

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

1. Mantenimiento en Flutter	1
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
2. Ejercicio grupal opcional	6
2.1. Descripción	6
2.2. Conexión a la API	6
2.3. Implementación del patrón Strategy	6
2.4. Ejemplos de ejecución	6
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.

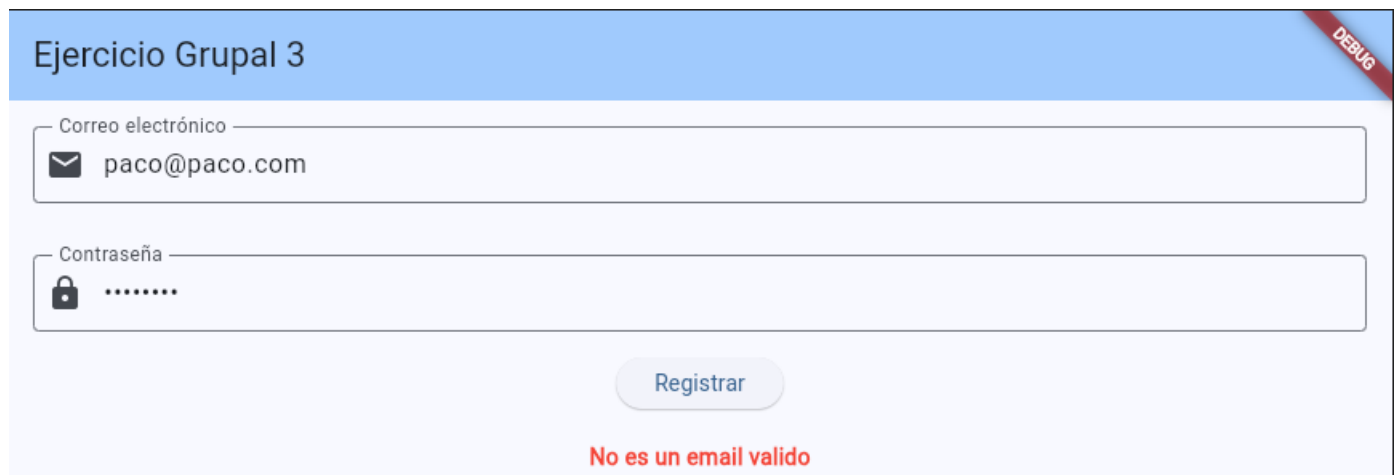
1.5. Sistema de notificaciones

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



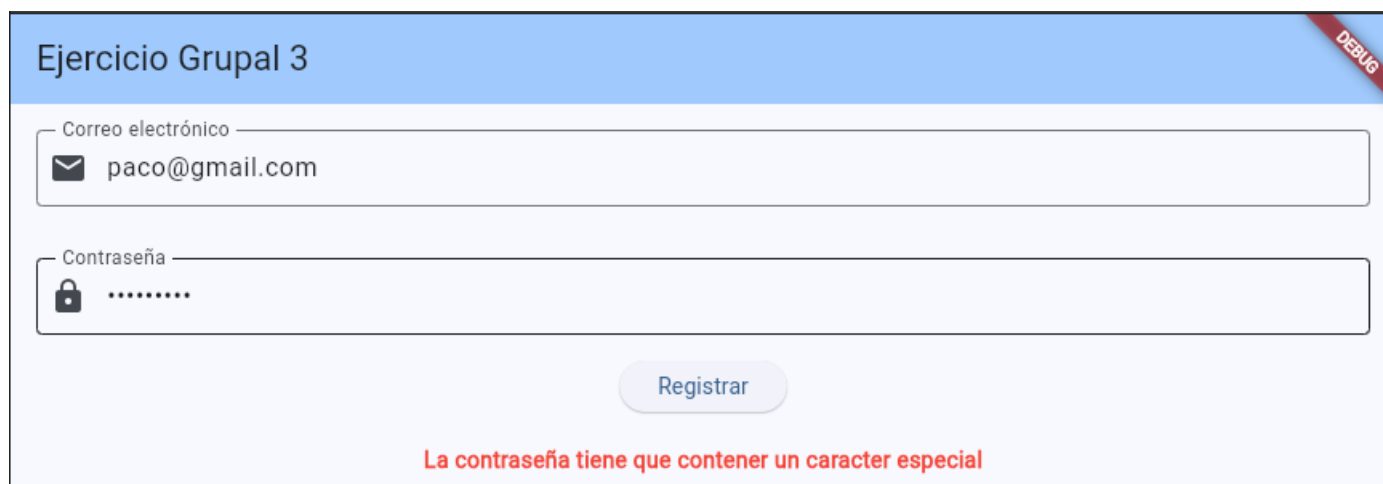
The screenshot shows a registration form titled "Ejercicio Grupal 3" with a blue header bar. In the top right corner of the header, there is a red ribbon with the word "DEBUG" in white. The form has two input fields: "Correo electrónico" (Email) and "Contraseña" (Password). The email field contains the text "paco@paco.com" and has an envelope icon on the left. The password field contains seven dots and has a lock icon on the left. Below the fields is a blue "Registrar" button. At the bottom of the form, there is a red error message: "No es un email valido".

