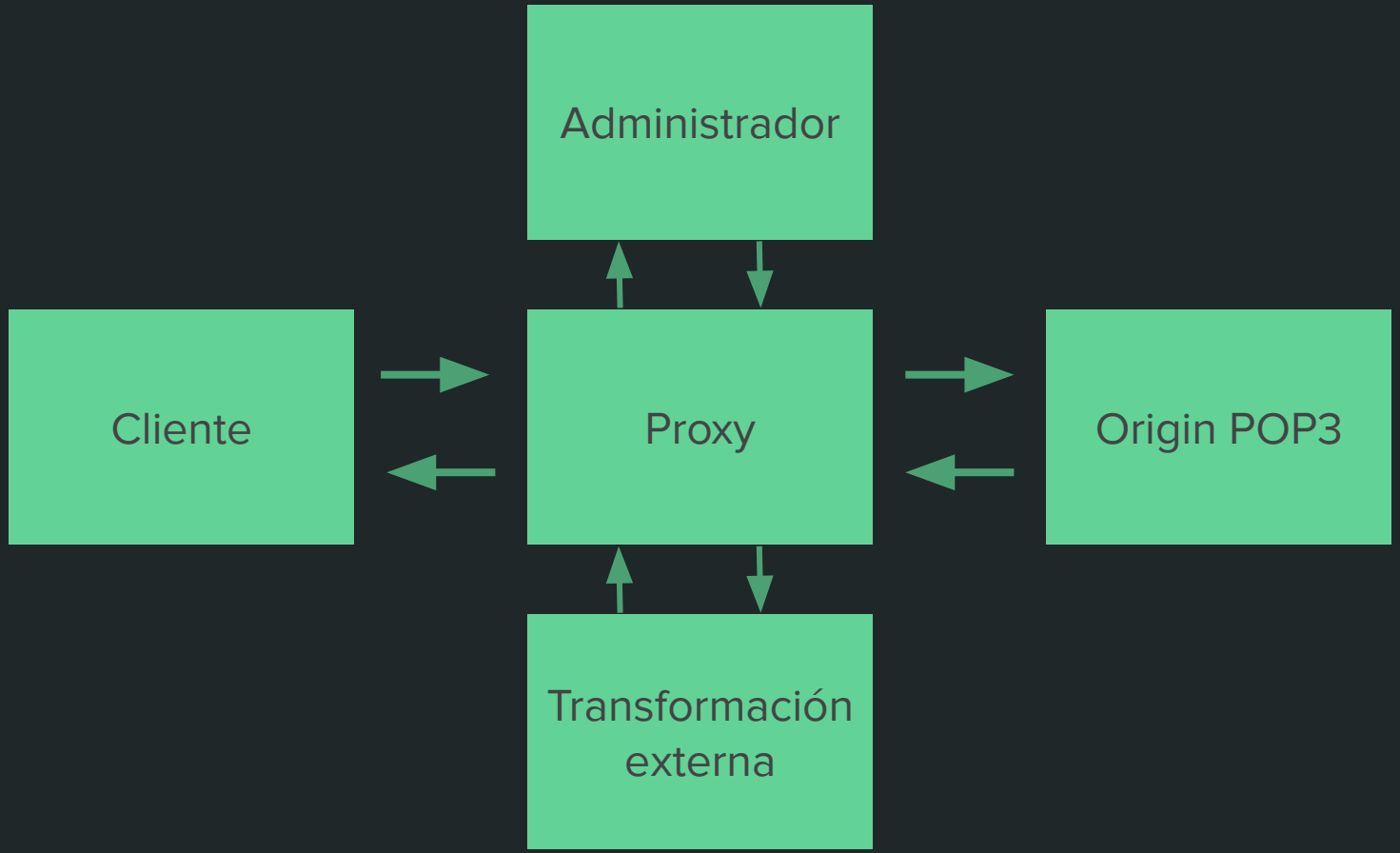
Servidor Proxy POP3

Grupo 6

Battilana, Joaquín
Dallas, Tomás
Dorado, Tomás
Princ, Guido

Diagrama del servidor



Protocolo de administrador

- ***Protocolo binario***
- ***Orientado a conexión***
- ***Orientado a sesión***

Protocolo de administrador

Comandos posibles

- ***get:*** Consulta información según el parámetro de operación solicitado.
- ***set:*** Habilita un media-type a filtrar o una transformación externa.
- ***rm:*** Deshabilita un media-type seteado anteriormente para filtrar.
- ***login:*** Autenticación del admin.
- ***logout:*** Deslogueo de la sesión.

Tipos de operaciones

- ***concurrent***: Se utiliza únicamente con el método get. Devuelve la cantidad de conexiones concurrentes al momento.
- ***accesses:*** Se utiliza únicamente con el método get. Devuelve la cantidad de accesos históricos al momento.
- ***bytes:*** Se utiliza únicamente con el método get. Devuelve la cantidad de bytes transferidos al momento.
- ***mtypes:*** Se utiliza con get/set/rm. Con get, devuelve los media-types que se están filtrando. Con set, setea un media-type para que el proxy lo filtre. Con rm, borra el media-type que se pase por parámetro.
- ***cmd:*** Se utiliza con get/set. Con get, devuelve el comando con el cual se están haciendo las transformaciones actualmente. Con set, setea la transformación pasada por parámetro. (Por default, cmd = cat)

Identificación de comandos por bytes *(commandCode)*

Todos los comandos son identificados utilizando 1 byte:

- ***login: 0x00***
- ***logout: 0x01***
- ***get: 0x02***
- ***set: 0x03***
- ***rm: 0x04***

