
Práctica 01

Bases de Datos no Estructuradas 2022-2

Marcela Cruz Larios
Yeudiel Lara Moreno
Pablo D. Castillo del Valle

Introducción

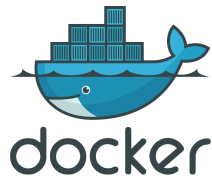
Para esta practica utilizaremos redis, docker y python.

Redis es una base de datos no estructurada basada en tablas de hashes llave-valor, debido a esto redis nos provee de incrementos de velocidad inmensos con respecto a una base de datos tradicional.

Redis también puede funcionar como una base de datos persistente, sin embargo este no es su objetivo principal. Este se enfoca principalmente a ser una base de datos en memoria cache por lo que al final se usa como punto intermedio entre una base de datos más robusta.

Las ventajas en cuanto a rendimiento que nos ofrece redis son perfectas para bases de datos que requieran una cantidad inmensa de solicitudes en un corto periodo de tiempo.

Es por eso que para nuestro proyecto es conveniente usar redis ya que se trata de un administrador de sesiones y un acortador de urls.



Docker es un programa que automatiza el despliegue de entornos virtuales.

De esta forma podemos desplegar redis de manera independiente a nuestro entorno de desarrollo, casi aislado y de forma privada en su propio sistema operativo.

De esta forma desplegaremos en un contenedor de Docker, nuestra base de datos de redis.

Python es uno de los lenguajes de programación de alto nivel más populares actualmente, debido a la facilidad de aprenderlo y a la cantidad de bibliotecas con las que cuenta.

Para este proyecto lo utilizaremos como medio de comunicación con nuestra base de datos en redis en el contenedor de Docker, para esto haremos uso de la biblioteca `redis-py`, la cuál nos permitirá añadir, eliminar y manipular nuestras tablas de hashes.

Del mismo modo, utilizaremos python para la creación de un UI que le permita tanto al usuario como a los administradores interactuar con nuestra base.

Los usuarios podrán registrarse, iniciar sesión y cerrar sesión, así como acortar urls, añadir urls a una lista de deseos, eliminar urls de la lista de deseos, añadir urls a una categoría, cambiar una url de categoría y cambiar la privacidad de cada una de sus urls. Por otro lado, el rol de administrador permite consultar la lista completa de usuarios del sistema, así como obtener la intersección entre categorías de dos usuarios.

