

# DESARROLLO DE SOFTWARE

## PRÁCTICA 2

Luis Alberto Mejía Troya  
Antonio Pancorbo Morales  
Manuel Jesús Junquera Lobón  
17 de Abril del 2025



# UNIVERSIDAD DE GRANADA

# Índice general

|   |          |
|---|----------|
| <b>1. Mantenimiento en Flutter</b>                | <b>1</b> |
| 1.1. Descripción . . . . .                        | 1        |
| 1.2. Mantenimiento adaptativo . . . . .           | 1        |
| 1.3. Mantenimiento perfecto . . . . .             | 1        |
| 1.4. Mantenimiento preventivo . . . . .           | 1        |
| 1.5. Sistema de notificaciones . . . . .          | 1        |
| 1.6. Ejemplos de ejecución . . . . .              | 2        |
| 1.7. Esquema UML . . . . .                        | 5        |
| <b>2. Ejercicio grupal opcional</b>               | <b>6</b> |
| 2.1. Descripción . . . . .                        | 6        |
| 2.2. Conexión a la API . . . . .                  | 6        |
| 2.3. Implementación del patrón Strategy . . . . . | 6        |
| 2.4. Ejemplos de ejecución . . . . .              | 7        |
| 2.5. Esquema UML . . . . .                        | 8        |

# Capítulo 1

## Mantenimiento en Flutter

### 1.1. Descripción

Realiza el mantenimiento adaptativo y perfectivo de la actividad del Patrón Filtros de Intercepción (email+pass) de la Práctica 1. Requisitos:

- Diseño e implementación en Flutter + dart (adaptativo).
- Mejora de los filtros (perfectivo).
- Nueva funcionalidad donde se compruebe si el email ha sido creado previamente (perfectivo/preventivo).
- Agregar un sistema de notificaciones que informe al usuario sobre qué filtro fue rechazado.

### 1.2. Mantenimiento adaptativo

La realización del mantenimiento adaptativo ha sido implementar el código al lenguaje `dart`. Además, hemos dividido en varios módulos y hemos integrado las clases necesarias para introducir el patrón `Filter`.

### 1.3. Mantenimiento perfectivo

Hemos añadido un nuevo filtro donde comprobamos si la contraseña tiene un número.

### 1.4. Mantenimiento preventivo

Para la realización de este mantenimiento y ya que no contamos con una base de datos, hemos creado una lista donde vamos guardando los emails que han sido válidos. De esta forma, si el usuario introduce un email existente, le saldrá un mensaje ya que este email ya está registrado.

