

Práctica 1



UGR

Desarrollo de Software

Autores:

- Alba Ayala Carmona
- María Ramos Martínez
- Sofía González Uceda
- Natalia Serrano Cerceda

Índice

1. Ejercicio 1	2
1.1. Explicación	2
1.2. UML	2
1.3. Resultados de la ejecución	3
2. Ejercicio 2	4
2.1. Explicación	4
2.2. UML	4
2.3. Resultados de la ejecución	5
3. Ejercicio 3	6
3.1. Explicación	6
3.2. UML	7
3.3. Resultados de la ejecución	7
4. Ejercicio 4	9
4.1. Explicación	9
4.2. UML	9
4.3. Resultados de la ejecución	10
5. Compilación y Ejecución	10

1. Ejercicio 1

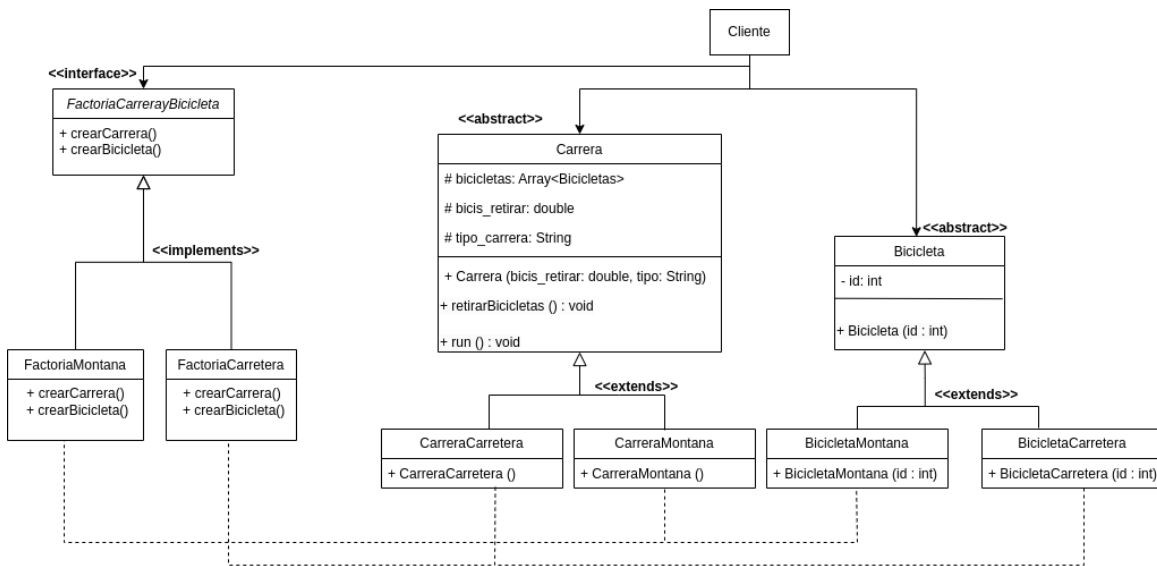
1.1. Explicación

El ejercicio 1 consiste en simular dos carreras de bicicletas simultáneas con el mismo número aleatorio de bicicletas. Para ello se han implementado los patrones de diseño Factoría Abstracta y Método Factoría.