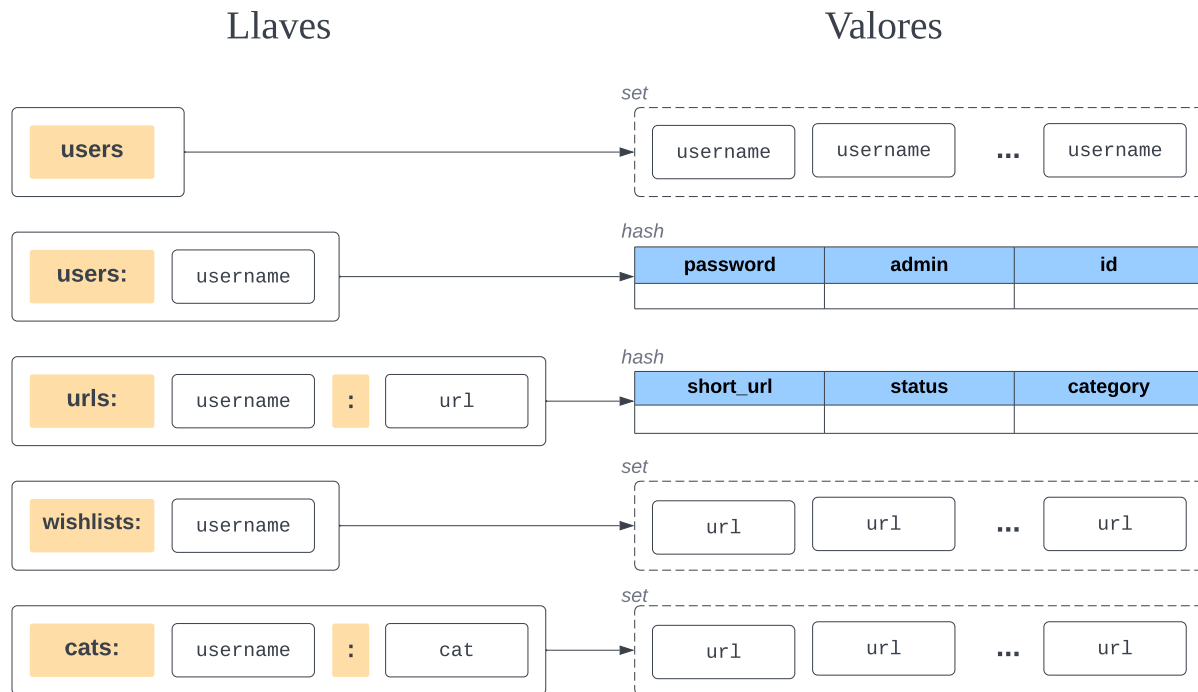


Diseño de base de datos

Dentro de nuestra base de datos conviven 4 elementos principales.

- Usuarios
- URLs por cada usuario
- Wishlist por cada usuario
- Listas de categorías por cada usuario

Estos se organizaron de la forma descrita en el siguiente diagrama:



Usuarios e información de usuario

Los tipos de datos utilizados fueron **sets** y **hashes**. Así, todos los nombres de usuarios están guardados dentro de un conjunto con la llave `users`. Luego, la información del usuario `username` se guarda en la llave `users:username` cuyo valor es un hash, los valores en este son:

password	Contraseña del usuario (encriptada)
admin	Valor 0 o 1 indicando si el usuario es administrador
id	Identificador del usuario

URLs

Cada URL del usuario `username` está guardada con una llave de la forma `urls:username:url` donde `url` es la cadena que corresponde a la URL en cuestión. La información de esta URL correspondiente a este usuario se guarda en un hash que tiene los siguientes atributos:

<code>short_url</code>	URL reducida
<code>status</code>	Valor 0 o 1 indicando si es privada o pública
<code>category</code>	Categoría en la que el usuario clasificó la URL

Wishlists

Cada usuario puede tener una wishlist en la cual guarda aquellas URLs que desea visitar en un futuro. Para el usuario con nombre `username`, su wishlist está guardada como un set bajo la llave `wishlists:username`, en este set se encuentran todas las URL's que el usuario ha añadido a su wishlist.

Categorías

Otra característica del sistema es que cada usuario puede categorizar sus URLs. En el sistema diseñado se manejó una lista fija de categorías: *Arte*, *Ciencia*, *Tecnología*, *Entretenimiento* y *Deportes*. Así, al ingresar una nueva URL al sistema, el usuario puede escoger una categoría a la cual añadirla.

Para poder consultar más rápido la categoría de una URL, se decidió añadir redundancia a esta información, por un lado, la categoría está guardada dentro de la información de la URL en `urls:username:url` y por otro lado, se mantiene un conjunto de las URLs en cada categoría. Este último puede accesarse con la llave `cats:username:category` donde `category` es la cadena que representa la categoría cuyas URLs se busca consultar. El valor guardado en esta llave corresponde a un set que contiene todas las URLs que han sido asignadas a esta categoría.

Implementación en Python

En la implementación dividimos el proyecto en dos etapas, la parte a la que el usuario no debería tener acceso (backend) como implementación de la base con `redis-py` y otra que hace la función de interactuar con el usuario (frontend).

Backend

Para la manipulación de la base de datos en Python creamos una clase denominada `DB` que guarda como atributo un objeto `Redis`. Algunos de los métodos que definimos para poder manipular la base de datos desde la clase `DB` son los siguientes:

1. `resetDataBase`: Borra toda la información de la base de datos por medio de `flushdb`
2. `add_user`: Para identificar a los usuarios requerimos un username, una contraseña y su estado (1: es administrador, 0: no es administrador). Se utilizan los métodos de Redis `scard`, `sadd`, `hset` para agregar la clave del usuario y sus atributos, en caso de que ambos se puedan agregar se procede a regresar el id del usuario en caso contrario se regresa un valor `None`. Similar a este método construimos otros como `add_url`, `add_url_category` para poder ir agregando datos a la base en memoria de Redis.
3. `get_user`: Devuelve la información de un usuario en particular, se localiza a través de su nombre de usuario y se regresan todos sus atributos por medio de `hgetall`. De manera similar se hicieron métodos como `get_users` que por medio de `smembers` regresa todos los usuarios y `get_password` o `is_admin` que obtienen información particular de un usuario por medio de `hget`.

