Programación científica
Técnicas y fundamentos para el desarrollo de software

Pablo Alcain / Cecilia Jarne /
Rodrigo Lugones / María Graciela Molina

Universidad Nacional de Quilmes Editorial
Bernal, 2022

Colección Nuevos enfoques en ciencia y tecnología
Dirigida por Diego Golombek

Programación científica: técnicas y fundamentos para el desarrollo de software / Pablo Alcain... [et al.]. - 1a ed. - Bernal: Universidad Nacional de Quilmes, 2022.
210 p.; 22 x 15 cm. - (Nuevos enfoques en ciencia y tecnología / Diego Golombek)

ISBN 978-987-558-795-3

1. Ciencias Tecnológicas. 2. Nuevas Tecnologías. 3. Lenguajes de Programación. I. Alcain, Pablo
CDD 005.1

Universidad Nacional de Quilmes
Roque Sáenz Peña 352
(B1876BXD) Bernal, Provincia de Buenos Aires
República Argentina

ediciones.unq.edu.ar
editorial@unq.edu.ar

ISBN: 978-987-558-795-3

Queda hecho el depósito que marca la ley 11.723
Impreso en Argentina

ÍNDICE

LOS AUTORES

Pablo Alcain. Físico de la Universidad de Buenos Aires (UBA). Hizo su doctorado en Astrofísica Nuclear, estudiando la estructura interna de las estrellas de neutrones a través de técnicas de dinámica molecular computacional. Se especializó en técnicas de cómputo de alto desempeño en placas gráficas y en cálculo distribuido, compatibilizando código escrito en C, Assembler, CUDA y Python. Actualmente se dedica a la ciencia de datos y al desarrollo de software para *machine learning*.

Cecilia Jarne. Realizó su doctorado en Física en el Instituto de Física de La Plata (IFLP) y el Departamento de Física de la Universidad Nacional de La Plata. Su experiencia en investigación se basa en análisis de grandes datos, primero en física de rayos cósmicos de alta energía y luego durante el posdoctorado en el Instituto de Física de Buenos Aires (IFIBA-UBA) analizando cantos de pájaros. Posee diversas publicaciones en revistas indexadas relacionadas con ambas áreas. Actualmente es investigadora asistente en Conicet y se encuentra trabajando en redes neuronales recurrentes y sistemas complejos. También es docente de grado y posgrado en la Universidad Nacional de Quilmes (UNQ) y semanalmente da clases en una escuela técnica (que fue su escuela), la EEST 2 de Quilmes.

Rodrigo Lugones. Es doctor en ciencias físicas de la Universidad de Buenos Aires y científico de datos en Despegar. Tiene más de diez años de experiencia docente en distintos ámbitos universitarios, tanto en materias de grado como de posgrado. En su doctorado, trabajó en turbulencia en plasmas astrofísicos, aplicando computación de alto desempeño para la realización de simulaciones numéricas en clusters. Desde 2018, trabaja en inteligencia artificial, principalmente en temáticas de visión por computación y aprendizaje de máquina más tradicional.

Es cofundador y CTO de Ninja IA, una empresa dedicada a proveer soluciones de inteligencia artificial.

María Graciela Molina. Es docente de grado e investigadora del Departamento de Ciencias de la Computación de la Facultad de Ciencias Exactas y Tecnología de la Universidad Nacional de Tucumán (FACET-UNT). Es docente de posgrado del Doctorado en Ciencias Exactas e Ingeniería y de la Maestría en Métodos Numéricos y Computacionales, ambos de la FACET-UNT. Actualmente, es directora del Laboratorio de Computación Científica (LabCC) y del centro de Space Weather de la FACET-UNT. Además, es docente de posgrado de la UNQ. Es Investigadora del Conicet. Desarrolla su investigación en el área de inteligencia artificial aplicada a Space Weather, sensado remoto y sistemas de radares.

