EJERCICIOS PRÁCTICOS

URL SHORTENER

Daniel Blanco Calviño

REQUISITOS

- **Requisitos funcionales.**
  - Dada una URL, debemos transformarla en una URL acortada y devolverla al usuario.
  - Dada una URL acortada, debemos redirigir al usuario a la dirección original.

- **Requisitos no funcionales.**
  - Carga intensa.
  - Alta disponibilidad.

HIPÓTESIS Y ESTIMACIONES

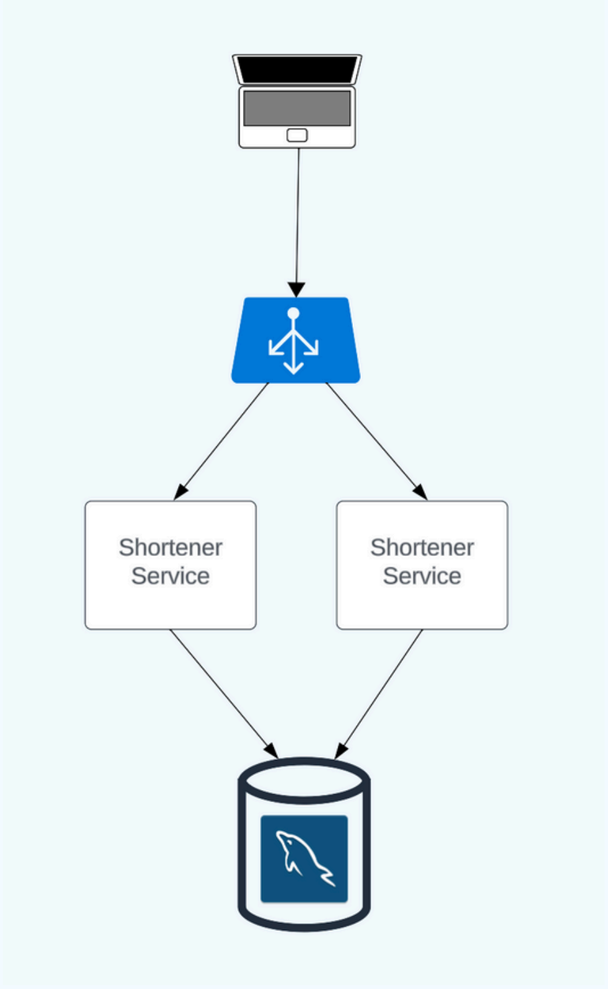
- 50 millones de URLs nuevas cada día.
  - Deben permanecer en el sistema al menos 10 años.
    - Debemos soportar 50M * 365 * 10 = **182.500M de registros.**

- Tamaño de URL media = 100 caracteres.
  - Almacenamiento necesario: 100 * 182.500M = **182.5 terabytes.**

- El tamaño de la URL debe ser el mínimo posible.

- Relación entre las lecturas y las escrituras de 10:1.

DISEÑO ALTO NIVEL



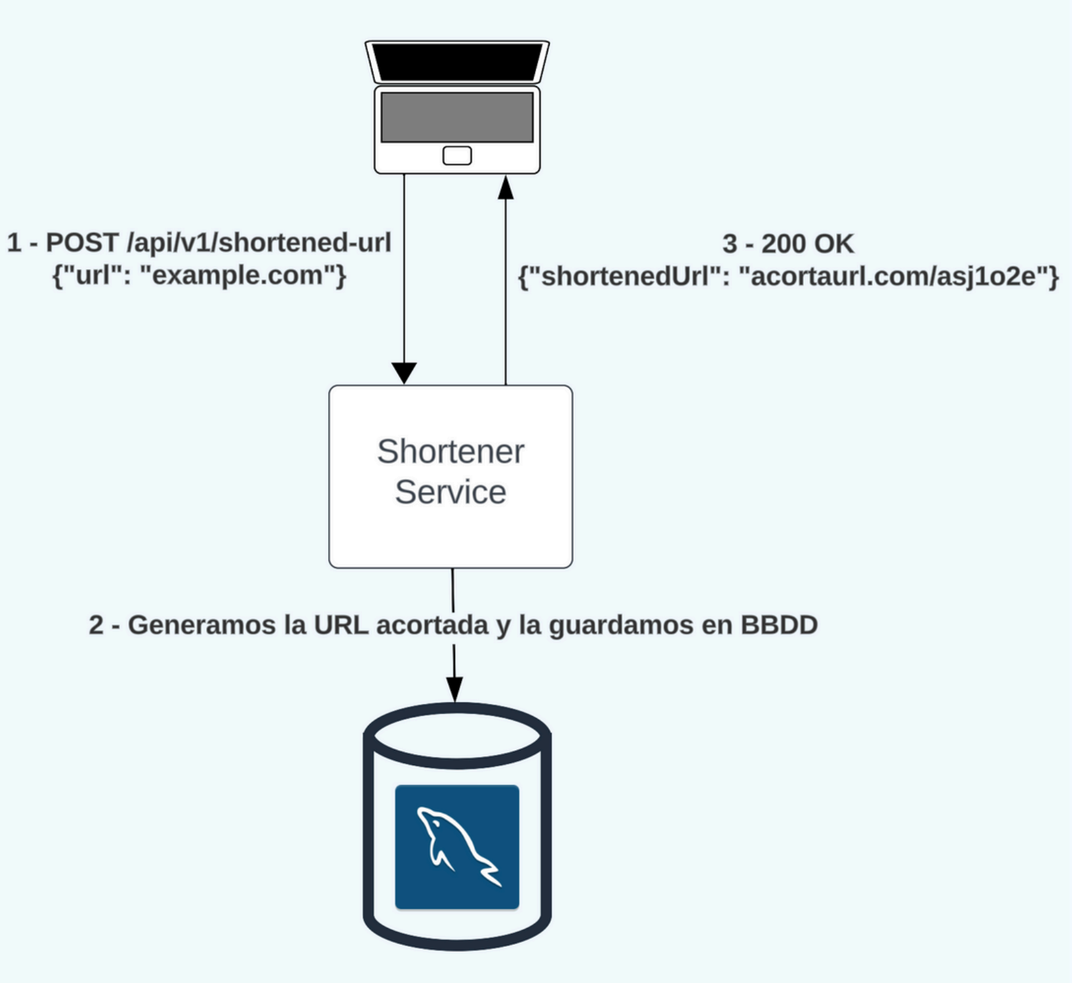
API ENDPOINTS

- **POST /api/v1/shortened-url**
  - Body: {”url”: “example.com”}
  - Response: ¨{”shortenedUrl”: “acortaurl.com/h2ik3k”}

- **GET acortaurl.com/h2ik3k**
  - Redirecciona a “example.com”.