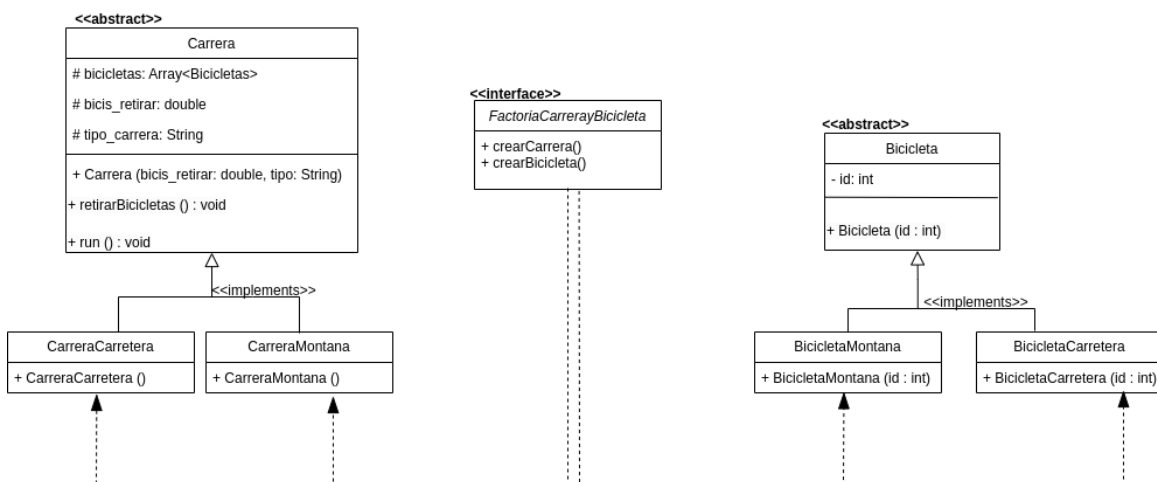
La clase Carrera implementa la interfaz *Runnable* de Java, que nos permite ejecutar procesos en paralelo y así simular la concurrencia de las carreras. Esta interfaz tiene un único método: *run*. Este método lo implementa obligatoriamente la clase Carrera y es el que va a ejecutar cada hebra.

En la ejecución se muestran, al inicio de la carrera, cuántas bicicletas hay; mientras que están corriendo, se muestra un mensaje cada 10 segundos indicando que aún están corriendo junto con el número de bicicletas. Tras pasar 60 segundos se acaban las dos carreras y se muestra un mensaje de finalización junto con el número de bicicletas que queda en cada carrera.

1.2. UML



(a) Patrón Abstract Factory



(b) Patrón Factory Method

1.3. Resultados de la ejecución

```
Iniciando Carrera de Montaña con 292 bicicletas.  
Iniciando Carrera de Carretera con 292 bicicletas.  
    Carrera de Carretera corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Carretera corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Carretera corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Carretera corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Carretera corriendo. 292 bicicletas  
    Carrera de Montaña corriendo. 292 bicicletas  
    Carrera de Carretera corriendo. 292 bicicletas  
Carrera de Montaña finalizada. Bicicletas restantes: 234  
Carrera de Carretera finalizada. Bicicletas restantes: 263  
Todas las carreras han finalizado.
```

Figura 2: Ejecución ejercicio 1

2. Ejercicio 2

2.1. Explicación

El ejercicio 2 consiste en interactuar con modelos de lenguaje usando la [API de Hugging Face](#). Para ello es necesario tener un *token* de *Hugging Face*. Este *token* se pedirá para la ejecución del programa (necesario para los LLM). Además, contamos con un archivo de configuración (por defecto es *config.json*) que servirá para especificar los parámetros necesarios en los constructores (los modelos a utilizar y los idiomas para la traducción) y el texto necesario para llamar a la función *generate_summary*. Para poder trabajar con este tipo de archivos necesitamos incluir el módulo *json*.

Hemos implementado el patrón decorador teniendo un LLM básico y, como decoradores, uno de traducción y otro de expansión. El LLM básico es un modelo que resume el texto que se le pasa. Los LLM decoradores (traducción y expansión) primero mandan el texto a procesar por el LLM que tengan asociado (especificado en el constructor) y luego lo decoran (traduciéndolo o expandiéndolo, según corresponda). Para poder realizar las peticiones a la API, necesitamos incluir el módulo *requests*, instalándolo de la siguiente forma:

```
pip install request
```