Figura 1.1: Ejemplo de e-mail no válido



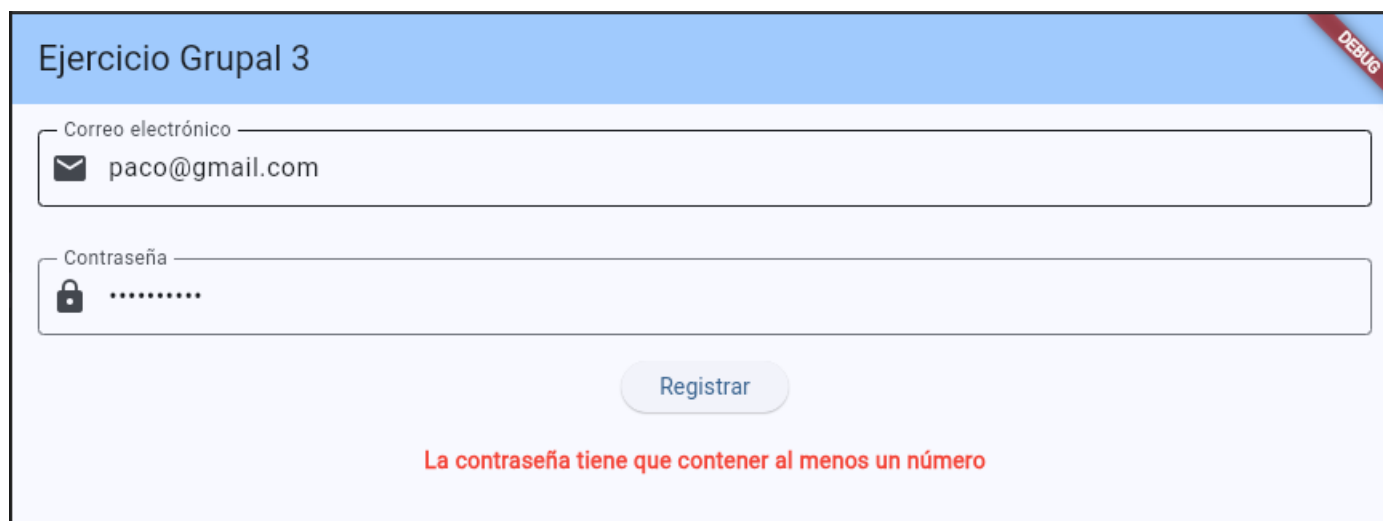
The screenshot shows the same registration form titled "Ejercicio Grupal 3" with the "DEBUG" ribbon. The email field now contains "paco@gmail.com". The password field still contains seven dots. The "Registrar" button is still present. At the bottom of the form, there is a red error message: "La contraseña tiene que contener una letra mayuscula y otra minuscula".