ACORTAR URLS

- Podemos aplicar un **hash** a la URL original para crear la URL acortada.
  - hash(example.com): **asj1o2e**
  - URL acortada: acortaurl.com/**asj1o2e**



ACORTAR UNA URL

- Podemos aplicar un **hash** a la URL original para crear la URL acortada.
  - hash(example.com): **asj1o2e**
  - URL acortada: acortaurl.com/**asj1o2e**

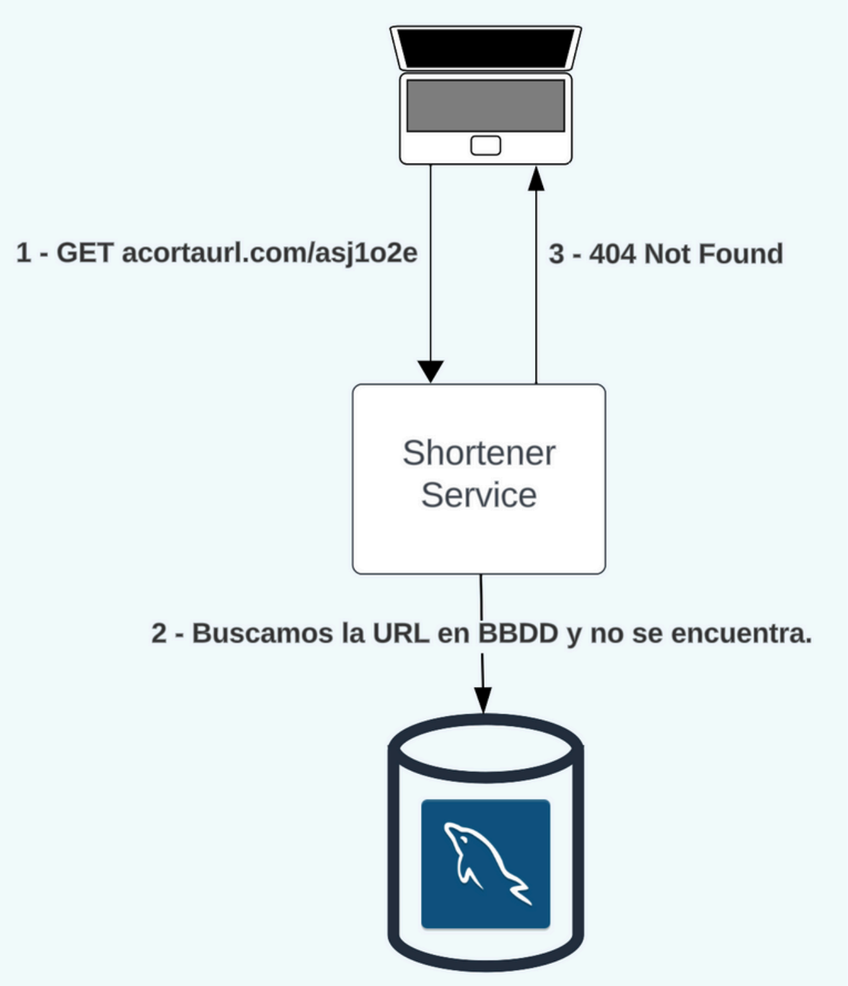
- La complicación que presenta son las **colisiones**.
  - Dos URLs distintas que presenten el mismo hash.
  - Si tenemos una colisión, **no podremos mapear la URL acortada a la URL original**.
  - hash(example.com): asj1o2e y hash(anotherexample.com): asj1o2e
    - ¿acortaurl.com/**asj1o2e** redirecciona a example.com o a anotherexample.com?

