# INFORME E1
## PATRÓN ESTADO

Hemos elegido el patrón estado para manejar los diferentes estados de los buques (EnBase, EnEjercicio, EnReparacion, PendienteReparacion, Hundido, Desmantelado). Cada estado es una clase que implementa la interfaz EstadoBuque, permitiendo que el comportamiento del buque cambie según su estado actual.

## Justificación:

-Flexibilidad: Permite añadir nuevos estados fácilmente sin modificar las clases existentes.
-Separación de Responsabilidades: Cada estado encapsula su propio comportamiento.
-Mantenibilidad: Facilita la modificación del comportamiento de un estado específico sin afectar a otros.
-Claridad del Código: Hace que las transiciones entre estados sean explícitas y fáciles de entender.

## Principios de Diseño SOLID Utilizados:

**1. Principio de Responsabilidad Única (SRP)**

Cada clase de estado (EnBase, EnEjercicio, etc.) tiene una única responsabilidad: manejar el comportamiento del buque en ese estado específico.
Ejemplo: La clase EnReparacion se encarga únicamente de las operaciones relacionadas con un buque en reparación.

**2. Principio de Abierto/Cerrado (OCP)**

El sistema está abierto a la extensión (se pueden añadir nuevos estados) pero cerrado a la modificación (no es necesario modificar la clase Buque para añadir nuevos estados).
Ejemplo: Se podría añadir un nuevo estado como TransportarMercancias sin modificar las clases existentes.

**3. Principio de Sustitución de Liskov (LSP)**

Todas las clases de estado pueden ser utilizadas de manera intercambiable a través de la interfaz EstadoBuque.
Ejemplo: La clase Buque trabaja con la interfaz EstadoBuque, permitiendo que cualquier estado concreto pueda ser utilizado sin afectar el funcionamiento del buque.

**4. Principio de Segregación de Interfaces (ISP)**

La interfaz EstadoBuque define métodos específicos para las operaciones de los buques, sin forzar a las clases a implementar métodos que no necesitan.
Ejemplo: Cada estado implementa solo los métodos relevantes de la interfaz EstadoBuque.

**5. Principio de Inversión de Dependencias (DIP)**

Las clases de alto nivel (Buque y Base) dependen de abstracciones (EstadoBuque) y no de implementaciones concretas.
Ejemplo: La clase Buque tiene una referencia a EstadoBuque, no a una implementación específica como EnBase o EnReparacion.

En conclusión, el diseño implementado utiliza el patrón Estado y añade los principios SOLID, que dan lugar a un sistema flexible, mantenible y extensible para la gestión de buques navales.