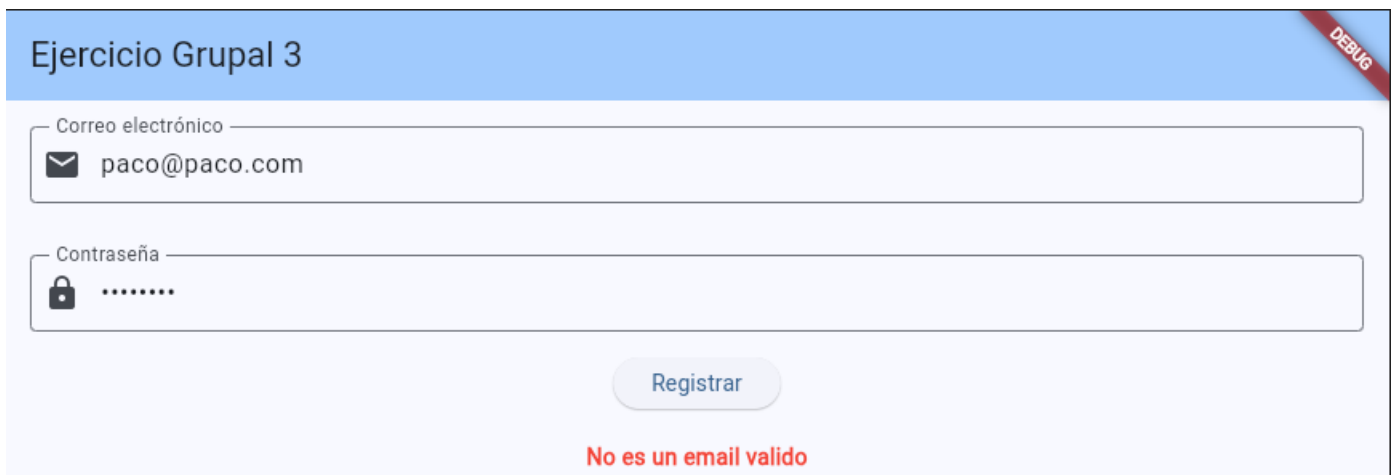
### 1.5. Sistema de notificaciones

Para agregar el sistema de notificaciones, hemos modificado la implementación original y hemos hecho que devuelva un objeto que tendrá:

- Descripción
- Valor booleano

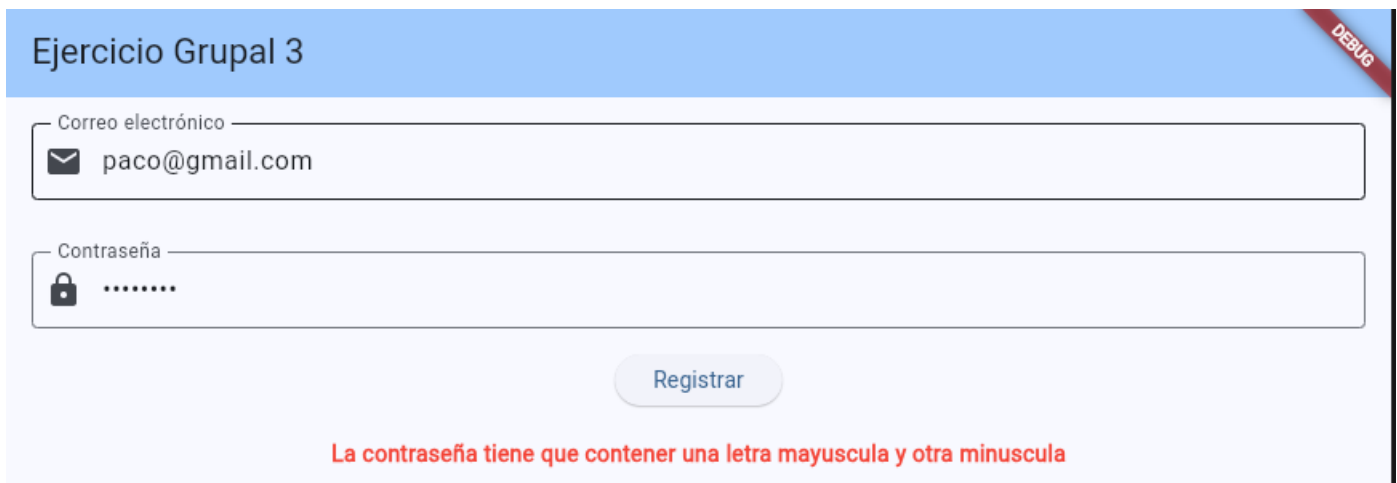
De esta forma, podemos comprobar si pasa los filtros y, en caso contrario, nos informará sobre el error que ha encontrado.