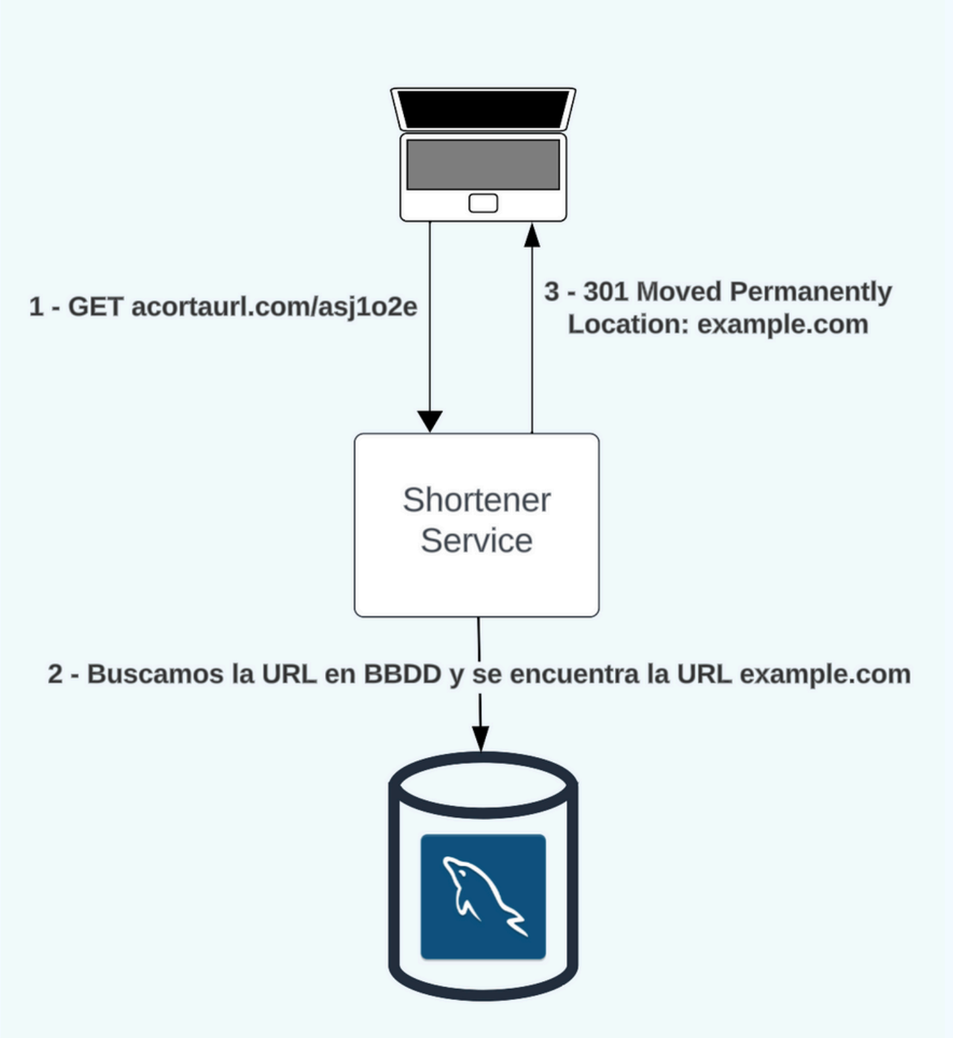
REDIRECCIÓN A LA URL ORIGINAL

- Redireccionamiento HTTP.
  - Código **301** Moved Permanently.
    - Indica que el recurso se ha movido **permanentemente** a la nueva dirección.
    - El navegador podrá **cachear** la petición. Es el más indicado si queremos reducir carga en nuestros servidores.

  - Código **302** Found.
    - El recurso se ha movido **temporalmente** a otra dirección.
    - El navegador **no va a cachear la nueva localización**, ya que el recurso puede volver a la url original en cualquier momento. Más adecuado si queremos llevar un registro de todas las peticiones.

REDIRECCIÓN A LA URL ORIGINAL





MODELO DE DATOS

url
id {PK} (autoincrement)
original_url
shortened_url

- **Replicación de datos** entre múltiples datacenters.
  - Aseguramos la **disponibilidad y la fiabilidad** del sistema

- Si es necesario, podríamos aplicar sharding.
  - No debería serlo, al tratarse de un **modelo y consultas sencillas**.

ACORTAR URLS EN DETALLE

- Requisito importante: El tamaño de la URL debe ser el **mínimo posible.**

- Caracteres soportados en nuestras URLs.
  - [0-9], [A-Z] y [a-z]. Un total de **62 caracteres.**

- Número de URLs soportadas.
  - 1 caracter: 62
    - acortaurl.com/0
    - acortaurl.com/1
    - acortaurl.com/z
  - 2 caracteres: 62^2
  - 3 caracteres: 62^3

ACORTAR URLS EN DETALLE

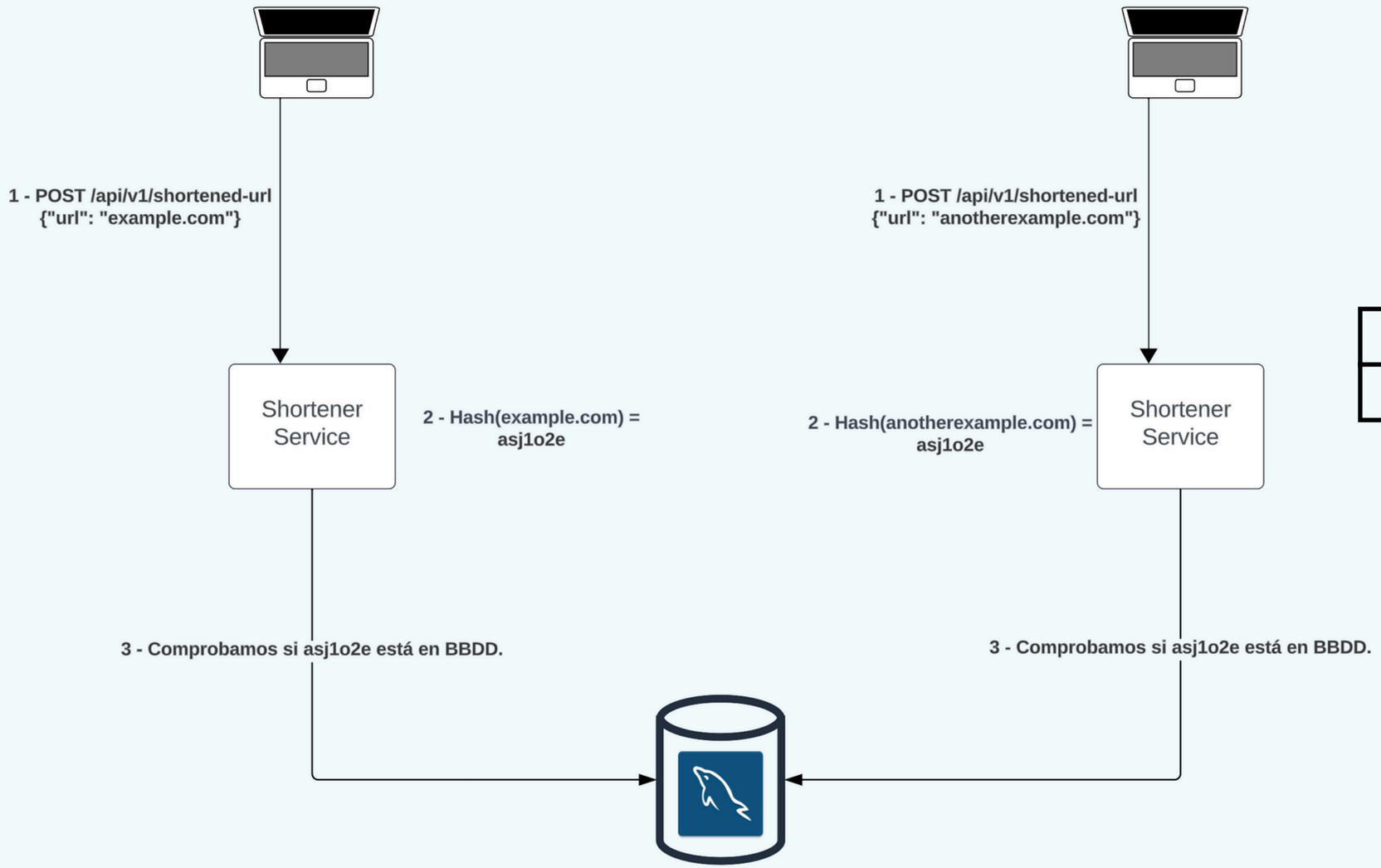
Potencia de 62	Valor
1	62
2	3.844
3	238.328
4	14.776.336
5	916.132.832
6	~57 mil millones < 182.5 mil millones
7	**~3.5 billones > 182.5 mil millones**

