**El impacto de los *frameworks* en los programadores novatos.**

Fidel Aguilar

**Introducción**

En poco más de una década, la programación y desarrollo de software pasó de girar en torno a prácticamente solo lenguajes base o “vanilla” a recurrir a docenas de *frameworks* especializados, que prometen acelerar la creación de productos. Según la *Stack Overflow Developer Survey 2024*, ***React*** es utilizado por aproximadamente 40 % de los profesionales y ***Angular*** por el 18 % cifras que cambian año a año conforme aparecen nuevas opciones. Esta gran cantidad de herramientas genera oportunidades laborales, pero también afecta a los más novatos, pues se les pide tener fluidez en estas nuevas áreas de conocimiento. La proliferación masiva de *frameworks* específicos dificulta la curva de aprendizaje de los nuevos programadores, porque introduce complejidad y dependencias que oscurecen los principios básicos de la programación y fomentan un aprendizaje superficial y poco transferible.

**Desarrollo**

Los estudios sobre enseñanza de la programación confirman que la complejidad estructural aumenta la carga mental para los nuevos estudiantes. “*Novice learners generated greater cognitive load when processing embedded structures compared to sequential structures*” (Hao et al., 2023). Cuando el primer contacto con el código ya rebasa la capacidad de procesamiento, el aprendizaje se vuelve un ejercicio de resistencia más que de comprensión. Este exceso cognitivo puede causar que el aprendizaje inicial de los diversos lenguajes de programación tenga una dificultad alta.

Empezar a aprender lo básico en el ámbito de la programación no es tan fácil como en otros ámbitos: “*Although introductory programming courses are considered to cause significant levels of student anxiety and frustration, and student engagement levels in computing are benchmarked as among the lowest of any discipline…* “(von Hausswolff, 2022). Ese bajo en el compromiso de nuevos estudiantes y curiosos se agrava cada que sale un nuevo “kit de herramientas”, o *framework*, que requiere de nuevo aprender los comandos y sintaxis de este nuevo “pseudolenguaje”.

Los propios desarrolladores experimentados reconocen la dificultad y poca practicabilidad de tener que manejar diversos *frameworks* para cada proyecto. “*We could dive deeply into React, and then switch to something else entirely for the next project and have to spend six months learning that instead of developing deep expertise in one technology*” (Finley, 2018). El tiempo que podría invertirse en dominio conceptual se consume en iterar más ciclos de aprendizaje tedioso de un nuevo espacio de trabajo. A eso también se le debe agregar el aumento en dificultad de cada nuevo *framework* que es hecho público.

Incluso la documentación “pro-framework” admite la barrera de entrada de algunos *frameworks*: “*Angular’s comprehensive nature can make it initially more complex than React*” (Full Scale, 2024). La comparación constante entre los distintos *frameworks* llega a abrumar al novato, porque cada nuevo espacio de trabajo tiene sus ventajas, desventajas, similitudes y totalmente diferente sintaxis el uno del otro. Con tantas posibilidades de entornos de trabajo distintos, ¿Vale más la pena especializarse en uno sólo?

La respuesta a esa pregunta es variada, pues algunos proyectos de trabajo piden dominar un *framework*, mientras que otros piden a uno ser fluido en más de uno a la vez, por el constante cambio de base en la arquitectura de sus proyectos. Sin embargo, a pesar de que por lógica la curva de aprendizaje debería ser cada vez más fácil por la repetitividad de ciertos patrones en los distintos *frameworks*, la verdad es que saber de uno sólo, no garantiza adaptarse a otros entornos o lenguajes “*Software developers typically know at least one programming language, but knowledge of multiple programming languages enhances their experience and competition in the job market*” (Sartore, 2024). A pesar de ello, cada vez se normaliza más la fluidez en varios lenguajes y espacios de trabajo para un puesto de desarrollador *Junior*. En los avisos laborales de hoy se percibe una presión creciente por manejar varias pilas tecnológicas a la vez. Sin embargo, la portabilidad real de esas destrezas no es tan automática como se presume. Tal como resume el portal ComputerScience.org: “*Software developers typically know at least one programming language, but knowledge of multiple programming languages enhances their experience and competition in the job market*” (Sartore, 2024). Esa mejora competitiva ocurre sólo si existen bases conceptuales sólidas; de lo contrario, aprender un nuevo framework equivale a memorizar otra lista de comandos desde cero.

Cada año, las tecnologías y *frameworks* más demandados en el campo de la programación cambian, en función de las necesidades del mercado y la evolución de la industria. Según la encuesta anual de *Stack Overflow Developer Survey (2023)*, los *frameworks* más populares en ese año fueron React con un 40.6%, seguido de Next.js (15.6%) y Angular (17.4%), lo que refleja una fuerte preferencia por tecnologías JavaScript modernas. Esto representa un cambio respecto a años anteriores como 2022, donde Angular y jQuery aún ocupaban posiciones más altas, pero fueron desplazados por herramientas más modernas y eficientes para desarrollo web. Esto causa que un desarrollador esté al tanto de cualquier novedad respecto al mundo de la programación y aunque esto parezca algo razonable, cada nueva tendencia en el uso de un nuevo *framework* se resume en horas de estudio, prácticas y proyectos para dominar esta nueva herramienta.