## 1.6. Ejemplos de ejecución



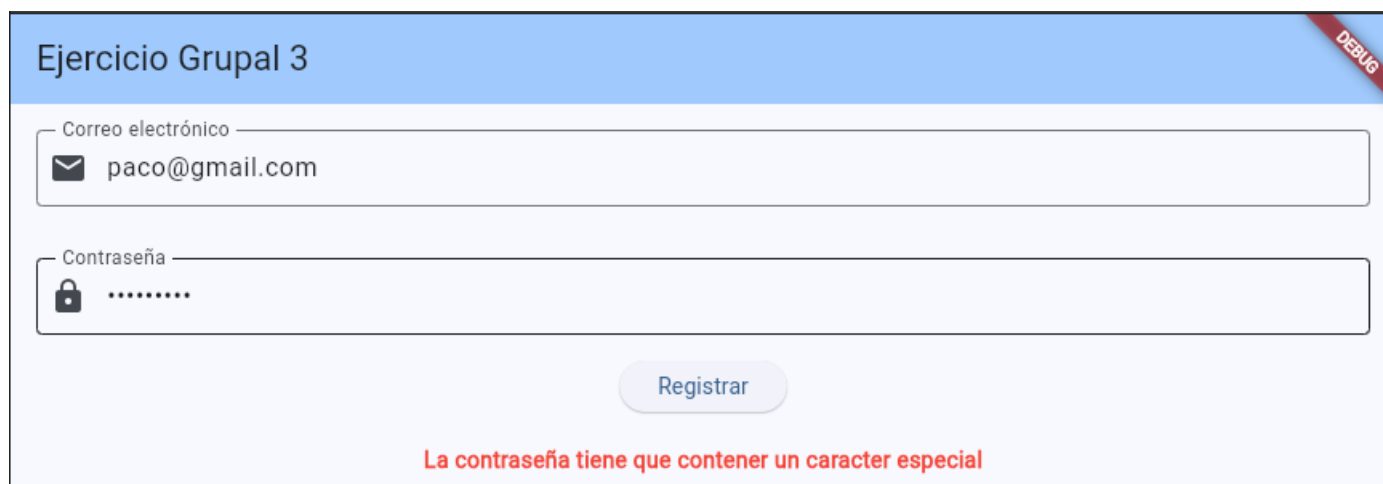
The screenshot shows a Flutter app interface titled "Ejercicio Grupal 3" in a blue header bar. A red "DEBUG" banner is in the top right corner. Below the header, there are two input fields: "Correo electrónico" (Email) and "Contraseña" (Password). The email field contains "paco@paco.com" and has an envelope icon. The password field contains seven dots and has a lock icon. Below the fields is a "Registrar" button. At the bottom, a red error message reads "No es un email valido".

Figura 1.1: Ejemplo de e-mail no válido



The screenshot shows the same Flutter app interface as Figure 1.1. The email field now contains "paco@gmail.com". The password field still contains seven dots. The "Registrar" button is still present. At the bottom, a red error message reads "La contraseña tiene que contener una letra mayuscula y otra minuscula".

