# Patrón Módulo

Carrillo Reyes Rogelio - 19211608

# Historia del patron modulo

El concepto de modularidad en el diseño de software se ha desarrollado a lo largo de décadas de evolución en la informática. A medida que los programas se volvían más grandes y complejos, los ingenieros de software buscaron formas de organizar y estructurar el código de manera más efectiva. Aunque no hay una única persona o evento que pueda atribuirse como el origen del patrón de diseño módulo, su evolución se remonta a hitos importantes en la historia del desarrollo de software. Estos incluyen el surgimiento de lenguajes de programación como Pascal, el desarrollo de la programación orientada a objetos, y la popularización de patrones de diseño en la década de 1990. La modularidad sigue siendo un principio fundamental en el diseño de software hoy en día, permitiendo la creación de sistemas más flexibles, escalables y mantenibles.

# ¿Que es la modularidad?

La modularidad es, en programación modular y más específicamente en programación orientada a objetos, la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las partes restantes.

Estructurada

Modular

# ¿En qué consiste el patrón módulo?

El patrón módulo consiste en un módulo donde se encapsula toda la lógica de nuestra aplicación o proyecto. Dentro de este módulo estarán declaradas todas las variables o funciones privadas y sólo serán visibles dentro del mismo.

# Características del patrón módulo

**Encapsulación:** Oculta los detalles internos de un módulo, previniendo conflictos y facilitando el mantenimiento del código al agrupar datos y funciones dentro de un solo componente.

**Abstracción:** Simplifica la complejidad del sistema al organizar componentes relacionados en módulos individuales lo que facilita la comprensión y el diseño del sistema en su conjunto.

**Reutilización:** Permite que los módulos sean reutilizables en diferentes partes de una aplicación o proyecto, al separar la funcionalidad en unidades independientes y cohesivas.

**Independencia:** Los módulos deben ser lo más independientes posible, lo que facilita el desarrollo y la prueba por separado, mejorando así la mantenibilidad y la escalabilidad del sistema.

**Coherencia y cohesión:** Los módulos deben estar bien estructurados y organizados, los componentes dentro de un módulo deben estar relacionados entre sí y trabajar juntos para lograr un propósito común, facilitando su comprensión y mantenimiento.

**Interfaz clara:** Cada módulo debe definir una interfaz clara y bien definida que especifique cómo interactuar con él, actuando como un contrato entre el módulo y otros componentes del sistema

# Componentes principales del patrón módulo

**Función envolvente:** Esta función envuelve todo el código del módulo y ayuda a crear un ámbito local para las variables y métodos privados.

**Variables y métodos privados:** Son las variables y funciones que están encapsuladas dentro del ámbito de la función envolvente. Estos elementos no son accesibles desde fuera del módulo y solo pueden ser utilizados por los métodos internos del módulo.

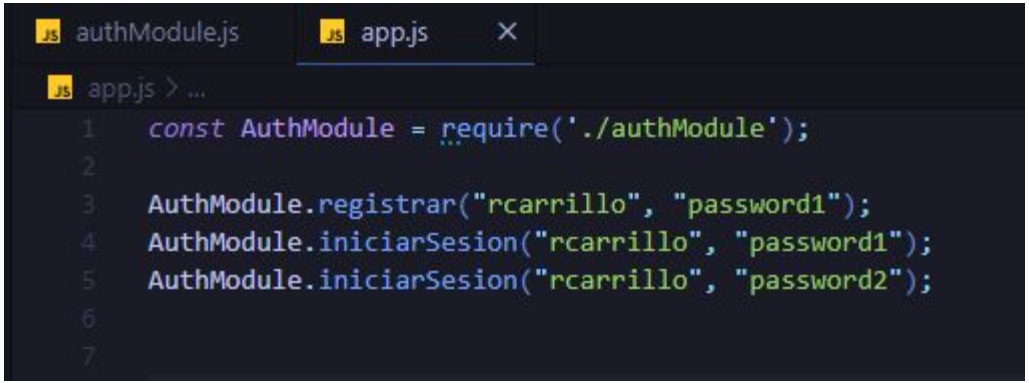
**Objeto de retorno:** Es el objeto que se devuelve al final de la función envolvente. Este objeto contiene los métodos y variables que deseamos hacer públicos, es decir, aquellos que queremos que estén disponibles fuera del módulo.

**Métodos públicos:** Son los métodos que están incluidos en el objeto de retorno y que se pueden utilizar desde fuera del módulo. Estos métodos pueden acceder a las variables y métodos privados dentro del ámbito de la función envolvente, lo que permite una interfaz controlada para interactuar con el módulo.

# Ejemplo del patrón módulo en JavaScript


```
const AuthModule = (function() {
  let usuariosRegistrados = [];

  function registrarUsuario(usuario, contraseña) {
      usuariosRegistrados.push({ usuario: usuario, contraseña: contraseña });
      console.log("Usuario registrado exitosamente:", usuario);
  }

  function iniciarSesion(usuario, contraseña) {
      for (const i = 0; i < usuariosRegistrados.length; i++) {
          if (usuariosRegistrados[i].usuario === usuario &&
            usuariosRegistrados[i].contraseña === contraseña) {
              console.log("Inicio de sesión exitoso:", usuario);
              return true;
          }
      }
      console.log("Error: Usuario o contraseña incorrectos.");
      return false;
  }

  return {
      registrar: function(usuario, contraseña) {
          registrarUsuario(usuario, contraseña);
      },
      iniciarSesion: function(usuario, contraseña) {
          return iniciarSesion(usuario, contraseña);
      }
  };
})();

module.exports = AuthModule;
```

# Ejemplo del patrón módulo en JavaScript



# ¡Gracias por su atención!