PRÓLOGO

El Abdus Salam International Centre for Theoretical Physics, también conocido por sus siglas ICTP, es un instituto de investigación científica auspiciado por la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (Unesco), el Organismo Internacional de Energía Atómica (IAEA) y el gobierno italiano. Sus instalaciones se ubican en la ciudad de Trieste, en Italia, a un costado del castillo de Miramar.

El ICTP es un lugar maravilloso por varias razones que exceden la geografía de sus instalaciones. La institución fue fundada en 1964 por el físico pakistaní y premio Nobel Abdus Salam, con el objetivo de promover para los científicos de los países en desarrollo la educación continua y las habilidades que necesitan para el desarrollo de disciplinas centradas en la física teórica y la matemática. Con el tiempo, estas disciplinas se fueron ampliando. El ICTP ha sido una herramienta creada con la intención de detener la fuga de cerebros científicos del mundo en desarrollo.

Estos párrafos anteriores pueden sonar a una historia poética o a un panfleto publicitario, pero afortunadamente son ciertos. El ICTP es el corazón de la historia de muchos investigadores que vivimos en países en vías de desarrollo como la Argentina. Acá comienza nuestra historia como grupo de colaboradores, luego de amigos, y de cómo surge la necesidad de escribir este libro.

El ICTP tiene varios eventos anuales internacionales de diversas disciplinas. En particular, tiene una escuela de Técnicas de Programación Científica y otra de Programación en Paralelo, ambas intensas, mentalmente muy demandantes y extremadamente satisfactorias. Pablo Alcain, Graciela Molina y yo (Cecilia Jarne) nos conocimos en el primero de estos eventos. En ese entonces, yo acababa de terminar mi doctorado en Física en la Universidad Nacional de La Plata (UNLP),

y estaba por comenzar mi posdoctorado en la Universidad de Buenos Aires (UBA). Por su parte, Pablo realizaba su doctorado en la UBA y Graciela, el suyo en la Universidad Nacional de Tucumán (UNT).

El evento del ICTP, antes de la pandemia, se realizaba cada dos años en Trieste, y los años intermedios en una locación internacional. A nosotros tres nos tocó conocernos en 2015 en la ciudad de San Pablo, Brasil. Entre líneas de código y caipiriñas, comenzamos a hablar de cuánta falta hacía promover estos conocimientos "en casa".

En ese curso también conocimos a otro colega, Pablo Echeverría, quien trabajaba en ese momento en el Servicio Meteorológico Nacional argentino. Él nos proporcionó la invitación clave para dar un curso allí, ese mismo año, el cual dio comienzo a una serie sin pausa de *workshops*, capacitaciones y finalmente cursos de posgrado que seguimos dando, para que principalmente estudiantes de doctorado e investigadores jóvenes puedan suplir las carencias de formación en técnicas de programación en las distintas carreras de grado.

Rodrigo Lugones y Pablo Alcain son amigos desde la escuela. Ambos estudiaron juntos en la secundaria y luego fueron juntos a la UBA, donde estudiaron la Licenciatura en Física. En mi caso, a Rodrigo lo conocí cuando por azar (aunque también por decantamiento) compartimos la Escuela de Programación en Paralelo del ICTP. Cuando Pablo Echeverría migró a España, hacia fines del 2016, Rodrigo ocupó su lugar, llegando a la conformación actual del grupo.

El vínculo de colaboración y de amistad entre los cuatro ha hecho que nos encontremos frecuentemente (cuando podemos) para escribir algún trabajo, para dar una charla, para dictar un curso o, como en este caso, para escribir un libro. Nos ha unido también el objetivo de universalizar el conocimiento: hemos rotado los cursos por diversos lugares del país (Buenos Aires, Tucumán, Quilmes, Córdoba), hemos puesto énfasis en la federalización de los participantes. Hemos sentido un gran compromiso por involucrar a más mujeres en el ámbito de la programación científica. Y si bien partimos de distintas formaciones (tres físicos pero trabajando en temáticas muy distintas, y Graciela, con su origen en informática), los cuatro frecuentemente compartimos nuestras experiencias, pues los desafíos a los que nos enfrentamos suelen ser parecidos: pensar en términos de código, plantear el mejor algoritmo para analizar datos, hacer una simulación o resolver un problema son situaciones comunes en ciencia a lo largo de muchas disciplinas.