ACORTAR URLS - OPCION 1: HASHING

- Aplicamos una función hash a la URL original para obtener la URL acortada.
  - MD5
  - SHA-1
  - SHA-256

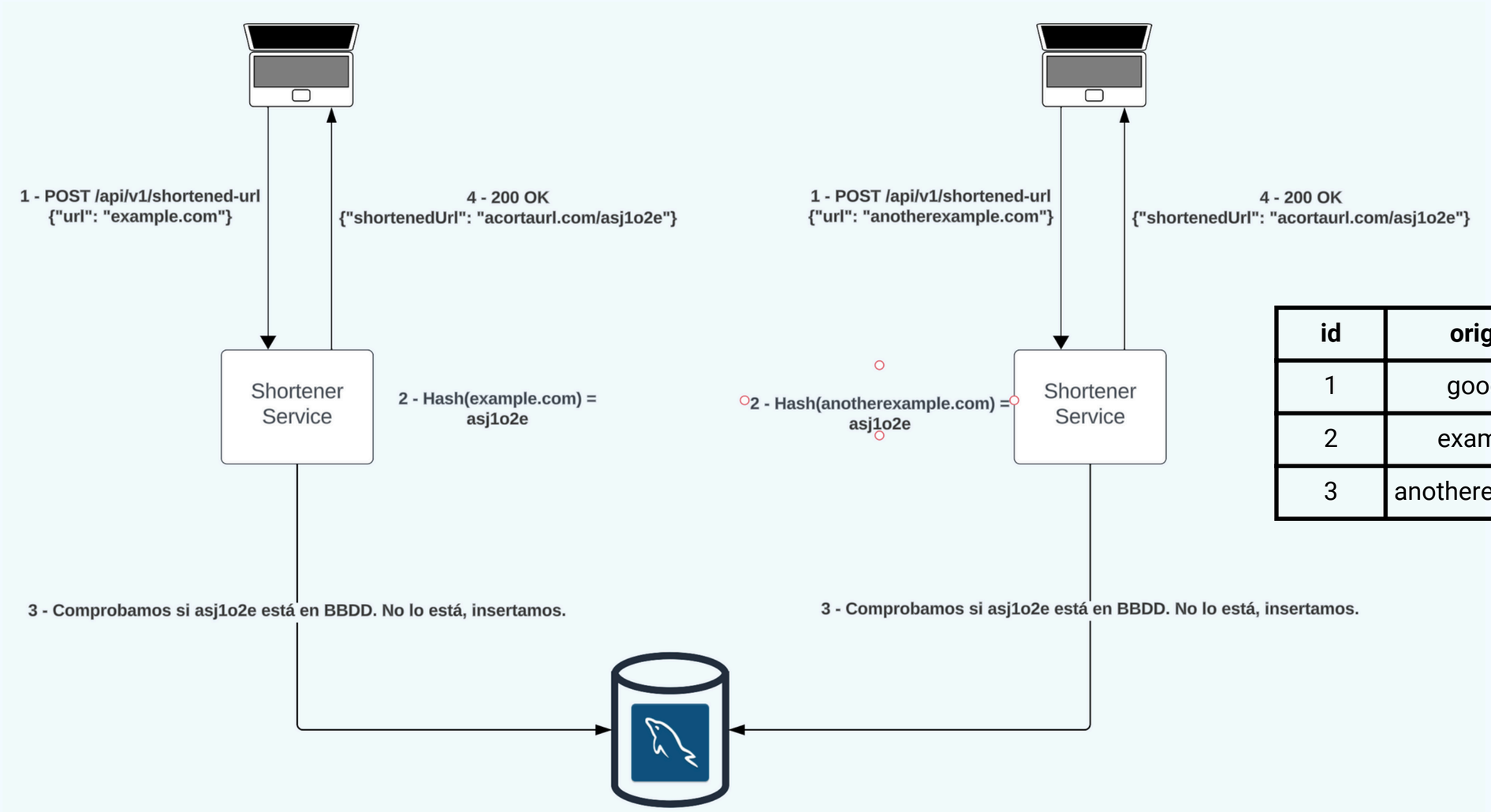
- Problemas.
  - El resultado de esos algoritmos son strings de tamaño mayor a 7 caracteres.
    - Podríamos tomar los 7 primeros.
  - **Colisiones**.
    - Tendríamos que **consultar la BBDD** para comprobar si ya existe el hash.
    - Debemos **bloquear las demás transacciones** para evitar carreras críticas.
    - No es factible para un sistema con tan alta carga.

ACORTAR URLS - OPCION 1: HASHING



id	original_url	shortened_url
1	google.com	2gshwa8

ACORTAR URLS - OPCION 1: HASHING



id	original_url	shortened_url
1	google.com	2gshwa8
2	example.com	asj1o2e
3	anotherexample.com	asj1o2e

ACORTAR URLS - OPCION 2: GENERADOR DE IDS

- No necesitamos hashear la URL original.
  - Necesitamos un valor único entre 0000000 y zzzzzzz para cada petición.

- Debemos generar un identificador único en un entorno distribuido de forma eficiente.