Figura 1.2: Ejemplo de contraseña sin mayúsculas



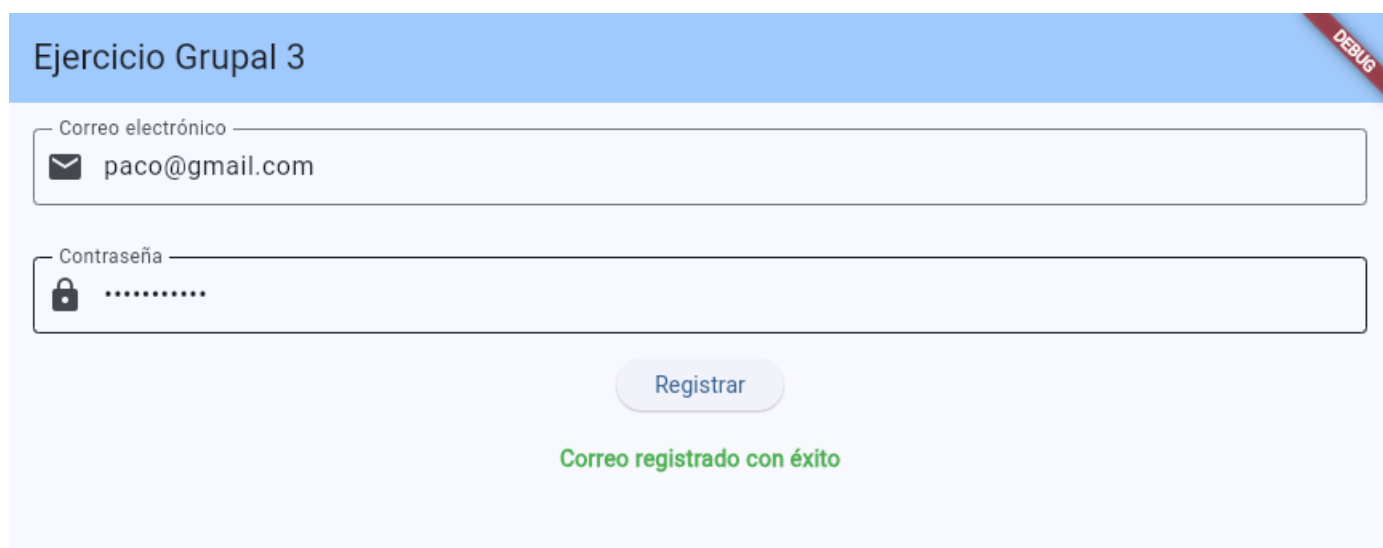
The screenshot shows a mobile app interface titled "Ejercicio Grupal 3" in a blue header bar. A red "DEBUG" banner is in the top right corner. Below the header, there are two input fields: "Correo electrónico" containing "paco@gmail.com" and "Contraseña" containing seven dots. A "Registrar" button is centered below the fields. A red error message, "La contraseña tiene que contener un caracter especial", is displayed at the bottom.

