Universidad de los Andes | FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Sistemas Operativos y Redes

Tarea 3, parte 1
Informe

Integrantes:
Ignacio F. Garcés Santander
Francisco J. Jiménez Iglesias

1. *REQUESTS* (CLIENTE)

Forma (basada en HTML):

```
<command>
Host: <URL>
Client port: <client port>
Server port: <server port>
[other tags]
```

No interesa la ruta (como en HTML), pues es una base de datos simplificada, no hay necesidad de usar directorios para el cliente al conectarse al servidor.

command puede ser:

- `connect`: se establece una conexión entre el cliente y el servidor.
- `disconnect`: se cierra conexión.
- `quit`: cierra el programa.
- `insert`: inserta un valor en la BD, especificando llave o no.
- `get`: obtiene el valor de una llave.
- `peek`: verifica si existe una llave en la base de datos.
- `update`: cambia el valor asociado a una llave.
- `delete`: borra una llave de la BD.
- `list`: solicita una lista de todos

other tags son los argumentos de las funciones en la tarea 1. Es decir, sólo se usa en los comandos `insert`, `get`, `peek`, `update`, `delete`. Ver tabla 1.1. En ella, `key` y `value` son datos tipo `int`, y `<null>` significa que no existe `other tags`, es decir, es un string vacío.

command	other tags
connect	<null>
disconnect	<null>
quit	<null>
insert	Key: <key> [Value: <value>]
get	Key: <key>
peek	Key: <key>
update	Key: <key> Value: <value>
delete	Key: <key>
list	<null>

Tabla 1.1: comandos y sus tags asociadas válidas.

2. *RESPONSES* (SERVIDOR)

```
<mensaje de estado> (code: <código de estado>)
Body: <mensaje servidor>
```

Por defecto, `mensaje servidor` es vacío. Sólo si la respuesta debe tener un mensaje (como el comando `list` del cliente), entonces su mensaje es distinto de vacío.

En las tablas 2.1 y 2.2, la columna "Código" es el valor de `código de estado` y "Mensaje de estado" es `mensaje de estado`.

2.1. ESTADOS DE RESPUESTA

Código	Mensaje de estado	Descripción
0	`Connection successful`	Conexión establecida satisfactoriamente entre el servidor y el cliente.
1	`Insert succesful`	La insercion de valor es realizada exitosamente, retorna un int con la clave
2	`Get succesfull`	La obtencion de valor ha sido exitosa, retorna int con el valor asociado a la key enviada
3	`Peek succesfull`	La operación peek ha sido completada, retorna un bool si esta o no en la base de datos
4	`List succesfull`	Accedió y retorna (en `items`) al cliente la lista de pares clave-valor (llave-valor).
5	`Update succesfull`	Logró sobreescribir el valor asociado a una llave.
6	`Delete succesfull`	Par clave-valor eliminado satisfactoriamente.

Tabla 2.1: respuestas de éxito.

Código	Mensaje	Descripción
100	`Connection timeout`	El tiempo para esperar a que el cliente se conecte expiró (el tiempo límite que decía la tarea 1).
101	`Server timeout`	Se agotó el tiempo de respuesta del servidor (15 segundos).
102	`Connection fail`	No se ha podido realizar una conexión con el servidor
103	`Bad access`	Intento de acceder a la BD sin conectarse primero usando el comando `connect`.
104	`Bad key read`	Se intentó leer una llave no existente en la BD.
105	`Bad key write`	Se intentó eliminar una llave no existente en la BD.
105	`Bad key overload`	Se intentó insertar un valor con llave ya existente.
200	`Bad request syntax`	La petición del cliente no está escrita en forma correcta.
201	`Bad tag`	Comando reconocido, pero `other tags` entregados no son válidos para dicho comando
202	`Bad host`	Host inválido
203	`Bad client port`	Error al conectar usando el puerto de cliente especificado
204	`Bad server port`	Error al conectar usando el puerto de servidor especificado

Tabla 2.2: respuestas de excepción.

3. EJEMPLOS

3.1. EJEMPLO 1

Lo que en la tarea 1 era:

Request:

```
peek(5)
```

Response:

```
false
```

Con el protocolo definido, queda:

Request:

```
peek
Host: www.URL_servidor.net
Client port: 1
Server port: 800
Key: 5
```

Response:

```
Peek sucessfull (code: 3)
Body: false
```

3.2. EJEMPLO 2

Request

```
list
Host: www.URL_servidor.net
Client port: 1
Server port: 800
```

Response

```
List sucessfull (code: 4)
Body:
       Key    Value
       1      -4
       5      309
       2      0
       7      134587
```

4. DIAGRAMA

El diagrama de comunicación de ejemplo se ve en la firugra 4.1.

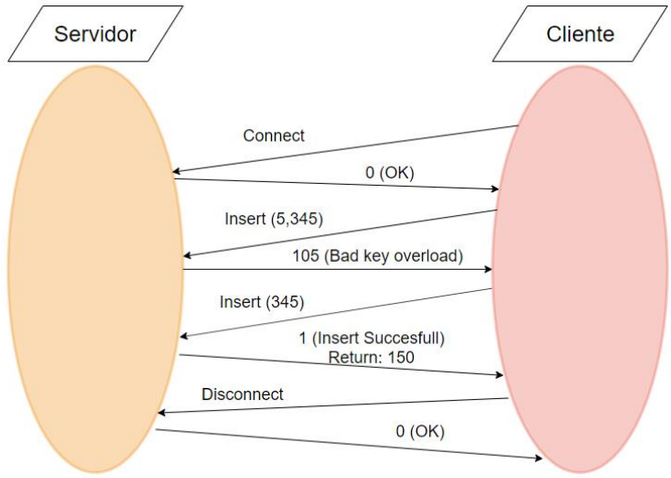

Figura 4.1: diagrama

5. REFERENCIAS

- An overview of HTTP. (2019). Retrieved 1 November 2019, from https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

- Anexo: Códigos de estado HTTP. (2019). Retrieved 5 November 2019, from https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP