# Críticas y reseñas

# Programación en Rust

Santiago Higuera de Frutos

Revista de Investigación

Volumen XIII, Número 1, pp. 171–174, ISSN 2174-0410
Recepción: 01 Feb'23; Aceptación: 27 Feb'23

1 de abril de 2023

**Resumen**

Este artículo hace una reseña del libro *Programación en Rust*. Este libro permitirá a los programadores introducirse en el lenguaje de programación Rust. Se trata del lenguaje que está siendo utilizado por los desarrolladores de los principales sistemas operativos (*Windows, Linux, Android*) para sustituir al C y al C++, debido a las mejoras que introduce en relación a la seguridad del código generado, sin perder la velocidad que caracteriza a C.

**Palabras Clave:** Programación, Lenguaje Rust

**Abstract**

This article reviews the book *Programming in Rust*. This book will allow programmers to get introduced to the Rust programming language. It is the language that is being used by the developers of the main operating systems (*Windows, Linux, Android*) to replace C and C++, due to the improvements it introduces in relation to code security generated, without losing the speed that characterizes C.

**Keywords:** Programming, Rust language

# 1. Ficha técnica


**Título:** Programación en Rust
**Autor:** Santiago Higuera de Frutos
**Editorial:** Garceta Grupo Editorial
**Edición:** 1ª edición en castellano
**Fecha publicación:** noviembre 2022
**ISBN:** 978-84-1728-988-1
**Páginas:** 244

# 2. El libro

Desde el año 2015, de acuerdo con la encuesta anual que realiza Stack Overflow entre más de 80.000 programadores de todo el mundo, Rust ha resultado ser el lenguaje más apreciado (*The most loved language*).

Rust se creó con el objetivo de obtener un lenguaje de prestaciones similares o superiores a las de C o C++, pero poniendo énfasis en la seguridad del código, aspecto en el que estos lenguajes han dado lugar a numerosos problemas. Más del 70 % de los errores de seguridad en los productos de Microsoft o de Google Chrome están originados por fallos de seguridad en el manejo de la memoria.

Pero Rust no solo ofrece código seguro. Rust ofrece una documentación de alta calidad, permite la programación concurrente de manera eficiente y segura, permite programar Web Assembly, ofrece un alto rendimiento en el procesamiento de grandes cantidades de datos y, además, dispone de un compilador muy efectivo y un entorno de desarrollo completo, con servicios para documentación de programas, servicios para test unitarios o de integración, servicios para control de versiones y mucho más.

Rust es un lenguaje de código abierto. Inicialmente se desarrolló al amparo de la empresa Mozilla, que desarrolla el popular navegador Firefox, entre otros muchos programas. Desde 2021, el desarrollo está coordinado por la Fundación Rust, que es apoyada y financiada por las grandes empresas del software.

La mayores influencias en el lenguaje Rust provienen de SML, OCaml, C++, Cyclone, Haskell y Erlang. Desde su primera versión estable de enero de 2014, Rust se utiliza en los desarrollos de grandes empresas, como Amazon, Discord, Dropbox, Facebook (Meta), Google (Alphabet) y Microsoft. Actualmente, Rust se utiliza para desarrollar el nucleo de sistemas operativos como Linux, Windows y Android.

Rust no es un lenguaje orientado a objetos propiamente dicho, aunque se pueden emular muchas de las técnicas que se utilizan en ese paradigma de programación. La programación funcional sí casa mejor con la filosofía del lenguaje Rust, que sin ser un lenguaje funcional estricto, permite realizar una programación funcional eficiente. Puede ser una buena herramienta para pasarse a la programación funcional.

Por supuesto, todas estas ventajas que ofrece Rust son a costa de una curva de aprendizaje un poco más pendiente al principio. Programar en Rust requiere más esfuerzo que programar en otros lenguajes, como por ejemplo, Python. Pero los resultados que se obtienen compensan el esfuerzo dedicado y, a pesar de que se trata de un lenguaje relativamente nuevo, dispone ya de una amplia librería de utilidades que facilitan la programación dentro de cualquier ámbito.

En este libro se da una extensa y profunda introducción al lenguaje Rust. El libro no está pensado para personas que no sepan programar, sino más bien para programadores de otros

lenguajes que quieran aprender a utilizar Rust. Se abordan y se explican detalladamente numerosos temas relacionados con la programación en lenguaje Rust. Se utilizan cientos de ejemplos de código que profundizan en numerosos aspectos del lenguaje. No obstante, es imposible en un solo libro abarcar la inmensidad de un lenguaje como este. Inevitablemente, algunos temas se explican de manera somera, a la espera de poderse ampliar en libros más específicos que el autor tiene en preparación.