2.2. UML

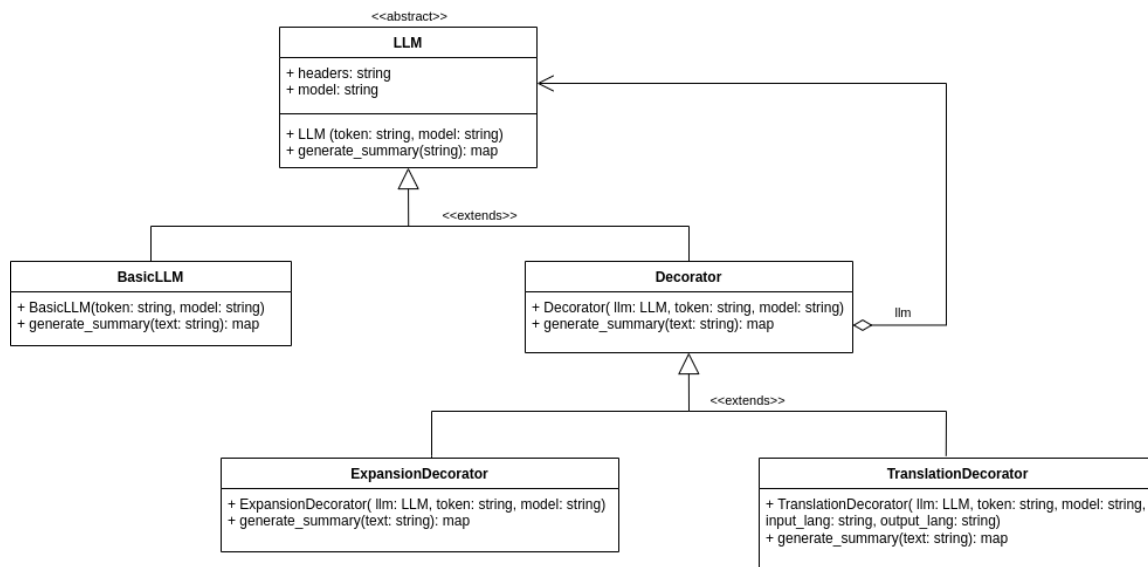


Figura 3: UML ejercicio 2

2.3. Resultados de la ejecución

Al iniciar el programa, aparece el siguiente menú, solicitando el token.

```
• sofia@sofia-IdeaPad-3-15ITL6:~/DS_Practica/Practical1/Ej2$ python3 main.py
Es necesario introducir un token para conectarse con Hugging Face
  Para introducir manualmente el token presione [m]
  Para leer el token de un fichero .txt presione cualquier otra tecla

q
  Para introducir manualmente el nombre del archivo presione [a]
  Para leer el token del fichero 'token.txt' presione cualquier otra tecla

q

Se va a leer 'config.json' como archivo de configuración.
```

Figura 4: Ejecución ejercicio 2

Una vez se ha introducido el *token*, se llama a los modelos de lenguaje. Un ejemplo de ejecución podría ser el siguiente.

```
-----
El archivo de configuración es el siguiente:
{'texto': 'Cats are fascinating animals, known for their independence and agility. Although they are often considered so
litary, many cats form strong bonds with their owners, showing affection in their own way, such as rubbing against them
or purring. Their curious nature leads them to explore their surroundings with great skill, using their exceptional sens
es, like night vision and keen hearing, to hunt or simply investigate. Additionally, their ability to jump and move with
agility makes them surprising and graceful animals, ideal both for companionship and admiration.', 'input_lang': 'en',
'output_lang': 'es', 'model_llm': 'facebook/bart-large-cnn', 'model_translation': 'Helsinki-NLP/opus-mt-input-output',
'model_expansion': 'facebook/blenderbot-400M-distill'}

Texto original:
'Cats are fascinating animals, known for their independence and agility. Although they are often considered solitary, ma
ny cats form strong bonds with their owners, showing affection in their own way, such as rubbing against them or purring
. Their curious nature leads them to explore their surroundings with great skill, using their exceptional senses, like n
ight vision and keen hearing, to hunt or simply investigate. Additionally, their ability to jump and move with agility m
akes them surprising and graceful animals, ideal both for companionship and admiration.'

Texto resumido
[{'summary_text': 'Cats are fascinating animals, known for their independence and agility. Many cats form strong bonds w
ith their owners, showing affection in their own way, such as rubbing against them or purring. Their curious nature lead
s them to explore their surroundings with great skill, using their exceptional senses.'}]

Texto traducido:
[{'translation_text': 'Los gatos son animales fascinantes, conocidos por su independencia y agilidad. Muchos gatos forma
n fuertes lazos con sus dueños, mostrando afecto a su manera, como frotarse contra ellos o ronronear. Su curiosa natural
eza los lleva a explorar su entorno con gran habilidad, utilizando sus sentidos excepcionales.'}]

Texto extendido:
[{'generated_text': ' I agree. Cats have a great sense of smell and can see in near darkness.'}]
sofia@sofia-IdeaPad-3-15ITL6:~/DS_Practica/Practical1/Ej2$
```

