Justificación de los patrones de diseño

Enrique Alonso

***Singleton***

Singleton es un patrón fácil de implementar y en este caso, fue la mejor opción para poder establecer la conexión entre mi base de datos SQL y mi programa en cuestión.

***Proxy***

El patrón de proxy facilita el trabajo con las API, y el hecho de que ya lo hayamos implementado previamente con el mismo api, facilitó la decisión.

***Adapter***

Adapter me permitió modificar la clase forecast para hacerla compatible con la weather en ciertos casos, en mi caso fue para la parte de hacer pronósticos, usando adapter me permitió reciclar una clase que usaba objetos del tipo weather y solo tener que hacer una función que permitiera la comunicación entre ambas partes.

***Memento***

Quería agregar algo más en cuanto a características a mi programa y el permitirle al usuario enmendar sus errores con un solo clic en el botón hizo que fuera una decisión que aunque fue pesada su implementación, valió la pena.

***Prototype***

Al momento de crear la copia de la clase para los mementos, descubrí que no se estaba almacenando la lista de manera adecuada, entonces usando el patrón de prototype cree un método que hiciera una copia profunda de la clase que estaba utilizándose.