Documentación de Patrones de Diseño — PocketVet

1. Patrón Singleton

El patrón Singleton se usa para que una clase tenga una única instancia accesible globalmente. Es útil cuando se requiere un único punto de acceso compartido, como una conexión a la base de datos, además debe existir una coherencia global entre los datos o servicios. En PocketVet, este patrón se usará en la inicialización del servicio con Firebase para evitar múltiples conexiones simultáneas, ya que, si cada módulo del proyecto (por ejemplo, Mascotas, Eventos, Notificaciones) intentara crear su propia instancia de Firebase, se generarían múltiples conexiones simultáneas, aumentando el consumo de recursos y provocando errores de sincronización.

2. Patrón Observer

El patrón Observer define una relación de suscripción entre objetos, de modo que cuando uno cambia de estado (el sujeto), todos sus dependientes (los observadores) son notificados automáticamente. En PocketVet, este patrón puede utilizarse para el sistema de notificaciones. Por ejemplo, cuando se crea o modifica un evento (vacuna, cita o baño), se notifica al usuario mediante un canal (push o email) en lugar de que el módulo de eventos tenga que llamar manualmente a cada módulo de notificación, se usa el patrón Observer