

SwiftUI Pills

Programación Orientada a Protocolos



Conceptos simples

Lo que necesitas saber en una primera instancia para luego poder manejar la complejidad de conceptos más avanzados.

Nivel principiante

Explicaciones de mínima complejidad para poner foco en lo que importa.

Código de prueba

Código corto y simple para que pongas a prueba ideas y conceptos.

Casos reales

Situaciones de aplicación en casos reales para que visualices el ejemplo como una herramienta lista para usar.

Explicaciones claras

Explicaciones que explican, no que complican..

Preguntas de entrevistas

Preguntas y respuestas sobre el tópico que usualmente aparecen en entrevistas técnicas.

El conocimiento en programación es como la construcción de un edificio: la solidez de toda la estructura depende de la calidad de cada ladrillo. Cada unidad de conocimiento debe ser clara, sólida y comprensible por sí misma, independiente de las demás. Con el tiempo, estos bloques se consolidan y se conectan creativamente con otros, formando una estructura robusta y armoniosa.

Los protocolos constituyen uno de los conceptos fundamentales de Swift y adquieren una importancia especial en SwiftUI. No se trata de una simple característica para resolver casos de uso específicos, sino de un paradigma completo de programación que transforma nuestra forma de entender la arquitectura de la aplicación y aspectos clave del proyecto, como la extensibilidad, reutilización y modularidad.

¿Entonces, qué es un protocolo?

Un **protocolo** en Swift es un tipo que define un conjunto de métodos, propiedades y otros requisitos que deben implementarse en cualquier clase, estructura o enumeración (**tipo conformable**) que lo adopte. Cuando un tipo conforma un protocolo, entra en un “**contrato**” que garantiza que implementará todo lo especificado por el protocolo.

¿Cuál es la importancia de los protocolos?

Los protocolos en Swift son fundamentales para escribir código modular, reutilizable, flexible y fácil de mantener. Al definir contratos de comportamiento, los protocolos nos permiten centrarnos en las interfaces y en el comportamiento esperado, sin importar las implementaciones específicas de los tipos que los adoptan.

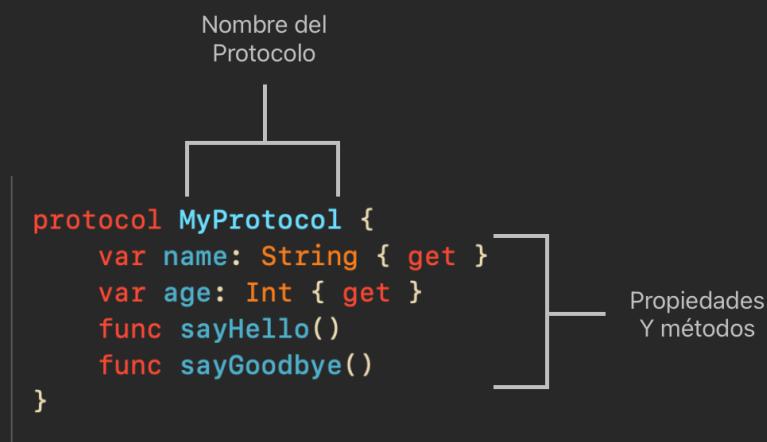
La importancia de los protocolos en Swift radica en dos aspectos clave:

1. Base de la Programación Orientada a Protocolos (POP): Apple ha promovido un paradigma de desarrollo conocido como Programación Orientada a Protocolos (POP), en el cual los protocolos desempeñan un papel central en el diseño y organización del código. Este enfoque fomenta la composición de comportamientos a través de protocolos en lugar de depender exclusivamente de la herencia de clases, permitiendo que distintos tipos compartan funcionalidades de manera flexible y eficiente.
2. Uso extenso en el ecosistema de Apple: Los protocolos son uno de los recursos más ampliamente utilizados en las bibliotecas de Apple, como Foundation y SwiftUI. La arquitectura de estas bibliotecas está diseñada para depender de protocolos que faciliten la interoperabilidad y flexibilidad en sus componentes. Apple recomienda activamente el uso de protocolos en el desarrollo de aplicaciones, ya que favorecen la claridad y la escalabilidad del código en proyectos complejos.

Gracias a los protocolos, los desarrolladores pueden definir estructuras y clases que cumplan con interfaces bien definidas y reutilizar módulos de manera más efectiva, promoviendo una arquitectura más robusta y adaptable.