Hay muchas personas e instituciones fuertemente responsables de la existencia de este libro. En primer lugar, Ivan Girotto, del ICTP, nos ha incentivado de distintas maneras para construir y dar nuestros cursos en la Argentina, y para los científicos de Latinoamérica en general. Cada vez que vamos por algún motivo al ICTP nos quiere ver, pregunta por los cuatro y nos acompaña moralmente. Eso nos llena de alegría.

Otra persona clave es Diego Golombek, quien estuvo allí para leer la propuesta de nuestro libro. Diego, además de hacer ciencia de calidad en su laboratorio de cronobiología en la Universidad Nacional de Quilmes (UNQ), también se dedica con gran compromiso a fomentar la formación científica y tecnológica en todos los niveles posibles; esto es un trabajo sumamente valioso.

Asimismo, la UNQ, mi lugar de trabajo, donde desarrollo mi actividad científica, nos ha brindado su apoyo incondicional a lo largo de varios años, otorgándonos un espacio formal para el desarrollo de proyectos científicos y educativos como este. Estamos completamente agradecidos por el espacio que nos dan.

Por supuesto, también queremos contarles que para que un libro quede bien, detrás de los autores hay un trabajo de edición muy extenso. A la Editorial de la UNQ también le agradecemos mucho.

Esta es una breve historia de quiénes somos, y de por qué y cómo decidimos escribir este libro. Intentamos que los contenidos sean perdurables, más que una moda, que ofrezcan una forma de pensar el programar y no simplemente cómo utilizar herramientas específicas. Deseamos que les permitan construir mejor conocimiento científico a través del uso del código de acceso abierto y eficiente. Y que al escuchar "eficiencia", no solo piensen que el programa corra rápidamente, sino también que se desarrolle, se mantenga, se use, se modifique y se adapte al nuevo *hardware* rápidamente.

Esperamos que puedan disfrutar leyendo este libro tanto como nosotros al escribirlo, y que puedan implementar algunas de las ideas que aquí compartimos.

Pablo Alcain / Cecilia Jarne / María Graciela Molina / Rodrigo Lugones

INTRODUCCIÓN

El uso de las computadoras y lenguajes de programación se han vuelto cada vez más imprescindibles para la investigación de punta en muchas áreas científicas y en el desarrollo de tecnología. La mayoría de las carreras de grado no contempla una formación específica en estos temas como el análisis computacional de datos o el desarrollo de simulaciones. Como consecuencia, el ecosistema de programación y desarrollo de software científico está alejado de las herramientas actuales que, en el mejor de los casos, se aprenden de fuentes muy diversas con calidad muy dispar y de manera informal.

Este libro tiene como objetivo ser un aglutinante de las distintas herramientas y habilidades que debe tener un científico para poder realizar las tareas que requieran el uso de una computadora, buscando transmitir una filosofía de trabajo que trascienda las herramientas concretas que se utilicen. El título de este texto no es casual, y es alrededor de la definición de *calidad* que se construye esta filosofía de trabajo. Un software de calidad tiene que tener cuatro características fundamentales: ser modular, reutilizable, verificable y documentado. Otras características deseables están relacionadas con la escalabilidad, la portabilidad y el ser amigable con el usuario.

La *modularidad* de un software refiere a separar la funcionalidad de un programa en distintas unidades, independientes entre sí, de modo de que cada una de ellas pueda realizar una y solo una tarea. La ventaja más evidente es que se puede dividir un proyecto en varios proyectos distintos, cada uno con un objetivo concreto y más rápido de alcanzar. De este modo, además, se vuelve más fácil la colaboración entre varios desarrolladores, ya que pueden trabajar en unidades separadas solo acordando cómo se comunican entre ellas.