4. `update_url_categories`: También se crearon métodos para poder editar información sobre las urls de cada usuario que usan `hset` para actualizar sus atributos. Se crearon métodos similares para actualizar las categorías o actualizar el estado de una url de publica a privada o viceversa.

Frontend

Después para manipular la información de la base de datos desde el lado del usuario cuentan con diferentes métodos. En este punto requerimos de variables globales que simulan la información actual como el usuario que esta logueado actualmente, las categorías disponibles y un objeto de la clase `DB` que es el que nos da acceso a la base de datos.

Manejo de Usuarios

Los usuarios, pueden loguearse, registrarse o cerrar su sesión. Para simular esta información existe la variable global `Usuario` que guarda la información del usuario actual. Los métodos son consistentes en revisar que la contraseña sea la correcta, que el usuario exista y dependiendo de si es administrador o no tiene acceso a diferentes tipos de menús con los debidos privilegios.

El menú general es el siguiente:

```
Menú
0. Cerrar Programa
1. Registrarse
2. Ingresar
¿Qué accion desea realizar?
```

Cuando un nuevo usuario pide registrarse se le solicita su nombre y una contraseña.

```
Elija un nombre de usuario: Yeudiel
Elija una contraseña:123
```

Después lo regresa al menú principal y puede ingresar, donde si el usuario se registro correctamente debería de poder observar el siguiente menú:

```
Ingrese nombre de usuario:Yeudiel
Ingrese la contraseña:123
Ok, las contraseñas coinciden, usuario logueado
Menú principal
1. URLs
2. Wishlist
3. Categorías
4. Cerrar sesión
Seleccione una opción

```

Un detalle interesante es que para diseñar la base de datos de forma ética con el usuario y por protección de datos no se debe guardar explícitamente la contraseña, para lo cual utilizamos la biblioteca `bcrypt` que por medio de `hash` codifica la entrada original.

Algunas mejoras a considerar son que la contraseña no se vea mientras se escribe o mejorar el aspecto visual, pero a como esta diseñado procura la consistencia en los datos sobre usuarios. Por ejemplo un usuario que no es administrador no podría acceder a los privilegios de estos.

Manejo de URLs

Una vez que un usuario esta logueado puede ingresar nuevas urls y acortarlas como se muestra a continuación:

```
Acciones de URLs
1. Añadir nueva URL y acortarla
2. Administrar privacidad de urls
3. Administrar categorías de urls
4. Regresar al menú principal
¿Qué accion desea realizar?
█
```

Un ejemplo de ejecución es que podemos utilizar el botón de añadir una url y se pide la siguiente información:

```
Introduzca la URL que quiere acortar
https://www.youtube.com/watch?v=r5MR7_INQwg█
```

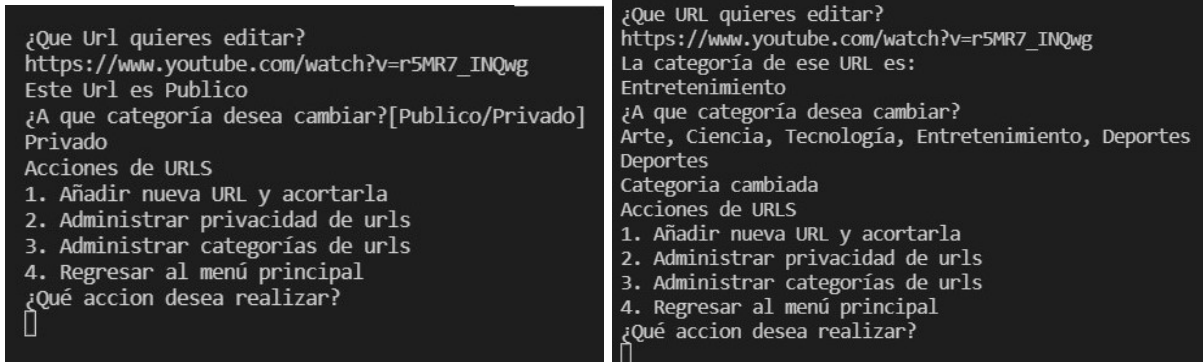
Como podemos observar en la siguiente imagen se guarda si la url es pública o privada y a que categoría pertenece.

```
Introduzca la URL que quiere acortar
https://www.youtube.com/watch?v=r5MR7_INQwg
¿Desea que su URL sea pública? (y/n)
y
Agregue su URL a una categoría
Arte, Ciencia, Tecnología, Entretenimiento, Deportes
Entretinimiento
Debe elegir una categoría válida
Agregue su URL a una categoría
Arte, Ciencia, Tecnología, Entretenimiento, Deportes
Entretinimiento

Su URL acortado es:
https://tinyurl.com/y6xxvxkc
Acciones de URLs
1. Añadir nueva URL y acortarla
2. Administrar privacidad de urls
3. Administrar categorías de urls
4. Regresar al menú principal
¿Qué accion desea realizar?
█
```

