -0.002 0.005
=====
Omnibus:           48.345 Durbin-Watson:      1.806
Prob(Omnibus):    0.000 Jarque-Bera (JB): 242.846
Skew:              -2.521 Prob(JB):      1.85e-53
Kurtosis:          12.798 Cond. No. 435.
=====
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
feature VIF
0 download_delay 4.741685
1 download_timeout 3.889254
2 concurrent_requests 4.294674
3 concurrent_requests_per_domain 4.088734
4 reactor_threadpool_maxsize 5.893881
RMSE: 0.10247089366589221
R^2: 0.8226257267237835
```

Ilustración 35: Primera regresión lineal para error.<sup>68</sup>

---

<sup>67</sup> Elaboración propia.

<sup>68</sup> Elaboración propia.

```

=====
Dep. Variable: downloader/response_status_count/200 R-squared:      0.799
Model:                                         OLS   Adj. R-squared:      0.786
Method:                                         Least Squares F-statistic:      58.46
Date:           Tue, 28 Nov 2023 Prob (F-statistic):    2.16e-15
Time:           20:04:17 Log-Likelihood:      46.406
No. Observations:      48 AIC:                  -84.81
Df Residuals:        44 BIC:                  -77.33
Df Model:                   3
Covariance Type:      nonrobust
=====

      coef    std err      t      P>|t|      [0.025      0.975]
-----
const            0.8733     0.057    15.210      0.000      0.758     0.989
download_timeout  0.0017     0.000     3.825      0.000      0.001     0.003
concurrent_requests -0.0116    0.002    -7.551      0.000     -0.015    -0.008
reactor_threadpool_maxsize  0.0028    0.002     1.746      0.088     -0.000     0.006
=====

Omnibus:             20.102 Durbin-Watson:      1.808
Prob(Omnibus):       0.000 Jarque-Bera (JB):    29.860
Skew:                -1.344 Prob(JB):          3.28e-07
Kurtosis:              5.776 Cond. No.         355.
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
feature      VIF
0 download_timeout  3.111460
1 concurrent_requests  2.087681
2 reactor_threadpool_maxsize  4.449264
RMSE: 0.04034458633017521
R^2: 0.9341761999721068

```

*Ilustración 36: Segunda regresión lineal para error.*<sup>69</sup>

---

<sup>69</sup> Elaboración propia.

```

def constraint(download_timeout, concurrent_requests, reactor_threadpool_maxsize):
    return 0.8438 + download_timeout * 0.0019 + concurrent_requests * -0.0117 + reactor_threadpool_maxsize * 0.0034

def elapsed_time(download_delay, download_timeout, concurrent_requests):
    return 400.0861 + download_delay * 14.5360 + download_timeout * 0.4602 + concurrent_requests * -7.2622

def is_valid(concurrent_requests, concurrent_requests_per_domain, reactor_threadpool_maxsize):
    return (concurrent_requests > 0 and concurrent_requests_per_domain > 0 and
           isinstance(concurrent_requests, int) and isinstance(concurrent_requests_per_domain, int) and
           reactor_threadpool_maxsize > 0 and isinstance(reactor_threadpool_maxsize, int))

def optimize(initial_state, error_threshold=0.05):
    min_percent_of_success_requests = 1 - error_threshold
    best_state = initial_state
    best_time = elapsed_time(*initial_state[:3])

    download_delay_range = np.arange(1, 10, 0.5)
    download_timeout_range = np.arange(10, 120, 5)
    concurrent_requests_range = range(1, 200)
    concurrent_requests_per_domain_range = range(1, 100)
    reactor_threadpool_maxsize_range = range(1, 25)

    for dd, dt, cr, crpd, rtms in itertools.product(download_delay_range, download_timeout_range, concurrent_requests_range, concurrent_requests_per_domain_range,
                                                    reactor_threadpool_maxsize_range):
        if is_valid(cr, crpd, rtms) and constraint(dt, cr, rtms) > min_percent_of_success_requests:
            current_time = elapsed_time(dd, dt, cr)
            if current_time < best_time:
                best_time = current_time
                best_state = [dd, dt, cr, crpd, rtms]

    return best_state, best_time

```

Ilustración 37: Función que obtiene el mínimo tiempo de ejecución dado un error máximo.<sup>70</sup>

## 7. Relación número de dominios con el tamaño del mercado

País	Dominios	Tamaño mercado (mil millones de euros)
DK	114089	22,8
FI	40305	5,84
NL	204992	27,9
NO	54583	10,78
SE	141217	12,76

Tabla 23: Relación número de dominios con el tamaño del mercado resulta en una correlación de 0,845.

---

<sup>70</sup> Elaboración propia.