El software es *reutilizable* si sus partes pueden ser utilizadas en otro programa sin ser escritas nuevamente. Evidentemente, un software re-

utilizable tiene que tener cierto grado de modularidad, pero no todo software modular es reutilizable: además, hay que tener precaución en que las interfaces sean lo más simples posibles y dependan de estructuras de datos estandarizadas.

La *verificabilidad* es un concepto fundamental en el desarrollo de la ciencia, por lo que *debe serlo también* para el desarrollo de software científico. Para que un software sea verificable tenemos que poder estudiar cada una de sus partes por separado, analizando casos de prueba generales que pueden incluir circunstancias que inicialmente no fueron pensadas. De esta manera, cuanto más modular y reutilizable sea el software, sin duda será más verificable. Pero al igual que antes, no todo software modular y reutilizable es verificable: para que esto ocurra, es necesario tener bien claro qué tipo de resultados esperamos de cada módulo y la posibilidad de contrastarlos con datos conocidos.

Finalmente, un software de calidad tiene que ser *documentado*. Esta documentación tiene dos aristas distintas y complementarias: por un lado, la destinada al desarrollador (es decir, las personas que posteriormente van a modificar y extender el programa), y por el otro, la que está orientada al usuario (quien no tiene por qué conocer detalles del funcionamiento interno del programa, sino saber fácil y rápidamente cómo utilizarlo). Documentar adecuadamente un software no es una tarea sencilla y requiere de mucha práctica, pero es fundamental para que el código sea accesible al resto de las personas. La documentación adecuada es una de las etapas esenciales que se deben tener en cuenta tendientes a lograr una interfaz amigable.

Estas características del software refieren a un objetivo más grande: que su desarrollo sea *colaborativo*. Esta tendencia requiere pensar el desarrollo, prácticamente desde el comienzo, en la clave de calidad que mencionamos. La idea no es nueva, y es de hecho como se desarrolla en la actualidad gran parte de los programas que usamos todos los días: es el movimiento de *software libre*.

Para poder desarrollar un software de calidad, es necesario conocer y utilizar las herramientas adecuadas, pero es igualmente necesario poder *abstraerse* de ellas. El universo de la programación es muy dinámico y estas herramientas cambian muy rápidamente. La única forma de mantenerse actualizado es comprendiendo cuál es el rol de cada herramienta dentro de la filosofía de trabajo y poder suplirla o identificar herramientas superadoras. Su uso forma parte de un conjunto de habilidades que llamamos *soft skills* o *habilidades blandas* en el ámbito de la programación: no son fundamentales y no cambian directamente la funcionalidad del software. De hecho, en la actualidad, gran parte de los programas científicos son realizados por desarrolladores que no siguen estos preceptos, y no por eso diríamos que no sirven. No obstante, sí complican su inserción a un universo de desarrollo colaborativo que permitiría extender su funcionalidad y su uso a una comunidad más amplia. Justamente, que no sean fundamentales en lo inmediato es lo que hace que muchos desarrolladores elijan no usarlas. Sin embargo, su potencial se desarrolla completamente cuando dejamos de verlas como herramientas separadas y las entendemos como parte de un engranaje más grande que lleva a desarrollar software colaborativo y de calidad. A lo largo de este libro describiremos estas herramientas, haciendo énfasis constantemente en cómo forman parte de un ecosistema para el desarrollo de software de calidad.

El libro se estructura de la siguiente manera: veremos algunas herramientas fundamentales en los primeros seis capítulos. Entre otras, estas son: Python como lenguaje de *scripting* para desarrollar un entorno amigable para los usuarios; Git como herramienta de control de versiones; Doxygen y Sphinx como herramientas de documentación; gdb, valgrind y gprof para el *debugging* y el *profiling* de software, y MPI y OpenMP como herramientas de paralelización. En el último capítulo describiremos ciertos ejemplos de la vida real en los que se utilizan estas herramientas: tanto ejemplos de personas trabajando en estos ámbitos y el uso de las herramientas, como el desarrollo paso por paso de un código que cumpla las características fundamentales del software colaborativo.