Figura 5: Ejecución ejercicio 2

3. Ejercicio 3

3.1. Explicación

En el ejercicio 3 se pide un patrón estrategia, con dos estrategias, que busque el texto, el autor y las etiquetas de <https://quotes.toscrape.com/>. Para ello hemos creado 1 interfaz (*Strategy*) y 3 clases (*BeaSoup*, *Selen*, *Contex*).

En *BeautifulSoupStra.py* hemos implementado la estrategia *BeautifulSoup*, para extraer la información con *BeautifulSoup* y *request*. Para poder utilizarlas hemos tenido que instalar *beautifulsoup4* y *request*:

```
pip install beautifulsoup4
```

```
pip install request
```

En la función *extraer_datos* a la que le pasamos la URL y el número de páginas hemos hecho un bucle que va actualizando la URL para pasar a la siguiente página. Con las funciones *find* y *find_all* buscamos los elementos que nos interesan. En este caso hemos buscado primero todos los *div* que contienen la información que buscamos para, a continuación, dentro de cada uno buscar la cita, el autor y las etiquetas de la página. Además, tenemos en cuenta que el autor no se repita. Si el autor aún no se ha añadido, se añade; en caso contrario, se añade la nueva cita (con las etiquetas) a su lista. Finalmente, devolvemos los datos que contienen la información.

En *SeleniumStra.py* hemos implementado la estrategia *Selenium*, para ello hemos tenido que descargar:

```
pip install selenium
```

```
pip install webdriver-manager
```

Nuestro programa lo hemos implementado para los navegadores Firefox y Chrome. En el caso de querer probarlo con Firefox debemos tener instalado el *geckodriver* (<https://github.com/mozilla/geckodriver/releases/>).

En *Selenium* hemos implementado un constructor al que le pasamos como parámetros el navegador, y un método *extraer_datos* en el que tendremos en cuenta el tipo de navegador para crear el *driver*. Para Firefox, con ayuda de la terminal, obtendremos la ruta de donde se encuentra el *geckodriver*, y para Chrome, comprobamos que esté la última versión instalada con *ChromeDriverManager().install*.

Una vez que tenemos el driver, abrimos el navegador y buscamos los elementos que queremos con *find_elements* y la biblioteca *BY*, que nos permite buscar por clase, contenido... Aquí hacemos lo mismo que en *BeautifulSoup* con *find_all*, y los almacenamos de la misma forma, comprobando también que los autores no se repitan. Para pasar a la siguiente página tenemos que buscar el botón correspondiente que, en este caso, es un enlace, y pulsarlo. Para ello, buscamos un enlace que contenga el texto *Next* y le indicamos que haga click en ese botón.

El fichero *Context.py* contiene la clase *Context* en la que hacemos uso de alguna de las estrategias. Para ello, tenemos un constructor al que le pasamos un objeto tipo *Strategy*, y un método que extrae los datos con esa estrategia, que hemos llamado *ejecutar*. En esta clase también tenemos un método que nos guarda los datos obtenidos en un fichero *yaml*, para ello hemos usado la biblioteca *yaml*.