Cada corporación grande parece haber levantado su propio “templo” tecnológico y, para entrar, hay que dominar el texto sagrado correcto. Google impulsa la mayor parte de sus productos web con Angular y suele publicitar que el mismo buscador se sirve de ese marco (Google Developers, 2025). Meta, por su lado, ha convertido a React en el corazón de su cultura de ingeniería, al punto de que miles de desarrolladores internos lo usan a diario (Meta Engineering, 2024). En Asia, Alibaba reforzó su ecosistema con Vue porque, según sus equipos, les permite crear interfaces para millones de usuarios con mayor rapidez (Alibaba Front-end Group, 2023). Cuando cada gigante exige maestría absoluta en *su* stack, el programador junior se ve obligado a “casarse” con un solo *framework* si quiere aspirar a puestos de alto nivel; de lo contrario, corre el riesgo de quedarse flotando entre tecnologías sin la profundidad que las compañías de referencia esperan. A veces, incluso el aprendizaje de nuevas tecnologías no es tan fácil a pesar de la motivación y el tiempo.

Los frameworks modernos no sólo exigen dominar nuevas APIs; también obligan a descargar paquetes que pesan cientos de megabytes, compilar dependencias y sincronizar repositorios en la nube. Para quienes viven con conexiones inestables o equipos modestos, ese simple “npm install” puede convertirse en una tarde perdida. Como advierte Marc Alan Sperber, “*one-third of the world’s population has never even used the internet, and many more than one-third don’t have reliable access*” (Sperber, 2023). Esa carencia se refleja en el aula: un estudio sobre educación a distancia halló que “*43 % de los docentes señalaron la falta de acceso confiable a Internet o Wi-Fi como uno de los problemas más desafiantes*” (American Foundation for the Blind, 2022). Cuando la misma infraestructura es un obstáculo, los estudiantes con menos recursos necesitan el doble de tiempo y esfuerzo para llegar al mismo punto de partida que sus pares mejor conectados, lo que ensancha la brecha de oportunidades desde el primer día.

**Conclusión**

Los frameworks han demostrado ser poderosos aliados para acelerar proyectos y satisfacer las cambiantes demandas del mercado. Sin embargo, su abundancia también ha creado barreras que pocos principiantes pueden superar sin tropiezos. A lo largo de este ensayo, hemos visto cómo la sobrecarga cognitiva, la ansiedad generada por cada nuevo pseudolenguaje, la volatilidad de las tendencias y la dependencia de la infraestructura amplían la brecha entre quienes tienen una base sólida y quienes apenas dan sus primeros pasos. En la práctica, especializarse ciegamente en una sola pila puede allanar el camino para algunas empresas, pero también expone a los desarrolladores a la obsolescencia y al estancamiento profesional si el mercado cambia de dirección.

El reto, por lo tanto, no reside en eliminar los frameworks ni ignorar las demandas del sector, sino en conciliar su uso con una formación que priorice los fundamentos del pensamiento algorítmico, la comprensión de la web tradicional y la capacidad de aprender a aprender. Solo así los futuros programadores podrán cambiar de herramienta sin tener que empezar desde cero y, lo que es más importante, sin verse limitados por limitaciones técnicas o de acceso. Si transformamos los frameworks en un complemento, no en una muleta, construiremos una comunidad de desarrolladores más resiliente, inclusiva y preparada para los inevitables cambios tecnológicos.

**Bibliografía**

Alibaba Front-end Group. (2023, 17 mayo). *Vue.js in action: Industries and use cases for the fast and flexible front-end framework*. Medium. https://medium.com/@computools/vue-js-in-action-industries-and-use-cases-for-the-fast-and-flexible-front-end-framework-d0ae3cb23b80

American Foundation for the Blind. (2022). *Access and Engagement III: Reflecting on the impacts of the COVID-19 pandemic on the education of children who are blind or have low vision*. https://www.afb.org/research-and-initiatives/education/access-engagement-iii

Finley, K. (2018, 19 septiembre). *For JavaScript developers, more choices mean hard choices*. Wired. <https://www.wired.com/story/javascript-developers-more-choices-mean-hard-choices/>

Full Scale. (2024, 22 octubre). *Angular vs. React: Which should you choose for frontend tech?* <https://fullscale.io/blog/angular-vs-react/>

Gechev, M. (2024, 22 mayo). *Angular v18 is now available!* Angular Blog. https://blog.angular.dev/angular-v18-is-now-available-70e05acc1b28

Hao, X., Xu, Z., Guo, M., Hu, Y., & Geng, F. (2023). The effect of embedded structures on cognitive load for novice learners during block-based code comprehension. *International Journal of STEM Education, 10*(42). <https://doi.org/10.1186/s40594-023-00432-9>

Meta Engineering. (2024, 2 octubre). *React at Meta Connect 2024*. <https://engineering.fb.com/2024/10/02/android/react-at-meta-connect-2024/>

Sartore, M. (2024, 20 septiembre). *What is a software developer? Skills and career paths*. ComputerScience.org. <https://www.computerscience.org/careers/software-developer/>

Sperber, M. A. (2023, 6 marzo). *SunSPOT model provides education access to areas with unreliable internet*. ASU News. https://news.asu.edu/20230306-global-engagement-sunspot-model-provides-education-access-areas-unreliable-internet

Stack Overflow. (2024). *2024 Stack Overflow Developer Survey*. https://survey.stackoverflow.co/2024/

Stack Overflow. (2023). *2023 Stack Overflow Developer Survey*. https://survey.stackoverflow.co/2023/

von Hausswolff, K. (2021). Practical thinking while learning to program – novices’ experiences and hands-on encounters. *Computer Science Education, 31*(4), 319-342. <https://doi.org/10.1080/08993408.2021.1953295>