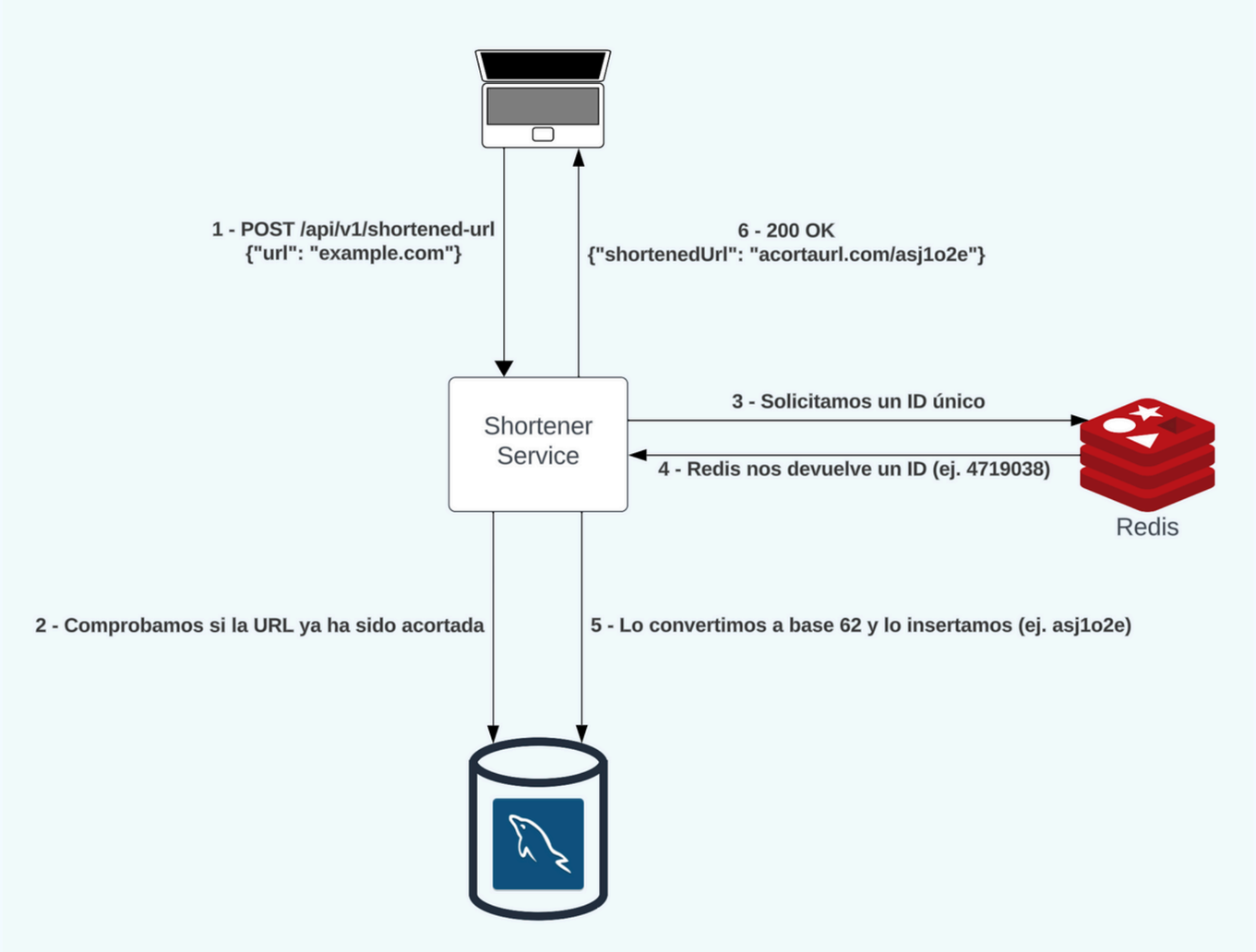
ACORTAR URLS - OPCION 2: GENERADOR DE IDS

- No necesitamos hashear la URL original.
  - Necesitamos un valor único entre 0000000 y zzzzzzz para cada petición.

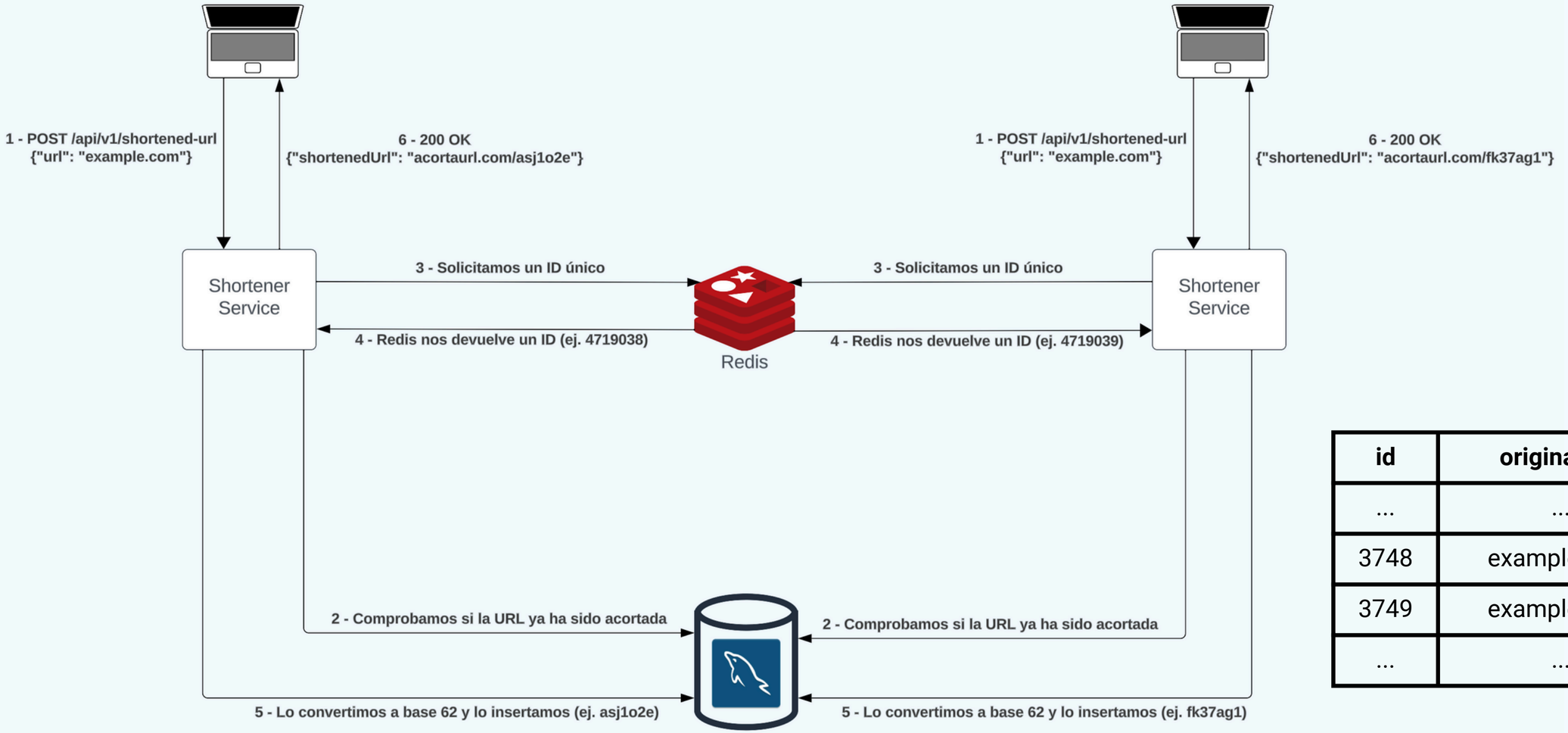
- Debemos generar un identificador único en un entorno distribuido de forma eficiente.
  - Problema muy complejo.
  - Redis proporciona esa funcionalidad.