Finalmente, tenemos un fichero *main.py* donde se nos pregunta que estrategia queremos escoger y dependiendo de esta creamos el objeto *Context* y lo ejecutamos. En el caso de *Selenium* también pregunta por el navegador que queremos usar. Por último, guardamos la información en el fichero que el usuario ha elegido. En el caso de que el nombre no termine en *.yaml*, se escogerá otro el nombre de fichero por defecto (*archivo.yaml*)

3.2. UML

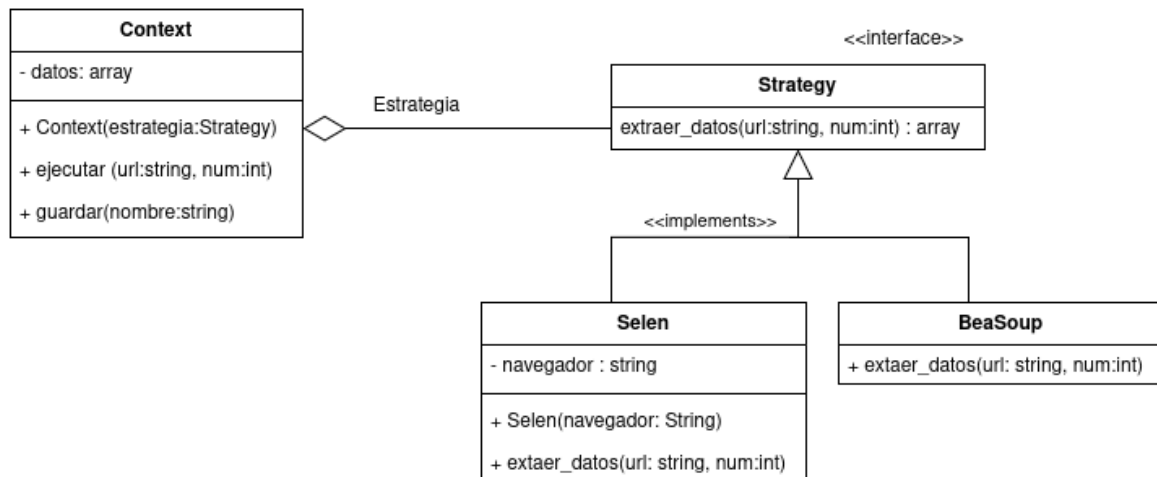


Figura 6: UML Ejercicio 3

3.3. Resultados de la ejecución

Si elegimos la opción de BeautifulSoup y ponemos un nombre de fichero correcto:

```
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA: ~/Escritorio/DS/DS_Practica
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$ python3 Practica1/Ej3/main.py
Elige un metodo:
  1 --> BeautifulSoup
  2 --> Selenium
1
Obteniendo información con BeautifulSoup ...
Nombre del fichero (que termine en .yaml, sino se escogera otro nombre): beasoup.yaml
Nombre aceptado
Guardando la información...
Información guardada con éxito!!
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$
```

Figura 7

Si ponemos opciones erróneas, y luego elegimos Selenium, Firefox y un nombre de fichero correcto:

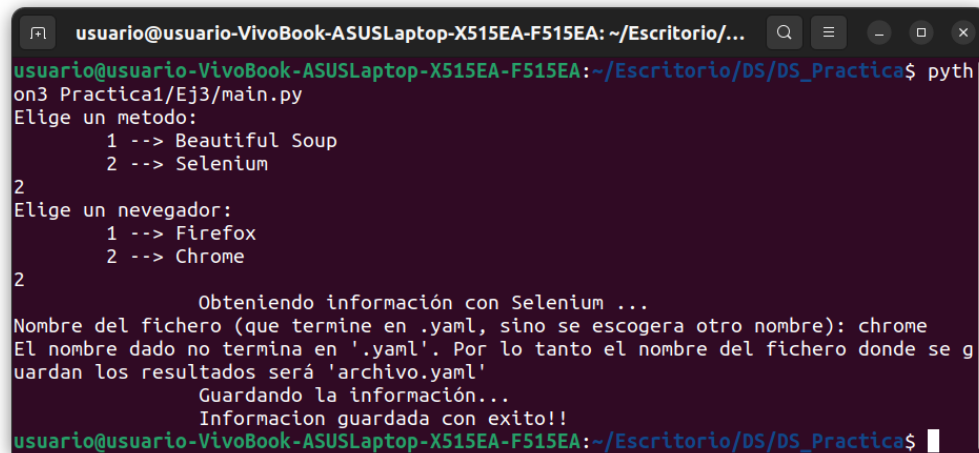
```
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA: ~/Escritorio/DS/DS_Practica
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$ python3 Practica1/Ej3/main.py
Elige un metodo:
  1 --> BeautifulSoup
  2 --> Selenium
prueba
Por favor, ingrese un número válido.

Elige un metodo:
  1 --> BeautifulSoup
  2 --> Selenium
2
Elige un navegador:
  1 --> Firefox
  2 --> Chrome
a
Por favor, ingrese un número válido.

Elige un navegador:
  1 --> Firefox
  2 --> Chrome
1
Obteniendo información con Selenium ...
Nombre del fichero (que termine en .yaml, sino se escogera otro nombre): fire.yaml
Nombre aceptado
Guardando la información...
Información guardada con éxito!!
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$
```