Figura 1.2: Ejemplo de contraseña sin mayúsculas



Ejercicio Grupal 3

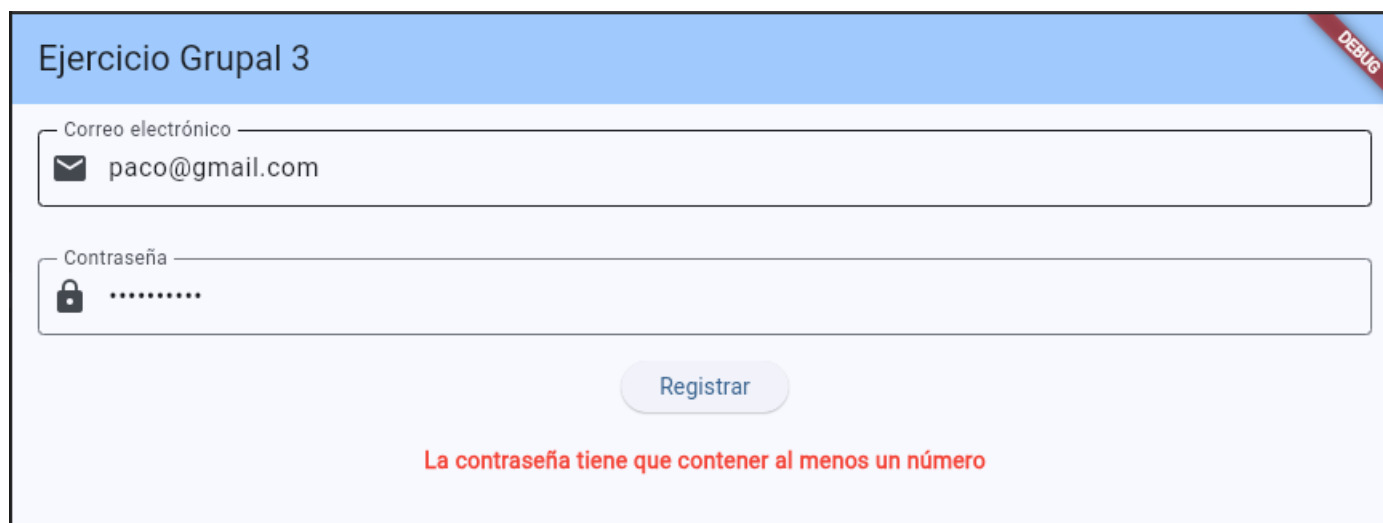
Correo electrónico —  
✉ paco@gmail.com

Contraseña —  
🔒 .....

Registrar

La contraseña tiene que contener un caracter especial

Figura 1.3: Ejemplo de contraseña sin carácter especial



Ejercicio Grupal 3

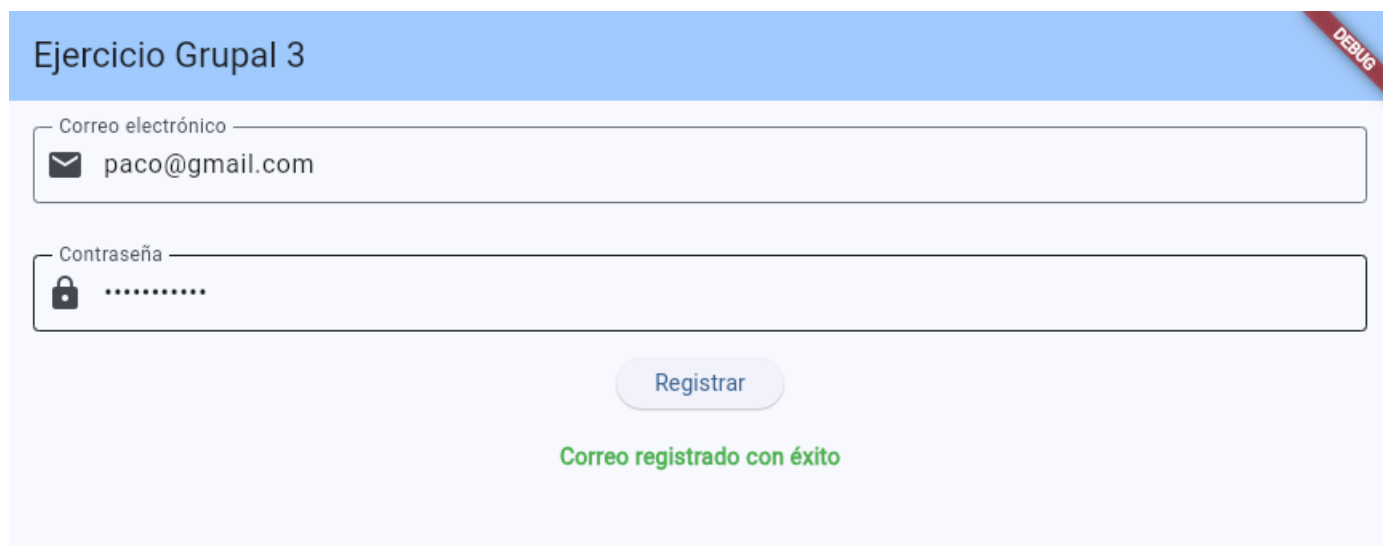
Correo electrónico —  
✉ paco@gmail.com

Contraseña —  
🔒 .....