Un detalle importante es que las urls acortadas realmente funcionan y esto se logra por medio de **Pyshorteners** es una biblioteca de python que permite acortar urls utilizando las API's más populares de acortamiento de urls disponibles. Además esto garantiza que dadas dos urls no se obtenga la misma reducción.

También podemos modificar si una url es privada o publica como se muestra a continuación:



```

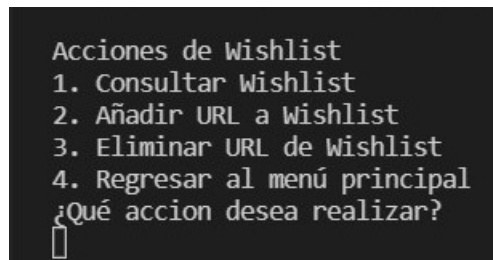
¿Que Url quieres editar?
https://www.youtube.com/watch?v=r5MR7_INQwg
Este Url es Publico
¿A que categoría desea cambiar?[Publico/Privado]
Privado
Acciones de URLS
1. Añadir nueva URL y acortarla
2. Administrar privacidad de urls
3. Administrar categorías de urls
4. Regresar al menú principal
¿Qué acción desea realizar?
1

¿Que URL quieres editar?
https://www.youtube.com/watch?v=r5MR7_INQwg
La categoría de ese URL es:
Entretenimiento
¿A que categoría desea cambiar?
Arte, Ciencia, Tecnología, Entretenimiento, Deportes
Deportes
Categoría cambiada
Acciones de URLS
1. Añadir nueva URL y acortarla
2. Administrar privacidad de urls
3. Administrar categorías de urls
4. Regresar al menú principal
¿Qué acción desea realizar?
1

```

Manejo de la Wishtlist

Una vez que un usuario tiene registradas sus urls puede añadir algunas de esta lista a su wishtlist. Las opciones que tiene disponibles son las siguientes:

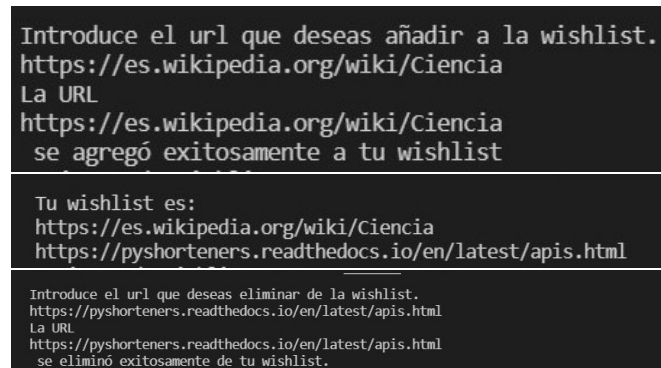


```

Acciones de Wishlist
1. Consultar Wishlist
2. Añadir URL a Wishlist
3. Eliminar URL de Wishlist
4. Regresar al menú principal
¿Qué acción desea realizar?
1

```

A continuación se muestran algunos resultados de utilizar la wishtlist, se pueden agregar urls, mostrar las urls en la wishtlist o eliminar urls de esta misma.



```

Introduce el url que deseas añadir a la wishlist.
https://es.wikipedia.org/wiki/Ciencia
La URL
https://es.wikipedia.org/wiki/Ciencia
se agregó exitosamente a tu wishlist

Tu wishlist es:
https://es.wikipedia.org/wiki/Ciencia
https://pyshorteners.readthedocs.io/en/latest/apis.html

Introduce el url que deseas eliminar de la wishlist.
https://pyshorteners.readthedocs.io/en/latest/apis.html
La URL
https://pyshorteners.readthedocs.io/en/latest/apis.html
se eliminó exitosamente de tu wishlist.