Figura 8

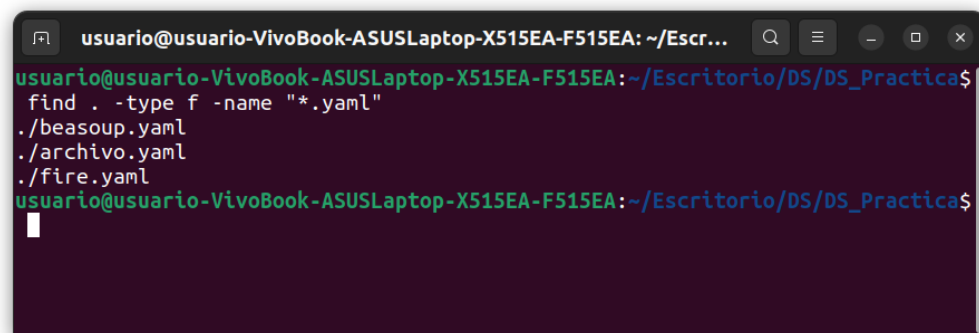
Si elegimos Selenium, Chrome y ponemos mal el nombre del fichero:



```
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA: ~/Escritorio/...
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$ python3 Practica1/Ej3/main.py
Elige un metodo:
    1 --> BeautifulSoup
    2 --> Selenium
2
Elige un navegador:
    1 --> Firefox
    2 --> Chrome
2
Obteniendo información con Selenium ...
Nombre del fichero (que termine en .yaml, sino se escogera otro nombre): chrome
El nombre dado no termina en '.yaml'. Por lo tanto el nombre del fichero donde se guardan los resultados será 'archivo.yaml'
Guardando la información...
Información guardada con éxito!!
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$
```