Registrar

La contraseña tiene que contener al menos un número

Figura 1.4: Ejemplo de contraseña sin carácter número



Ejercicio Grupal 3

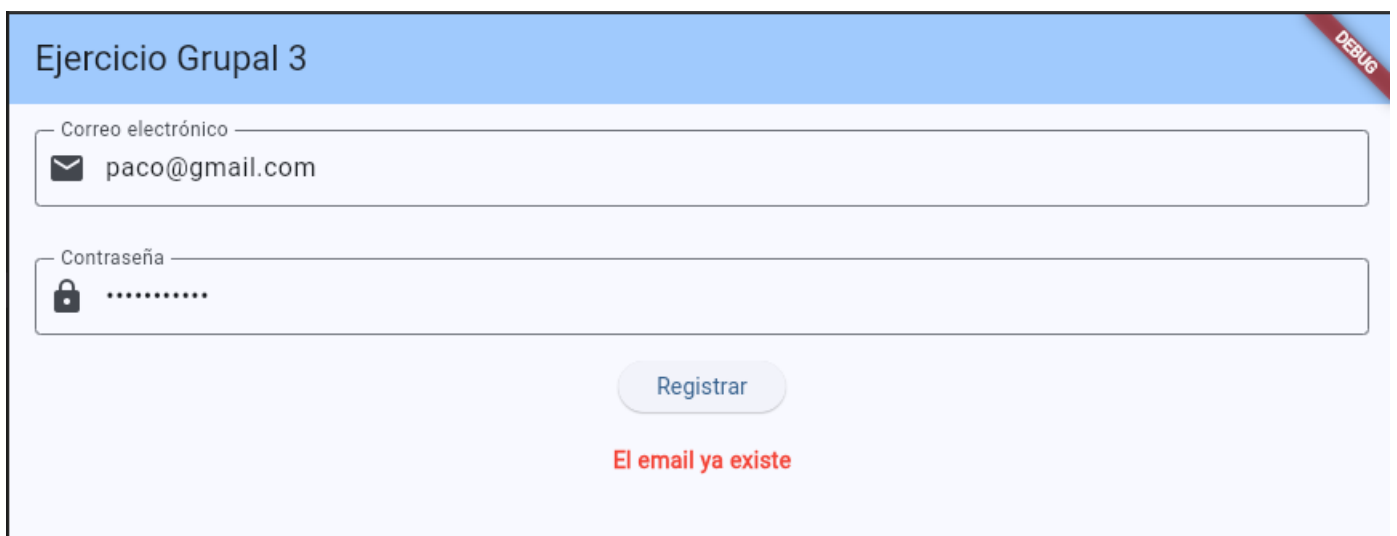
Correo electrónico —  
✉ paco@gmail.com

Contraseña —  
🔒 .....

Registrar

Correo registrado con éxito

Figura 1.5: Ejemplo de correo registrado con éxito



The image shows a Flutter application interface for a registration form. At the top, there is a blue header bar with the text "Ejercicio Grupal 3" on the left and a red "DEBUG" banner on the right. Below the header, there are two input fields. The first field is labeled "Correo electrónico" and contains the text "paco@gmail.com". The second field is labeled "Contraseña" and contains a series of dots, indicating a password. Below these fields, there is a button labeled "Registrar". Underneath the button, there is a red error message that says "El email ya existe".