- Podemos solicitar un nuevo identificador que se incremente en cada petición.
  - Está en base 10, por lo que debemos convertirlo a base 62.

ACORTAR URLS - OPCION 2: GENERADOR DE IDS

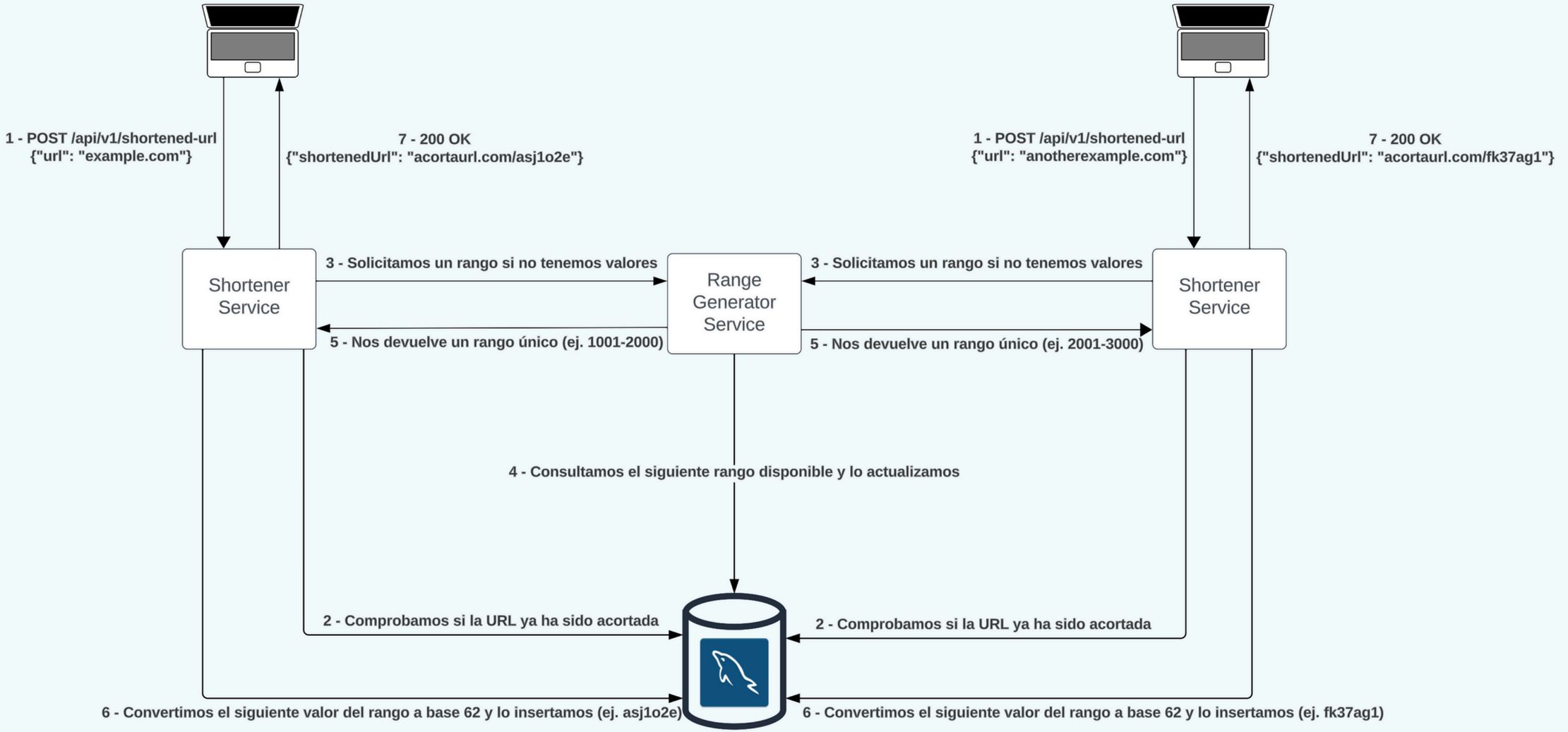


ACORTAR URLS - OPCION 2: GENERADOR DE IDS