Figura 9

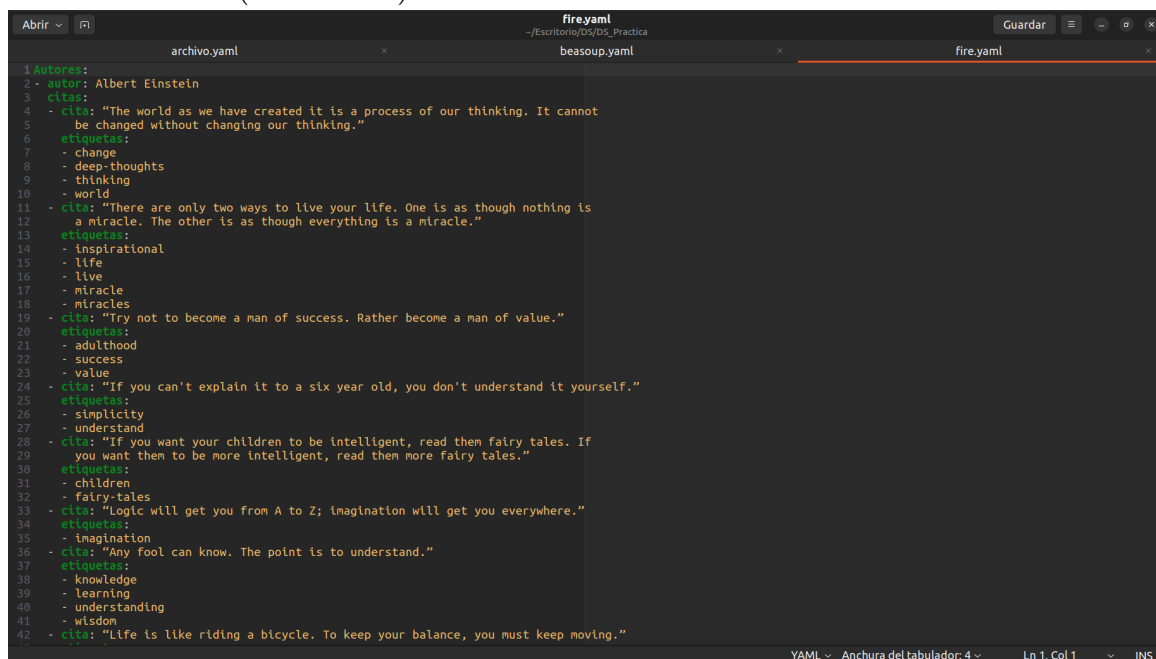
Resultados de los ficheros:



```
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA: ~/Escritorio/DS/DS_Practica$ find . -type f -name "*.yaml"
./beasoup.yaml
./archivo.yaml
./fire.yaml
usuario@usuario-VivoBook-ASUSLaptop-X515EA-F515EA:~/Escritorio/DS/DS_Practica$
```

Figura 10

Contenido de los ficheros (son el mismo):



```
Abrir  fire.yaml
~/Escritorio/DS/DS_Practica
Guardar
archivo.yaml  beasoup.yaml  fire.yaml

1  Autor:
2  - autor: Albert Einstein
3  citas:
4  - cita: "The world as we have created it is a process of our thinking. It cannot
5    be changed without changing our thinking."
6  etiquetas:
7  - change
8  - deep-thoughts
9  - thinking
10 - world
11 - cita: "There are only two ways to live your life. One is as though nothing is
12   a miracle. The other is as though everything is a miracle."
13 etiquetas:
14 - inspirational
15 - life
16 - live
17 - miracle
18 - miracles
19 - cita: "Try not to become a man of success. Rather become a man of value."
20 etiquetas:
21 - adulthood
22 - success
23 - value
24 - cita: "If you can't explain it to a six year old, you don't understand it yourself."
25 etiquetas:
26 - simplicity
27 - understand
28 - cita: "If you want your children to be intelligent, read them fairy tales. If
29   you want them to be more intelligent, read them more fairy tales."
30 etiquetas:
31 - children
32 - fairy-tales
33 - cita: "Logic will get you from A to Z; imagination will get you everywhere."
34 etiquetas:
35 - imagination
36 - cita: "Any fool can know. The point is to understand."
37 etiquetas:
38 - knowledge
39 - learning
40 - understanding
41 - wisdom
42 - cita: "Life is like riding a bicycle. To keep your balance, you must keep moving."
```

Figura 11

4. Ejercicio 4

4.1. Explicación

El ejercicio 4 trata sobre la creación de una cuenta de usuario validando el usuario y la contraseña. Para la validación se pasa la cuenta por una serie de filtros. En cada filtro, se prueba una condición y de no cumplirse muestra un error por pantalla y no comprueba el resto de filtros. Si el correo y la contraseña pasan todos los filtros se muestra al usuario una confirmación de la validación de su nueva cuenta.