Figura 1.3: Ejemplo de contraseña sin carácter especial



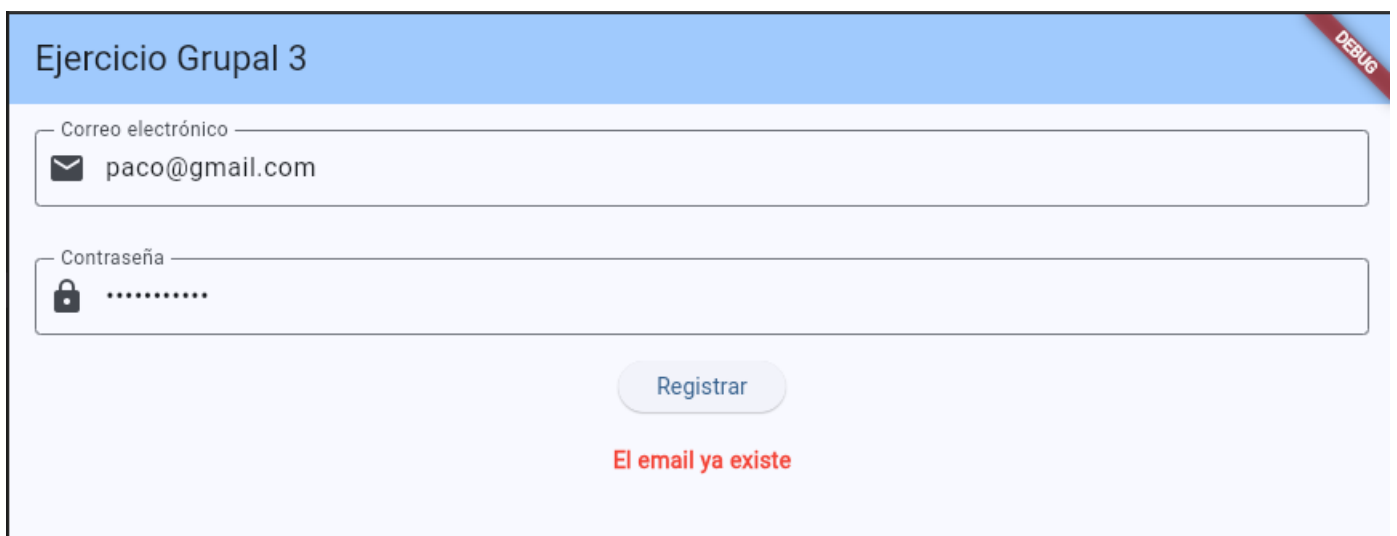
This screenshot is similar to the previous one, showing the same registration form. However, the red error message at the bottom now reads, "La contraseña tiene que contener al menos un número".

Figura 1.4: Ejemplo de contraseña sin carácter número



This screenshot shows the registration form after a successful registration. The error message has been replaced by a green success message, "Correo registrado con éxito", located below the "Registrar" button.

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 reads "El email ya existe".