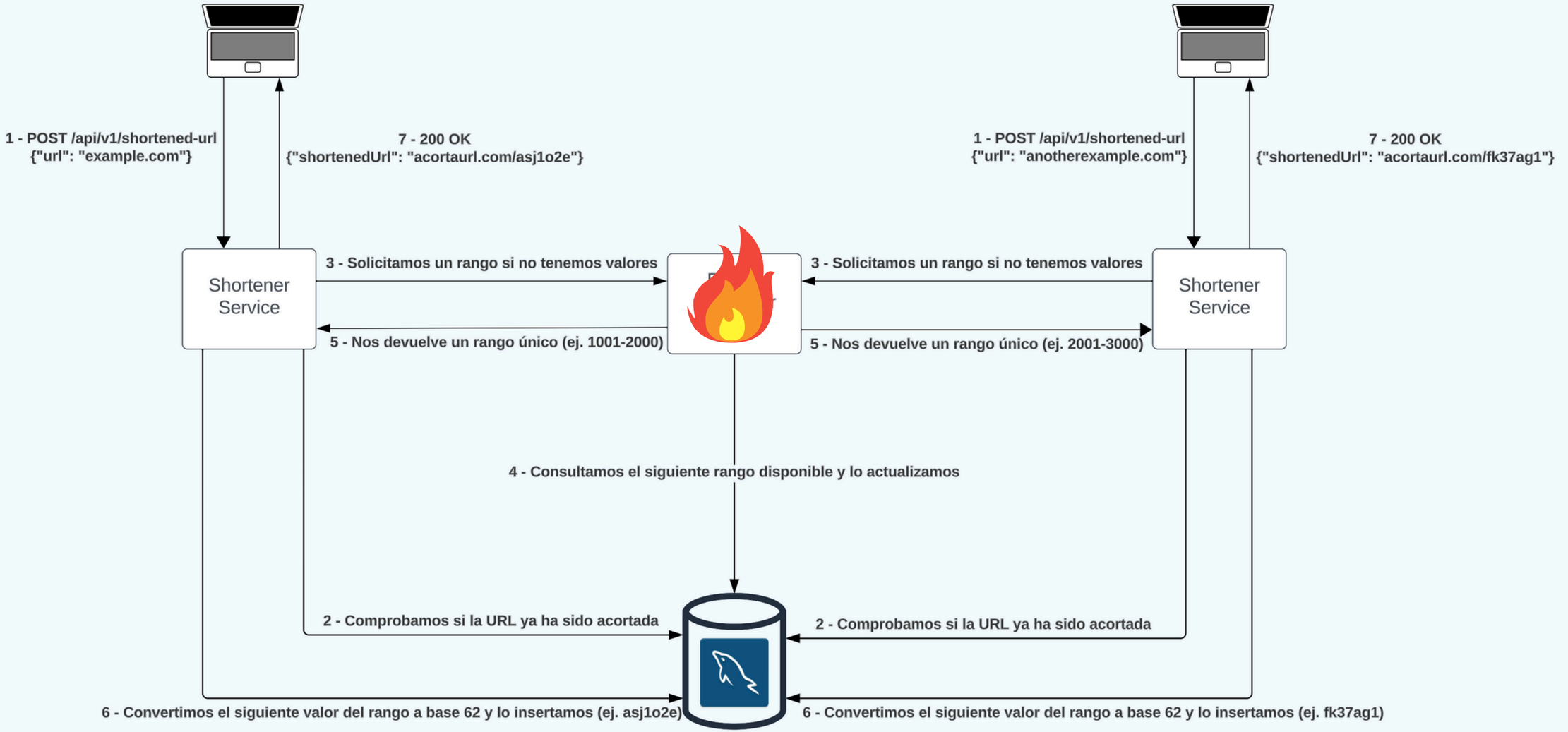


id	original_url	shortened_url
...	...	...
3748	example.com	asj1o2e
3749	example.com	fk37ag1
...	...	...

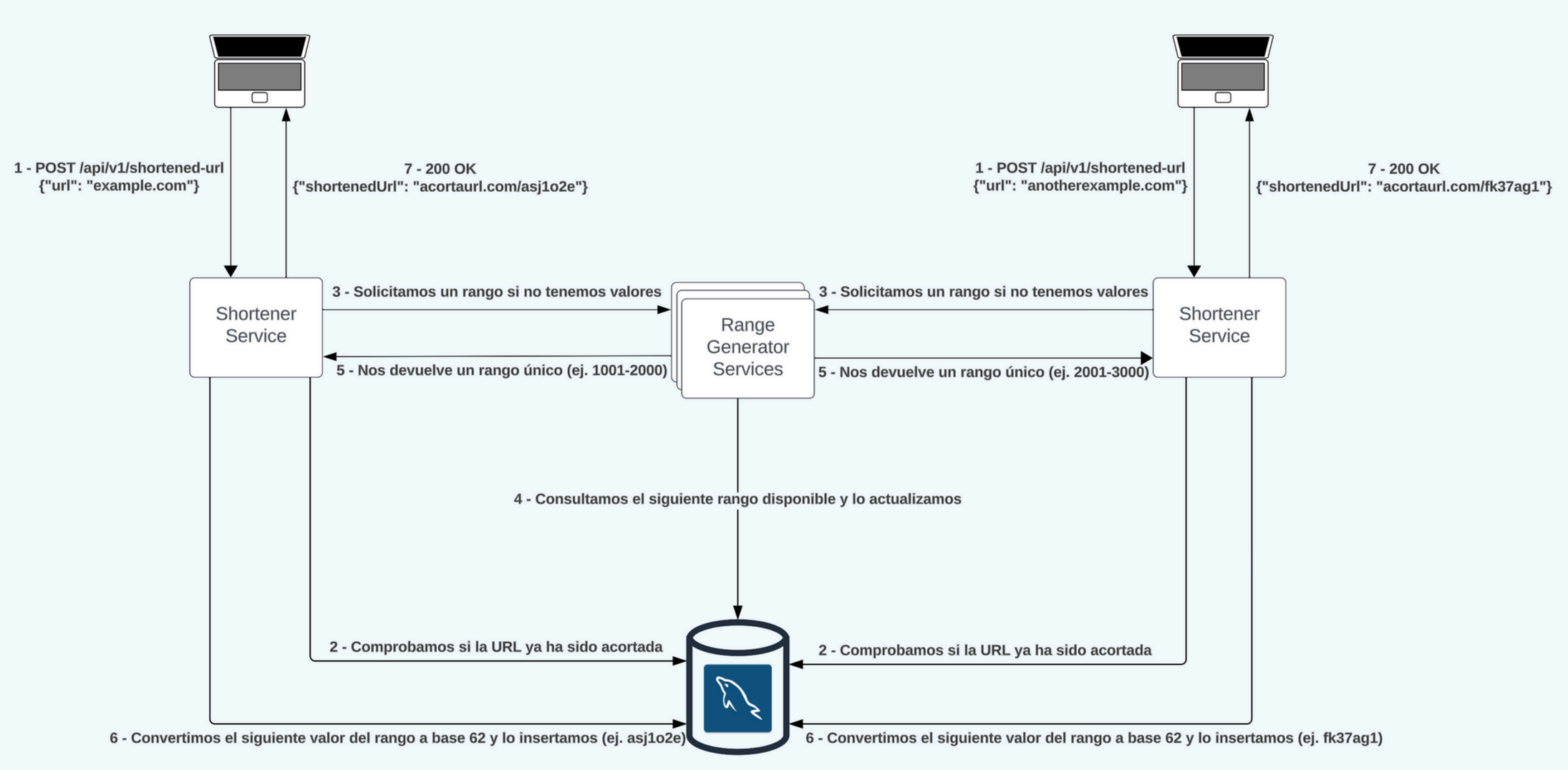
ACORTAR URLS - OPCION 3: GENERADOR DE RANGOS