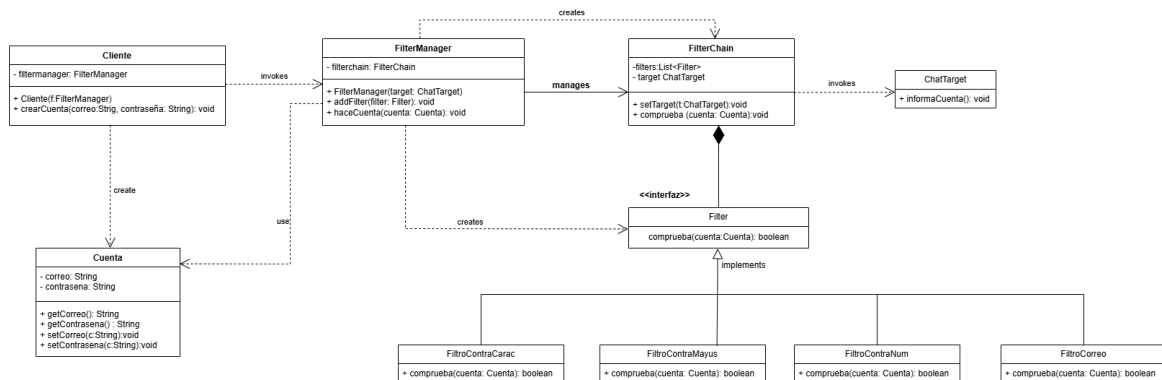
Para implementar esta estructura tenía que usarse el patrón de filtros de intercepción. En el proceso de ejecución lo primero que se hace, es pedir al usuario por pantalla una dirección de correo y una contraseña, informándole de las características necesarias para que sean válidos. Hemos incorporado que, al introducir la contraseña no se muestre por pantalla.

Tras introducir los datos, se crea un *Cliente* con la información proporcionada y llamamos a un método que le crea una cuenta y la manda al *gestor de filtros*. El *gestor de filtros*, la manda a la *cadena de filtros*, donde se comprueban todos los filtros. Si todos los filtros salen correctos, el *target* muestra un mensaje que confirma la correcta configuración de la cuenta. Si alguno de los filtros falla, se muestra un mensaje de error informando de que filtro ha fallado y no comprueba los filtros restantes.

Los filtros que hemos utilizado en la contraseña son:

- Que tenga al menos 8 caracteres
- Que al menos uno de los caracteres sea una mayúscula
- Que al menos uno de los caracteres sea numérico

4.2. UML



4.3. Resultados de la ejecución

```
PROGRAMA CREADOR DE USUARIOS
Formato del correo: <nombre>@<dominio>
Siendo el dominio: gmail.com o hotmail.com
La contraseña debe tener al menos 8 caracteres, siendo al menos uno de ellos una mayúscula y otro un número

Introduzca una dirección de correo: correo@yahoo.com
Introduzca una contraseña:
X Error: No es un dominio de correo válido
```

(a) Ejemplo de error en el correo

```
PROGRAMA CREADOR DE USUARIOS
Formato del correo: <nombre>@<dominio>
Siendo el dominio: gmail.com o hotmail.com
La contraseña debe tener al menos 8 caracteres, siendo al menos uno de ellos una mayúscula y otro un número

Introduzca una dirección de correo: correo@gmail.com
Introduzca una contraseña:
X Error: La contraseña debe contener al menos un caracter numérico
```

(b) Ejemplo de error en la contraseña

```
PROGRAMA CREADOR DE USUARIOS
Formato del correo: <nombre>@<dominio>
Siendo el dominio: gmail.com o hotmail.com
La contraseña debe tener al menos 8 caracteres, siendo al menos uno de ellos una mayúscula y otro un número

Introduzca una dirección de correo: correo@gmail.com
Introduzca una contraseña:
✓ Cuenta configurada con éxito
```

(c) Ejemplo de configuración correcta

5. Compilación y Ejecución

- **Ejercicio 1:** La compilación se hace con el *Makefile*. Para ejecutarlo: *java Main*
- **Ejercicio 2:** Para ejecutarlo: *python3 main.py*
- **Ejercicio 3:** Para ejecutarlo: *python3 main.py*
- **Ejercicio 4:** La compilación se hace con el *Makefile*. Para ejecutarlo: *java Main*