```

Otras opciones de usuario

También cada usuario tiene sus urls organizadas por categorías dentro de la opción de categorías puede consultar todas las urls que ha guardado.

```

Introduce la categoría que deseas consultar
Arte, Ciencia, Tecnología, Entretenimiento, Deportes
Deportes
Las URLs en la categoría Deportes son:
https://www.youtube.com/watch?v=r5MR7_INQwg
Acciones de categorías
1. Consultar categoría
2. Regresar al menú principal
¿Qué acción desea realizar?

```

Opciones de Administrador

Dentro de las opciones de administrador podemos revisar listas de usuarios y también intersecar urls en común por categoría. Aquí se ve el menú de opciones que tiene el administrador:

```

Ingrese nombre de usuario:ADMIN
Ingrese la contraseña:123
Ok, las contraseñas coinciden, usuario logueado
Acciones de administrador:
1. Consultar lista de usuarios
2. Intersecar categorías de dos usuarios
3. Cerrar sesión
Seleccione una opción

```

Y la ejecución de una consulta de los usuarios y una intersección de una categoría entre dos usuarios.

```

La lista de usuarios es la siguiente:
Yeudiel
ADMIN

```

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
Introduce el segundo nombre de usuario
Pablo
Introduce la categoría del usuario Pablo a consultar
Arte, Ciencia, Tecnología, Entretenimiento, Deportes
Ciencia
La intersección de las categorías es:
https://redis.io/topics/persistence#text-by%20default%20redis%20saves%20snapshots,the%20SAVE%20or%20BGSAVE%20commands.&text=This%20strategy%20is%20known%20as%20snapshottng.

```

Conclusiones/Dificultades

El desarrollo del proyecto nos permitió aprender diversas cosas, pues tuvimos que ingeniarnos el cómo resolver distintos problemas. Del mismo modo nos quedamos con ciertas ideas que se podrían implementar a futuro como diseñar una interfaz gráfica o más opciones para los usuarios y administradores.

Una de las principales observaciones que se hicieron durante el desarrollo del proyecto fue la facilidad para manipular datos en la base de datos Redis. Esto debido a que su sintaxis resulta muy simple y concreta, lo que permite abstraer de manera sencilla lo que estamos haciendo.

Por el contrario, una de las dificultades fue precisamente la poca familiaridad con bases de datos llave-valor, pues

nuestra experiencia consistía mayormente en haber trabajado en bases de datos relacionales o convencional. Esto implicó un reto al diseñar la base de datos y al aprender cómo guardar y extraer la información que necesitábamos.

Fue interesante el flujo de trabajo y experimentar un proyecto en donde se usara más de un programa o tecnología al mismo tiempo y se requirieran de múltiples herramientas para su correcto funcionamiento. El desarrollo de este trabajo permitió poner en práctica la teoría vista en clase, más aún, fue una oportunidad para investigar por nuestra cuenta sobre las tecnologías que estábamos utilizando.

Cabe señalar que este proyecto es perfectamente escalable. Algunos elementos que dejamos de lado podrían ser: la posibilidad de que el usuario pueda crear categorías propias además de las preestablecidas y poder asignar cada url a múltiples categorías como si de etiquetas se tratase. Otra característica que puede añadirse en un futuro es la opción de que nuestro usuario pudiera visualizar, a manera de tabla, cada url que ha acertado junto con sus atributos (privacidad y categorías). Además, podría explorarse la posibilidad de cada usuario de consultar la lista de urls públicas, y desplegar dicha información mostrando la url, su versión acertada, el nombre del usuario que la publicó, y la categoría.

A pesar de las dificultades y de los puntos de mejora, consideramos que el proyecto nos permitió aprender mucho y fue una manera interesante de poner en práctica el uso de bases de datos llave valor a un caso de la vida real.