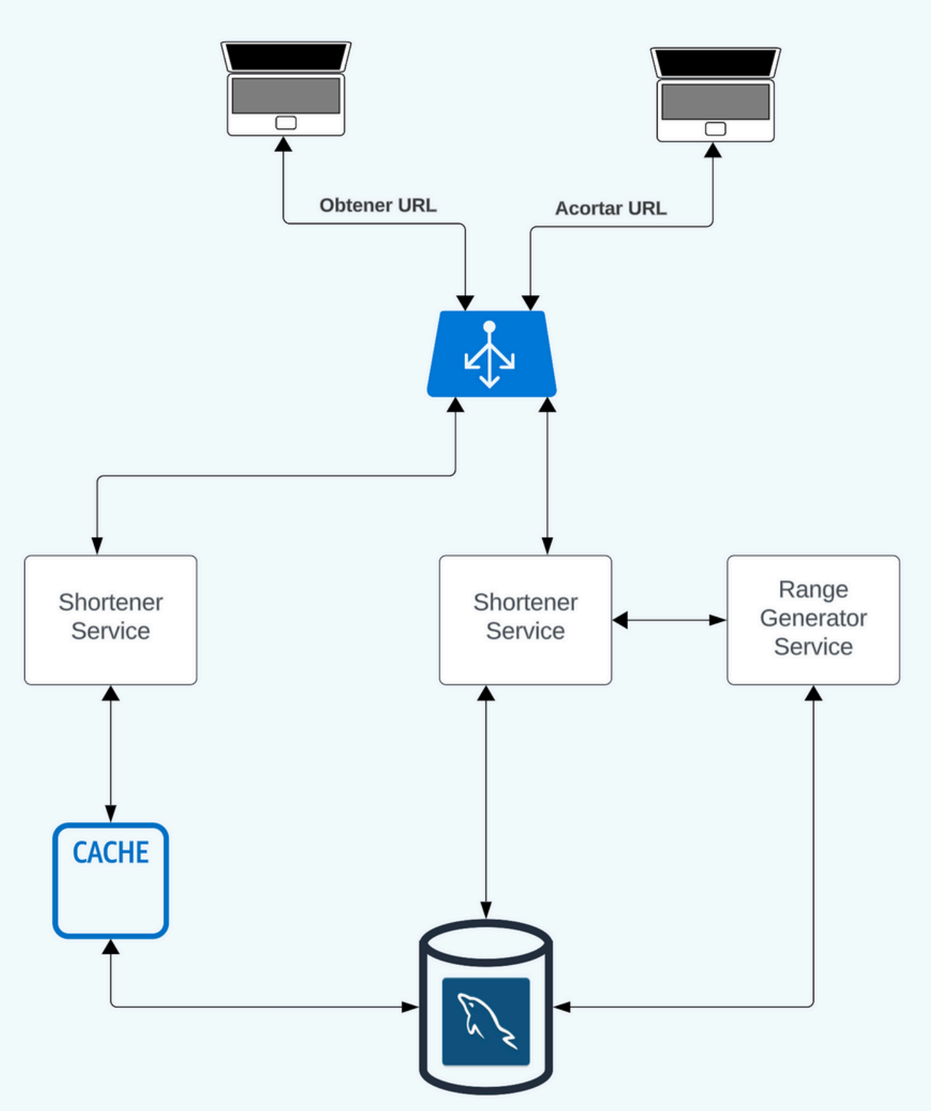
ACORTAR URLS - OPCION 3: GENERADOR DE RANGOS



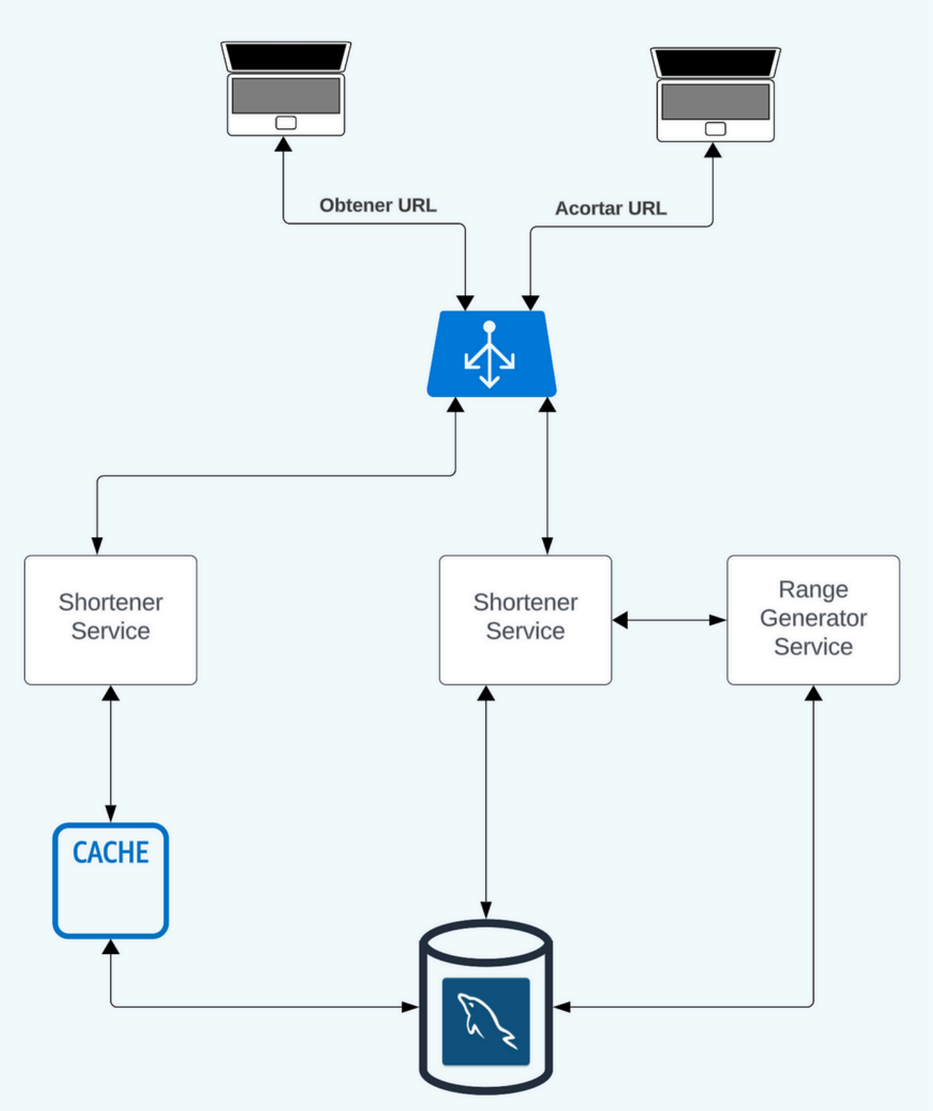
ACORTAR URLS - OPCION 3: GENERADOR DE RANGOS



DISEÑO FINAL



DISEÑO FINAL



- Servicios escalados horizontalmente.
- Cache **read-through para mejorar las lecturas.**
  - Relación 10:1 entre lecturas y escrituras.
- Despliegue en múltiples datacenters con replicación de datos.
  - Sharding en caso de ser necesario.