Figura 1.6: Ejemplo de existencia de un correo igual previo

## 1.7. Esquema UML

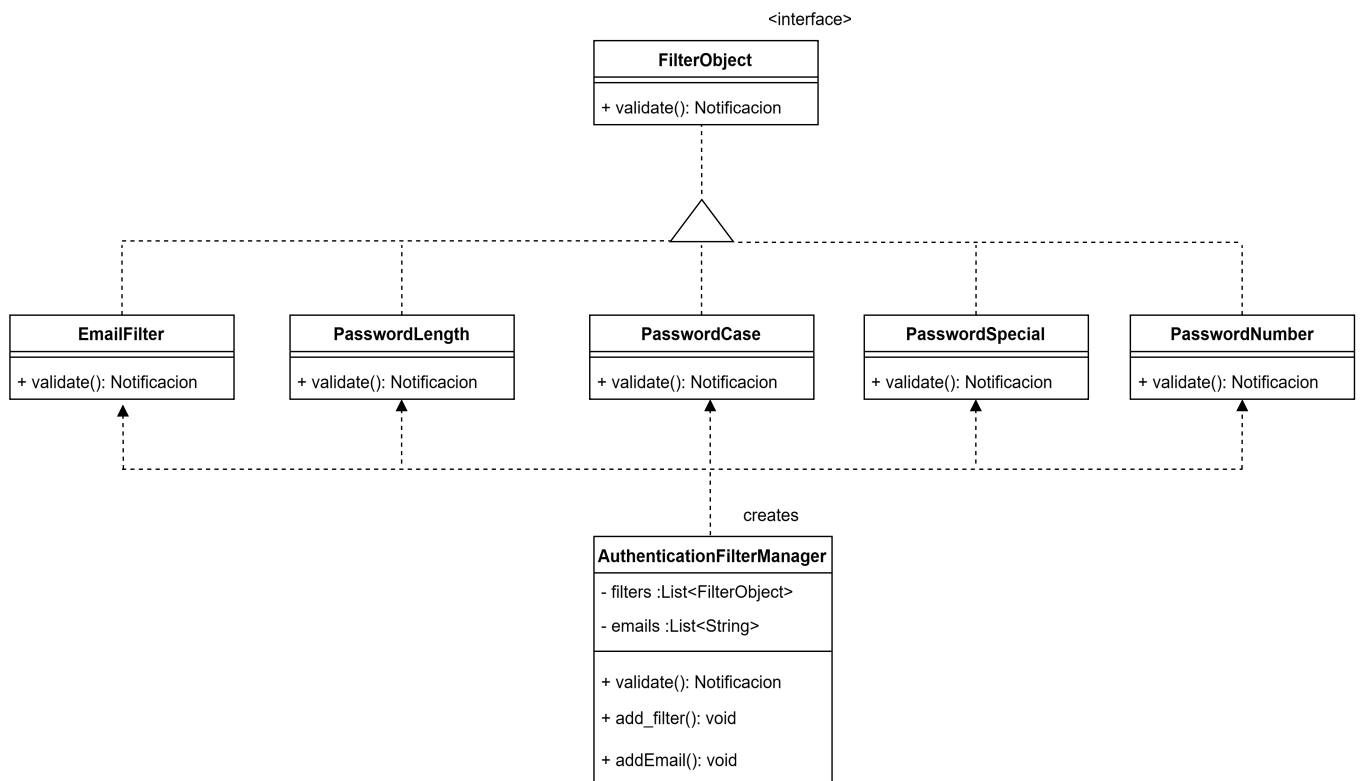


Figura 1.7: Diagrama UML del ejercicio 3.1

## Capítulo 2

# Ejercicio grupal opcional

### 2.1. Descripción

Investiga cómo es posible conectarse a Hugging Face desde Dart/Flutter. Requisitos:

- Haz una aplicación que se conecte a la API de Hugging Face.
- Implementa un patrón Strategy donde el usuario pueda seleccionar un LLM desde la interfaz, escriba un mensaje y obtenga una respuesta.

