**Programación asincrónica en PHP**

Programación asincrónica permite que una aplicación realice múltiples tareas de manera simultanea sin esperar a que una tarea termine antes de comenzar otra. Esto se logra por medio del uso de técnicas que evitan el bloqueo del flujo de ejecución que permiten el flujo concurrente.

* Por naturaleza, PHP es un lenguaje síncrono y bloqueante (Una tarea debe terminar antes de empezar otra).
* Usar herramientas como ReactPHP permiten a aplicaciones tener la capacidad de manejar múltiples operaciones de entrada/salida, mejorando el rendimiento ante alta concurrencia.

**Librería ReactPHP**

Es una librería de bajo nivel para la programación que facilita el uso del paradigma event-driven (programación asíncrona y basada en eventos) permitiendo manejar operaciones no bloqueantes.

ReactPHP funciona bajo el bucle de eventos (evento loop), la cual es el responsable de monitorear y despachar eventos a los manejadores correspondientes. Este bucle permite que la aplicación responda a eventos a medida que ocurren, facilitando la ejecución de operaciones asíncronas.

Usos comunes de programación asincrónica:

* Chats en tiempo real, sistema de notificaciones, operaciones online.
* Servidores de APIs que manejan múltiples solicitudes concurrentes.
* Aplicaciones que necesitan comunicarse con otros servicios sin bloquear el flujo de ejecución.

**Fuente:** <https://reactphp.org/>

**Fuente:** <https://beyondco.de/video-courses/learning-reactphp>

**Bucle de eventos (Event Loop)**

Un bucle de eventos es un patrón de programación que espera y distribuye eventos o mensajes en un programa. Es fundamental en la programación dirigida por eventos y permite que una aplicación responda a eventos a medida que ocurren.​

* ReactPHP implementa su propio Event Loop para el manejo de operaciones asincrónicas.
* El bucle de eventos monitorea múltiples fuentes de eventos y ejecuta las funciones de devolución de llamada asociadas cuando se detecta un evento.

**Entrada/Salida No bloqueante (Non blocking I/O)**

La E/S no bloqueante permite que las operaciones de entrada y salida se realicen sin detener el flujo de ejecución del programa. En lugar de esperar a que una operación de E/S se complete, el programa puede continuar ejecutando otras tareas.

* El uso de E/S no bloqueante es esencial para construir aplicaciones eficientes y escalables que manejan múltiples conexiones simultáneamente.

**Pasos para el desarrollo de un sitio web asincrónico con el uso de ReactPHP**

Una captura de pantalla de un celular con texto e imagen

El contenido generado por IA puede ser incorrecto.

1. **Estructura de carpetas basada en el patrón MVC (Modelo-Vista-Controlador) donde**:

* **App:** Contiene a los controladores, modelos y vistas.
* **Public:** Contiene recursos como estilos CSS y las imágenes.
* **Routes:** Maneja las rutas de cada archivo por medio de web.php.
* **Server.php:** Aquí se crea el servidor Event-Driven para usarlo con ReactPHP.

**2) Ejecutar las dependencias necesarias para el proyecto (ES NECESARIO TENER INSTALADO COMPOSER)**

* **composer require react/http** 🡪 Permite levantar el **servidor no bloqueante** propio en PHP, que maneja rutas, peticiones y respuestas de forma asíncrona.
* **composer require react/event-loop** 🡪 Es la parte fundamental para el uso del ReactPHP, ya que este se encarga de ejecutar tareas asincrónicas sin bloquear. Por ejemplo, leer archivos, hacer consultas a una base de datos, etc. ReactPHP trabaja sobre este loop.
* **composer require react/socket 🡪** Permite crear servidores de bajo nivel o trabajar directamente con puertos. “react/http” lo usa internamente, y puede usarse para comunicación TCP/IP no bloqueante si se desea.
* **composer require react/filesystem 🡪** Es la dependencia que da el acceso asincrónico al sistema de archivos, com leer, escribir y eliminar archivos de forma no bloqueante. Es ideal para leer archivos HTML o loga sin congelar el servidor.
* **composer require react/promise 🡪** Permite trabajar con flujos asincrónicos de forma organizada, como consultas a la base de datos no bloqueante, solicitudes HTTP asíncronas, etc. ReactPHP funciona sobre un bucle de eventos. Como no bloquea el proceso principal, necesita una forma de saber cuándo una operación ha terminado sin detener todo.
* **composer require react/mysql 🡪** Biblioteca asíncrona para interactuar con bases de datos MySQL dentro de aplicaciones que usan ReactPHP. Esta diseñado específicamente para trabajar con el bucle de eventos de ReactPHP. Permite realizar consultas a bases de datos sin bloquear el flujo de ejecución, lo cual es útil en aplicaciones de alto rendimiento y que gestionan múltiples solicitudes concurrentes, como en un servidor web basado en eventos.