¿Cómo se define un protocolo?

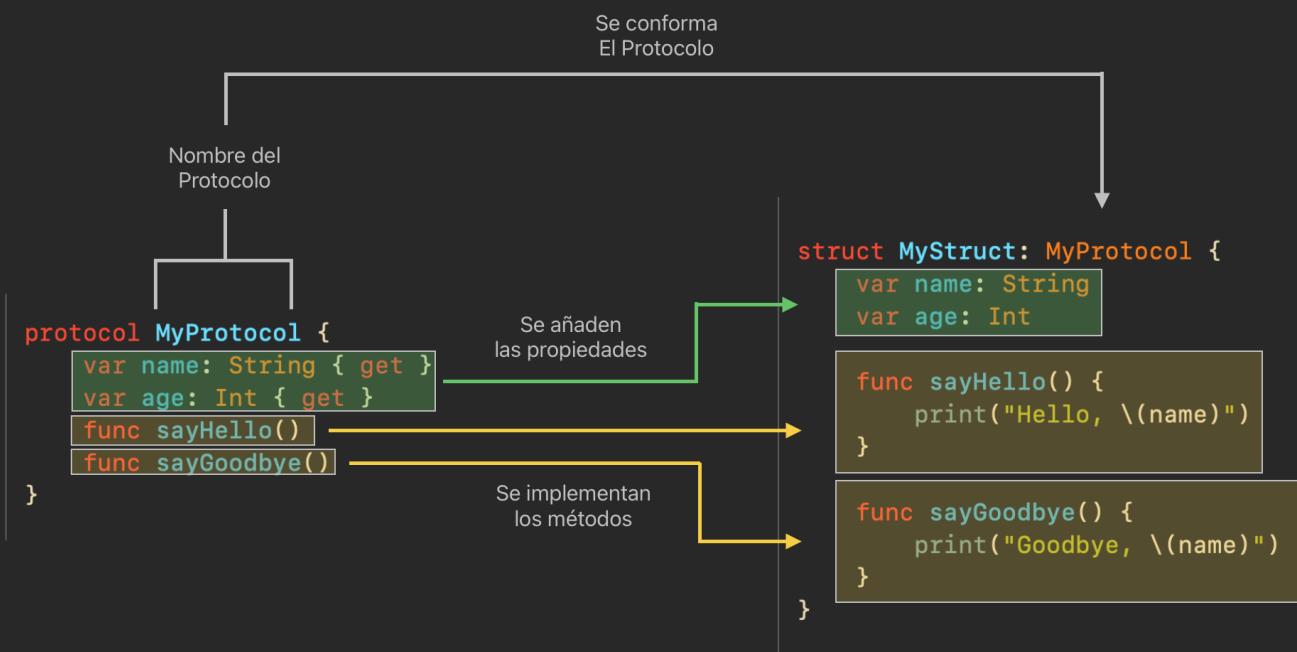
Se escribe la palabra reservada **protocol**, se le suministra un nombre y -entre llaves- se definen sus propiedades y métodos.



* Nótese que sólo se escribe la firma del método, no su implementación.

¿Cómo hace un tipo conformable (estructura, enumeración o clase) para conformar un protocolo?

Luego del nombre del tipo, se escriben dos puntos (:) y a continuación el nombre del protocolo a conformar (en mayúsculas por hacer referencia a un tipo de datos).



Finalmente así es como queda nuestra estructura `MyStruct` conformando el protocolo `MyProtocol`.

```

struct MyStruct: MyProtocol {
    var name: String
    var age: Int

    func sayHello() {
        print("Hello, \(name)")
    }

    func sayGoodbye() {
        print("Goodbye, \(name)")
    }
}

```

Lo que hemos aprendido hasta ahora es qué son los protocolos, cuál es su importancia y de qué manera una estructura, clase o enumeración lo conforma e implementa sus propiedades y métodos.

En publicaciones posteriores seguiremos avanzando para responder nuevas preguntas:

- En que casos los protocolos funcionan mejor que la herencia de clases?
- Que situaciones pueden resolverse únicamente con protocolos y no de otra manera?
- Los protocolos de Swift tienen su equivalente en lenguajes como Java, JavaScript, C++, .NET o Python?

Entonces, hasta la próxima!