### 2.2. Conexión a la API

Para poder acceder a la API de Hugging Face y utilizarla en nuestra aplicación deberemos tener creada una cuenta y tener un token. Además, para poder usarla en nuestra aplicación, hemos tenido que añadir una dependencia para poder usar el protocolo http, la dependencia específica es `http: 0.13.6`

Tenemos distintos nombres de modelos (facebook/blenderbot-400M-distill, por ejemplo) el cual se concatena a la URL `https://api-inference.huggingface.co/models/`, dando lugar a la URL completa `https://api-inference.huggingface.co/models/facebook/blenderbot-400M-distill`.

Enviamos con una solicitud POST con el mensaje introducido por el usuario y añadimos el token de autenticación para poder acceder a los modelos.

### 2.3. Implementación del patrón Strategy

Tenemos una clase abstracta llamada `LLMStrategy`, la cual define qué debe tener cada estrategia. De esta clase, heredan las clases `BasicLLM` y `ExpansionLLM`, las cuales tienen “asociadas” un modelo de Hugging Face.

Cabe destacar que el `Context` en nuestro caso es el `Main`, donde se le da al usuario la opción de elegir entre distintas estrategias, introducir un mensaje y obtener la respuesta de acuerdo al modelo seleccionado.



## 2.4. Ejemplos de ejecución


A continuación mostraremos la ejecución de la aplicación. En primer lugar, nos encontramos con el `Basic_llm`



The screenshot shows a web application titled "Ejercicio Grupal Opcional" with a "DEBUG" button in the top right corner. The authors' names are listed: Antonio Pancorbo Morales, Luis Alberto Mejía Troya, and Manuel Jesús Junquera Lobón. Under the heading "Selecciona el modelo LLM deseado:", the "BART (Resumen)" model is selected. A text input field contains the message "hola adios", and an "Enviar" button is below it. The response, labeled "Respuesta:", is displayed in a black box with blue text: "Hola adios. Hola. I love you all. hola. Adios. I'll see you in a few days. hilo. I hope you have a good night. hula. I will see you next week. hala. I'm off."

Figura 2.1: Ejemplo de uso de BasicLLM

Por otro lado, tenemos el `Expasion_llm`



The screenshot shows a web application titled "Ejercicio Grupal Opcional" with a "DEBUG" button in the top right corner. The authors' names are listed: Antonio Pancorbo Morales, Luis Alberto Mejía Troya, and Manuel Jesús Junquera Lobón. Under the heading "Selecciona el modelo LLM deseado:", the "BlenderBot (Chat)" model is selected. A text input field contains the message "hola adios", and an "Enviar" button is below it. The response, labeled "Respuesta:", is displayed in a black box with blue text: "Have you ever heard of Holiday Adios? It's an American entertainment company."

Figura 2.2: Ejemplo de uso de ExpansionLLM

## 2.5. Esquema UML

Vamos a pasar a mostrar el esquema UML que hemos realizado para hacer esta práctica

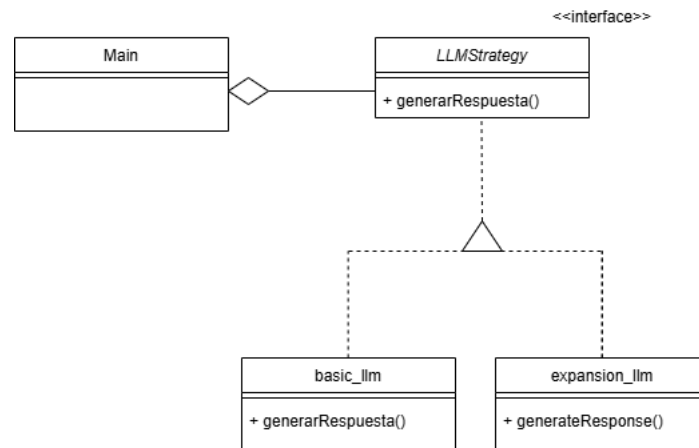


Figura 2.3: Diagrama UML del ejercicio 3.2