## 3. Estructura y contenido del libro

El libro consta de 19 capítulos. El primer capítulo se dedica a explicar cómo instalar Rust, cómo poner a punto el entorno de desarrollo, y cómo hacer el típico programa *Hola mundo*. Los tres siguientes capítulos abordan las variables, los tipos de datos que ofrece el lenguaje y cómo implementar bifurcaciones, bucles y funciones. Se supone que el lector ya ha programado antes, por lo que se incide en aspectos como la diferencia entre una variable y el valor que guarda, o la forma de almacenar los valores en memoria. También se explica aquí la diferencia en programación entre una *sentencia*, que no devuelve ningún valor, y una *expresión*, que da lugar a un resultado. En Rust, a diferencia de otros lenguajes, las asignaciones son sentencias y las bifurcaciones son expresiones. También se explica la bifurcación *match*, característica del lenguaje Rust, que realiza una comprobación extensiva de que se han valorado todos los posibles resultados. Es una característica que sabrán apreciar los programadores.

El capítulo 5 explica en profundidad el concepto de *propiedad de los valores*; ésta es sin duda la característica distintiva de Rust respecto de otros lenguajes para el tratamiento de los valores almacenados en las variables. Aunque al principio puede resultar difícil acostumbrarse a operar en términos de *propiedad de los valores*, se trata de un concepto clave en la seguridad del código generado. El capítulo 6 se dedica a los *arrays* y los temas relacionados con cadenas de caracteres. El capítulo 7 introduce las *estructuras*, que son similares a los objetos de los lenguajes de programación orientados a objetos.

El capítulo 8 explica las enumeraciones. Se explica en él la enumeración *Option*, que da pie a otra de las características distintivas de Rust: la ausencia de valor nulo. En Rust no hay valores nulos, y esto le dota de un nuevo nivel de seguridad del código. El valor *null* lo inventó Tony Hoare, mientras trabajaba en el desarrollo del lenguaje *Algol*. Esto es lo que dijo unos años después:

> *Yo lo llamo mi error de mil millones de dólares... En ese momento, estaba diseñando el primer sistema completo de tipos para referencias en un lenguaje orientado a objetos. Mi objetivo era garantizar que todos los usos de las referencias fueran absolutamente seguros, con la verificación realizada automáticamente por el compilador. Pero no pude resistir la tentación de poner una referencia nula, simplemente porque era muy fácil de implementar. Esto ha llevado a innumerables errores, vulnerabilidades y bloqueos del sistema, que probablemente han causado mil millones de dólares en dolor y daños en los últimos cuarenta años. – (Tony Hoare, inventor de ALGOL)*

Tras hacer repaso en el capítulo 9 a algunas de las colecciones existentes en Rust, como los vectores o los *hash-maps*, también habituales en otros lenguajes, el capítulo 10 explica el concepto de *vida útil* de las referencias, otro de los conceptos distintivos de Rust y que aportan máxima seguridad en el manejo de la memoria. Se evita así la posibilidad de que existan *punteros colgados* (*dangling pointers*) apuntando a valores que ya han sido retirados de memoria. Es uno de los problemas de seguridad más recurrentes en lenguajes como C o C++, y que dan lugar a algunas de las mayores quiebras de seguridad existentes.

Los capítulos 11 y 12 explican la manera de utilizar los *traits* y tipos de datos *genéricos*, que proporcionan a Rust la posibilidad de trabajar de manera muy similar a la de los lenguajes

orientados a objetos. El capítulo 13 se dedica a las *closures* y los *iteradores*, y se introduce al lector en la programación funcional. Los conceptos anteriores se completan en el capítulo 14 con la explicación de los diferentes tipos de punteros que se pueden utilizar.

En los capítulos 15 al 18 se habla de cómo dividir los programas en módulos, como hacer librerías de funciones, la gestión de la entrada y salida por pantalla y a ficheros y de cómo utilizar la extensa librería de utilidades externas existentes para Rust.

Por último, el capítulo 19 explica el funcionamiento del sistema integrado en Rust para documentar e implementar test en los programas. Rust permite utilizar test unitarios, test de integración y test de documentación, lo que favorece enormemente la programación con metodología *Test-Driven Development (TDD)*, minimizando los errores al codificar.

**Sobre el/los autor/es:**
*Nombre:* Santiago Higuera de Frutos
*Correo electrónico:* santiago.higuera@upm.es
*Institución:* Universidad Politécnica de Madrid