Formato de pedidos y respuestas de login y logout

- ***request = login <token>***
- ***response = status***
- ***Cantidad de bytes totales request = 1 byte*** *(login)* ***+ 8 bytes*** *(token)*
- ***Cantidad de bytes totales response = 1 byte*** *(status)*
- ***request = logout***
- ***response = status***
- ***Cantidad de bytes para ambos = 1 byte***

Identificación de operaciones por bytes *(opCode)*

Todas las operaciones son identificadas utilizando 1 byte:

- ***concurrent: 0x01***
- ***accesses: 0x02***
- ***bytes: 0x03***
- ***mtypes: 0x04***
- ***cmd: 0x05***

Formato de pedidos y respuesta de métricas

- ***request = get metric***
- ***response = status dataResponse***
- ***Cantidad de bytes totales request = 2 bytes*** *(get: 1 byte, metric: 1 byte)*
- ***Cantidad de bytes totales response = 1 byte*** *(status)* ***+ 4 Bytes*** *(dataResponse)*

concurrent 0x01

- *Request:*
  - *get concurrent*
- *Response:*
  - *OK <concurrentResponse>*
  - *concurrentResponse: 4 bytes*

accesses 0x02

- *Request:*
  - *get accesses*
- *Response:*
  - *OK <accessesResponse>*
  - *accessesResponse: 4 bytes*

bytes 0x03

- *Request:*
  - *get bytes*
- *Response:*
  - *OK <bytesResponse>*
  - *bytesResponse: 4 bytes*

Formato de pedidos de Media Types

- ***request = commandCode opCode mtypesQty [mtypes]***
- ***(En [mtypes] entre cada parámetro hay un ‘\0’)***
- ***Cantidad de bytes totales request = 4 bytes*** *(commandCode: 1 byte, opCode: 1 byte, mtypesQty: 2 bytes)* ***+ n bytes*** *(mtypes)*

Formato de respuesta de Media Types

- ***response = opCodeStatus mtypesQty [mtypes]***
- ***(En [params] entre cada parámetro hay un ‘\0’)***
- ***Cantidad de bytes totales response = 1 byte*** *(opCode)* ***+ 2 bytes*** *paramsQty: 2 bytes +* ***n Bytes*** *(params)*

mtypes 0x04

- *Requests:*
  - *get mtypes*
  - *set mtypes paramsQty [params]*
  - *rm mtypes paramsQty [params]*
- *Responses:*
  - *get mtypes → OK mtypesQty [mtypes]*
  - *set mtypes paramsQty [params] → OK/ERROR qty [mtypes]*
  - *rm mtypes paramsQty [params] → OK/ERROR qty [mtypes]*

Formato de pedidos de Cmd

- ***request = commandCode opCode <cmd>***
- ***cmd está finalizado con un \0 y solo va en set***
- ***Cantidad de bytes totales request = 2 bytes*** *(commandCode: 1 byte, opCode: 1 byte)* ***+ n bytes*** *(cmd)*

Formato de respuesta de Cmd

- ***response = status cmd***
- ***(El cmd está finalizado con un '\0')***
- ***Cantidad de bytes totales response = 1 byte*** *(status) +* ***n Bytes*** *(cmd)*

cmd 0x05

- ***Requests:***
  - ***get cmd***
  - ***set cmd <cmd>***
- ***Responses:***
  - ***get cmd → OK <cmdResponse>***
  - ***set cmd <cmd> → OK/ERROR***

El comando por default es *cat*.

En el get al final del cmd y cmdReponse hay un \0

MIME logic

Para la la lógica que se va a usar para el parseo de los body se tuvo de base el RFC 2045 y el RFC 2046

Headers parser

Primero se van a parsear los headers provenientes del paquete que viene del servidor POP3 para poder tener la versión de MIME utilizada y el tipo de contenido del body

En caso de no contener o que sean inválidos los headers MIME-version y Content-type se van a utilizar tipos default
Mime-version -> 1.0
Content-type -> text/plain; charset: US-ASCII

RFC 2045

Conjunto de headers que describen la estructura de los mensajes del MIME:

- **Versión MIME**: distinguir mensajes generados por softwares viejos o no conformantes.
- **Content-Type**: especifica el tipo y subtipo de la data del body.
- **Content-Transfer-Encoding**: especifica la transformación realizada al body y el dominio del resultado.
- **Content-ID y Content-Description**: describen aún más la data del body.

RFC 2046

Describe la estructura general del sistema de media types del MIME y define un set initial de media types:

1. Texto
2. Imagen
3. Audio
4. Video
5. Aplicación (otro tipo de data)
6. Multiparte
7. Mensaje

Body parser

Una vez que tengamos los headers ya parseados e identificados, en el caso de que el primary type del header Content-type no sea multipart se va a mandar el body a la transformación y luego se va a continuar con el flujo normal. En el caso que si sea, se va a mandar a parsear el body para identificar cada Content-type dentro del body dentro de los boundaries (Lo que este afuera va a ser identificado como text/plain). Luego de tener identificado cada sub-contenido con su subtipo van a ser enviados a cada transformación correspondiente y se va a continuar con el flujo normal.

En el caso que haya subcontenidos inválidos van a ser procesados como texto plano. Se llegó a esta decisión evaluando que es posible que el mismo paquete que es enviado a nuestro proxy podría ser enviado a otro que si acepte estos subcontenidos, por lo que es mejor enviarlo igual que como llegó a que se envíe un error.

Diagrama de flujo de lógica MIME