Figura 1.6: Ejemplo de existencia de un correo igual previo

1.7. Esquema UML

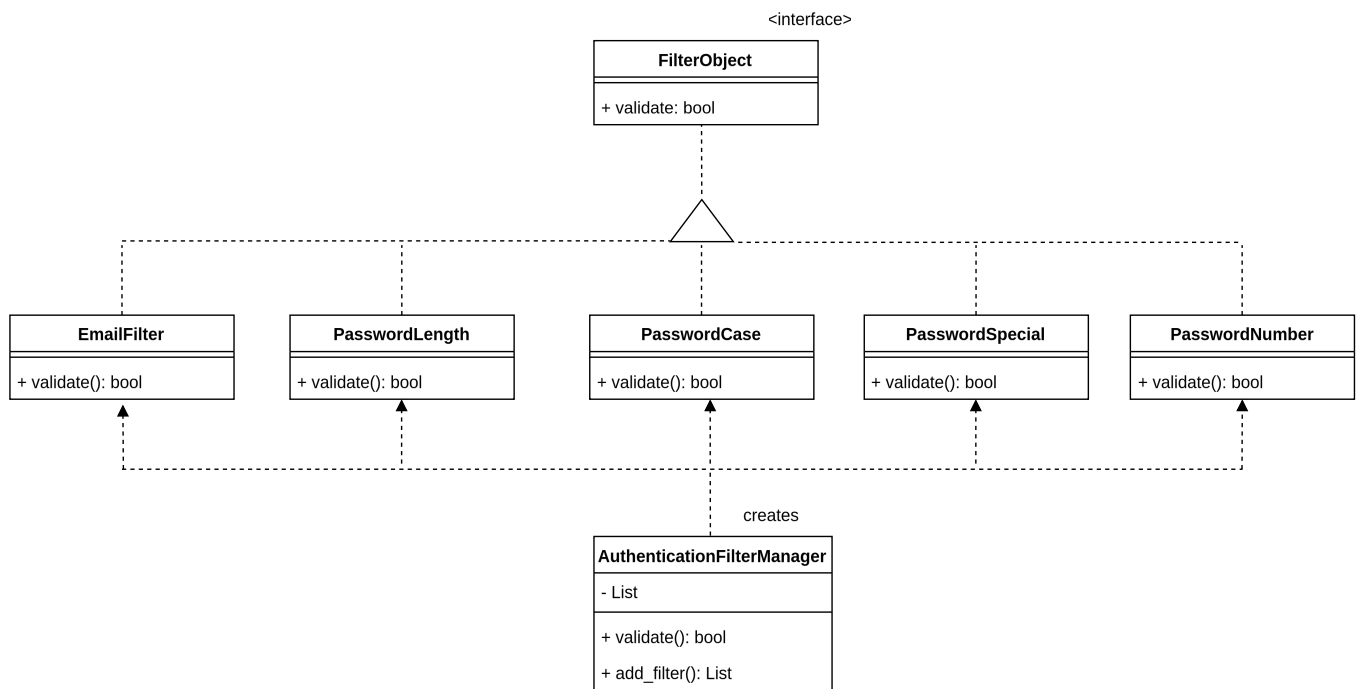


Figura 1.7: Diagrama UML del ejercicio 3.1

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.

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

Ejercicio Grupal Opcional

Autores:
Antonio Pancorbo Morales
Luis Alberto Mejía Troya
Manuel Jesús Junquera Lobón

Selecciona el modelo LLM deseado:
BART (Resumen) ▾

Escribe tu mensaje
hola adios

Enviar

Respuesta:
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

Ejercicio Grupal Opcional

Autores:
Antonio Pancorbo Morales
Luis Alberto Mejía Troya
Manuel Jesús Junquera Lobón

Selecciona el modelo LLM deseado:
BlenderBot (Chat) ▾

Escribe tu mensaje
hola adios

Enviar

Respuesta:
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

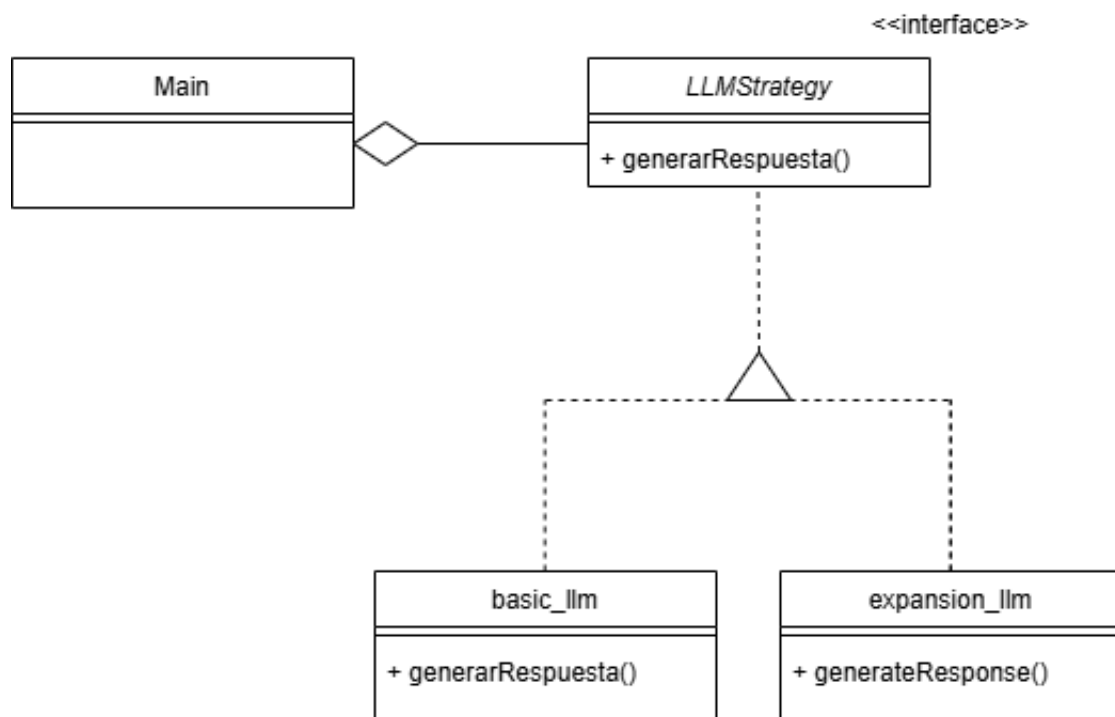


Figura 2.3: Diagrama UML del ejercicio 3.2