Patrones de Diseño para CookShare

En el desarrollo de nuestro proyecto “CookShare” se escogió un principal patrón de diseño. Este fue Singleton, el cuál es necesario para el manejo de la base de datos. Otro patrón que se podría utilizar podría ser Strategy, ya que podría aportar al sistema.

Singleton busca que haya una sola instancia global de una clase, a lo largo de toda la aplicación. Esto evita duplicaciones innecesarias y asegura la centralización de acceso a un solo recurso. Es utilizado ampliamente para el manejo de las bases de datos, ya que generar varias conexiones podría generar una sobrecarga o inconsistencias. En nuestra aplicación, Singleton ya está implementado implícitamente. Esto se debe a que el framework que usamos, Django, ya tiene este patrón implementado, y administra la conexión con la base de datos utilizando un esquema de Singleton. Por este mismo motivo, no fue necesario implementar o desarrollar de forma manual un Singleton adicional para la base de datos y para el funcionamiento de nuestra aplicación.

Por otro lado, hay otros patrones que se podrían utilizar en nuestra aplicación. Uno podría ser Strategy. Este permite la definición de múltiples algoritmos que se pueden intercambiar, para seleccionarlos según la necesidad del sistema. En la aplicación, este patrón podría ser implementado en el buscador de recetas, ya que se podrían ofrecer diferentes estrategias, tales como la filtración por popularidad, calificación, ingredientes, tiempo de preparación, u otros atributos. Esto se implementaría de forma independiente, pero permitiría que se puedan filtrar recetas en el código, sin afectar la lógica o funcionamiento principal de este mismo.