Bitcoin Core está compuesto por múltiples interfaces para interactuar con la Blockchain de Bitcoin. Tal vez te preguntes **cómo logra funcionar técnicamente todas estas partes de este gran sistema**.

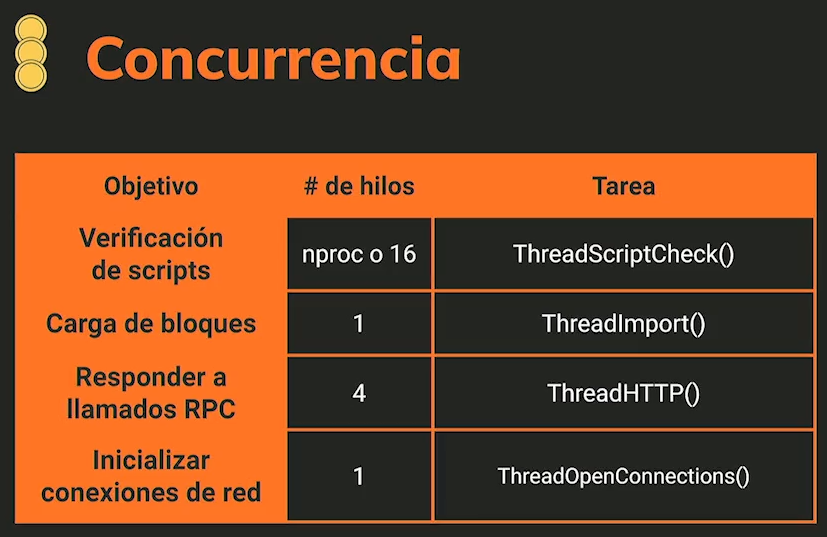
**Funcionamiento técnico de Bitcoin Core**

El funcionamiento de Bitcoin Core utiliza el poder de la [concurrencia](https://platzi.com/clases/2232-programacion-golang/35756-que-es-la-concurrencia/) y la [asincronicidad](https://platzi.com/clases/3175-asincronismo-js/50088-que-es-el-asincronismo/" \t "_blank) para lograr que múltiples subsistemas trabajen en paralelo.

**Concurrencia en Bitcoin Core**

La concurrencia le permite a Bitcoin Core realizar tareas simultáneas utilizando hilos de ejecución. Posee interfaces de validación que le permiten registrar *callback* para responder a eventos para, por ejemplo, enviar notificaciones con ZMQ.

En concurrencia existen **múltiples procesos relacionados con las funcionalidades de la red**. Veamos algunos de ellos:



Existen procesos para la verificación de scripts, para la carga de bloques, respuestas a llamados RPC o para la inicialización de conexiones de red, entre otros. Cada proceso utiliza N cantidad de hilos y es ejecutado por una tarea. Los hilos se cargan al iniciar Bitcoin Core para estar listos para su funcionamiento.

**Sección en Bitcoin Core**

Dentro de Bitcoin Core existen **múltiples secciones para responder a todos los subsistemas e interfaces** para que Bitcoin y todo su protocolo pueda funcionar.

El código fuente Bitcoin Core está dividido en secciones. Puedes encontrar cada una de ellas en el [repositorio del proyecto](https://github.com/bitcoin/bitcoin/tree/master/src).



* [net](https://github.com/bitcoin/bitcoin/blob/master/src/net.cpp): base de todas las secciones, es donde se realiza toda la conexión P2P y todo el proceso de mensajería.
* [net\_processing](https://github.com/bitcoin/bitcoin/blob/master/src/net_processing.cpp): responde a los eventos de mensajería de **net**.
* [validation](https://github.com/bitcoin/bitcoin/blob/master/src/validation.cpp): reglas que permiten validar cuál es la cadena con mayor poder de cómputo acumulado y agregar transacciones al mempool.
* [txmempool](https://github.com/bitcoin/bitcoin/blob/master/src/txmempool.cpp): implementación base del mempool de Bitcoin.
* coins & txdb: implementación base para el funcionamiento de UTXO (salidas no gastadas).
* [script/](https://github.com/bitcoin/bitcoin/tree/master/src/script): directorio que contiene scripts para la ejecución de tareas como la creación de transacciones.
* [consensus/](https://github.com/bitcoin/bitcoin/tree/master/src/consensus): parámetros para el funcionamiento del consenso de Bitcoin.
* [policy/](https://github.com/bitcoin/bitcoin/tree/master/src/policy): estimación de las tarifas, existen empresas dedicadas solo a estos cálculos.
* [indexes/](https://github.com/bitcoin/bitcoin/tree/master/src/index): creación de índices para la búsqueda de bloques o transacciones.
* [wallet/](https://github.com/bitcoin/bitcoin/tree/master/src/wallet): billetera que no se suele utilizar en producción, pero nos sirve como base para desarrollar nuevas o hacer pruebas.

**Conclusión**

Entendiendo que por detrás del Bitcoin Core existen complejos mecanismos de concurrencia, multitareas y asincronicidad, puedes al menos entender mejor su funcionamiento y la obra de ingeniería de software qué es Bitcoin.

El código fuente de Bitcoin Core también es muy grande y complejo. Si lo divides en secciones, y entendiendo básicamente en qué consiste cada una de ellas, puedes navegar por el código fuente de Bitcoin de formas más sencillas y entender el mismo.

**Concurrencia y regiones**

* **Concurrencia**
  + Bitcoin coin realiza tareas simultaneas.
  + Hilos.
  + Interfaz de validación. - **Secciones**
  + Subsistemas existentes en Bitcoin Core.
  + Procedimientos necesarios para la operación de Bitcoin.