cURL

Inicios de cURL

cURL empieza hace 25 años (HTTP GET 1996-1997) y surge como una pequeña aplicación para descargar tasas de cambio en HTTP. Pasa a llamarse más adelante URL GET 1997 v2 y v3.

Durante el año 1997 se añade soporte para Gopher y pasa a llamarse URL GET y se empieza a añadir soporte para FTP y el nombre vuelve a actualizarse y pasa a llamarse ya cURL, siendo la versión 4.0.

En los 4 primeros años de la aplicación se trabajó en un menú de comandos con múltiples opciones, que fueron añadiendo y parcheando, hasta que en el año 2000 nació libcurl, una librería que provee una API y capacidad de transferencia a través de internet.

El primer lenguaje en utilizar esta librería fue PHP, lo que dio a conocer esta librería y la hizo despegar, lo que les obligó a resolver muchos bugs para ser compatible con una comunidad tan grande como la de PHP. A partir de ese punto, se fueron añadiendo soporte para otros protocolos como HTTPS y otros backends, también se daba soporte a protocolos como SS, luego convertidos en SSL (SSLE, GNU y NSS).

El crecimiento de cURL

La prioridad de Daniel (el creador de esta librería) siempre ha sido HTTP, ya que ha sido lol que ha dado a conocer la librería. También comenta que no ha sido nada fácil actualizar y mejorar el soporte para este protocolo aún hoy en día y que tiene que enfrentarse a reportes de bug de código de hace más de 10 años, lo que dificulta la mantenibilidad del código, debido a los avances tan rápidos que se vienen dando en los últimos años, consecuencia de las múltiples evoluciones y que aún no esté terminado.

En cuanto a C como lenguaje de programación, a Daniel le gusta ya que en los 90 era el único lenguaje apto para el desarrollo de una librería como la que tenía en mente por aquel entonces. Se decantó por C en vez de C++ ya que C++ en aquellos tiempos aún no estaba tan avanzado. Además, comenta que C es el lenguaje que hace posible que libcurl y cURL estén disponibles en cualquier lugar, lo que hace que Daniel no se plantee cambiar de lenguaje.

Bugs y el uso de librerías externas

Debido a que cURL ha crecido tanto ha hecho que hayan aumentado también el número de bugs que se reportan y se encuentran por los usuarios.

Como últimas implementaciones que se están dando lugar, está la implementación de otras librerías, como Hyper, escrita en Rust.

Después de un rato comentando alguno fallos con licencias, las interfaces gráficas de hoy en día y la mala comunicación entre las empresas y los consumidores y su feedback, pasan a comentar más sobre el uso de otras librerías externas que se están utilizando en cURL, para implementaciones que no habían hecho ellos, pero que creían que podía ser una buena adición a la librería.

Daniel normaliza el uso de las librerías y comenta como se puede producir cierta transición entre C y Rust, siento estas 2 opciones muy válidas y compatibles, pero que C tiene varios problemas y bugs relacionados con el uso de C y la programación en este lenguaje (estima que sobre la mitad de los bugs vienen del lenguaje C).

Open Source

Daniel explica que su librería es Open Source sin embargo siempre intentan llevar un riguroso timeline de realeases y resolución de fallos para que se vea una evolución continuada.

Daniel estructura los lanzamientos de la siguiente manera: Hay un reléase cada 8 semanas, las 4 primeras semanas se utilizan para añadir código y funcionalidades a la aplicación, la segunda mitad, se dedica a la solución de bugs y errores que se hayan podido encontrar o reportar.

Las ventajas que comenta Daniel de trabajar así, son la gran capacidad y el tiempo que tienen para solucionar bugs, que es una de las partes más importantes del desarrollo, y también la forma en la que cada 8 semanas se planta con una lista en blanco de "nada que hacer" lo que indica que todo lo que se ha hecho en las últimas 8 semanas ha sido completado con éxito y que todo va según lo previsto. Esto le resulta muy satisfactorio, y hace que sea un gusto para el trabajar en esta librería de esta forma.

Como última parte, se comenta la forma de trabajar en un proyecto Open Source, donde la gente usuaria de la aplicación puede aportar pull requests que añadan más funcionalidad o arreglen código que fuera erróneo o no óptimo. Daniel explica como la manera que tienen de aceptar pull requests es más o menos "pseudo-aleatoria" y que suelen aceptar propuestas maduras y serias, que la funcionalidad que añadan sea revisable por el equipo de desarrollo.

También se comenta como a veces tiene que rechazar algunas propuestas o cosas que él no quiere añadir a la librería, no por motivos de implementación si no por que simplemente no le cuadra añadirlo a la librería.

Para finalizar y como resumen, Daniel ahonda en que una de las principales cualidades de cURL y libcurl ha sido la persistencia y la manera en la que en todo momento se ha trabajado sobre unos cimientos funcionales y como se ha trabajado sin esperar éxito, con la premisa de que si tienes una buena idea que no triunfa en un primer momento, no quiere decir que tu idea sea mala; que si crees en lo que haces y perseveras, tienes la posibilidad de que tu idea acabe siendo conocida y triunfe