CAPÍTULO CINCUENTA Y SEIS

¡NI SE TE OCURRA APRENDER A PROGRAMAR!

START WALKING
REPEAT {
WAIT ORDER
IF ORDER = "IZQ" THEN LEFT
WAIT ORDER
IF ORDER = "DER" THEN RIGHT
}

¿Viste cuando comprás un horno a microondas, lo desempacás, tirás ese librito a la basura y empezás a calentar comida? "Ese librito" es el manual de instrucciones. Hay estudios que dicen que el 68 por ciento de la población los tira. Del tercio que los guarda, el 7 por ciento los lee. De ellos, solo un 10 por ciento es capaz de entender las instrucciones y seguirlas al primer intento: 10 por ciento del 7 por ciento lee y entiende las instrucciones; tal vez la mitad (no estaba en el estudio) las obedece. El mismo estudio volcó la información de esos manuales en computadoras y llegó a la conclusión de que el 100 por ciento las entiende y sigue al primer intento.

-Decidite, Leo, estás pensando en irte de Staples desde hace años -me dijo un amigo ansioso un día que hice un comentario, sin entender que un proceso de decisión puede tomar mucho tiempo.

Desde afuera se ve ambigüedad y una gran contradicción. Pero lo importante es que adentro tiene que haber aprendizaje constante, evolución. Una puerta que se cierra muy lento, mientras se abren otras. Esa puerta que chirría en forma de comentarios que no siempre se entienden.

La misma contradicción que debe sentir quien me escuchó hace un tiempo en la radio recomendando estudiar programación, defendiendo incluso que los desempleados deberían hacerlo porque, además, es gratis. Sentí que no me entendían, me frustré un poco.

Nuevamente una sensación negativa empujó mi cabeza a pensar soluciones. Y entendí que no podemos ser todos programadores; algunos no tenemos la capacidad pero, más importante, no necesitamos tantos programadores.

Si queremos convivir con las computadoras, tenemos que entenderlas.

Eso sí, necesitamos algunos conceptos que se aprenden programando, para poder entender el mundo del futuro.

Toda tarea repetitiva puede automatizarse: cuando leí lo que este programador (John) había hecho me pareció increíble. Tenía un par de *conference calls* por semana. *Conference calls*, probablemente la enfermedad más contagiosa en una corporación. Se expande tan rápido que no tiene traducción y ni siquiera sabemos si es femenino o masculino. John había hecho un software que "escuchaba" la llamada y, cuando alguien decía su nombre, contestaba: "Sí, no escuché bien, ¿me lo repetís?", en una de cinco variantes. En ese momento, interrumpía la pantalla de John -en la que estaba haciendo otra cosa- con el texto de los últimos dos minutos de conversación, para que se pusiera al día y pudiera responder. Magia emprendedora, dentro de una corporación. Cada vez hay más ejemplos, incluyendo los que le avisan a la esposa que va a llegar tarde con una excusa al azar.

Tus tareas repetitivas en el trabajo van a desaparecer.

Lo más curioso -no gracioso- es que seguro que si se enteran, a John lo echan en vez de hacerle un monumento. John es un pseudónimo, no se preocupen.

> Aprendé del más vago.

Hace poco supe de otro programador que, si algo le toma más de 30 minutos, prefiere hacer un software que realice esa tarea antes que efectuar la tarea misma. Invierte tiempo, en vez de gastarlo.

Prototipado: una vez envié uno de mis mails semanales con un error. Enseguida alguien me avisó, pero ya era demasiado tarde. Diez mil personas lo tenían en sus casillas. Y yo había descuidado uno de mis principios: testear en chiquito antes de lanzar algo nuevo. Este concepto se aplica básicamente a cualquier idea que tengamos.

> Hay quien dice que es 95 por ciento transpiración y 5 por ciento inspiración. Falta el cien por ciento de testeo.

En el futuro imagino una profesión que sea "Diseñador de tests para cualquier cosa".

> Tu opinión no me interesa -testealo y charlamos.

Eficiencia en el código: no podría haberme destacado en aquel trabajo que tuve ordenando millones de declaraciones juradas impositivas sin saber programar. Empecé ordenando como podía, hasta que recordé el método *bubble sort*. Primero, separé en mitades; luego, en mitades de nuevo; y, así, sucesivamente. Ordenar muy poquitos documentos era súper rápido y no necesitaba manos gigantes. Para programar es clave entender que, cuanto más simple y eficiente el código, mejor: el resultado será más rápido, más fácil de revisar y ocupará menos espacio. Un concepto clave aquí son las "librerías" (paradójico para mí viniendo de Officenet): los "lugares" en donde se coloca el código cuando alguien programa algo que anda bien y otros pueden necesitar.

> No programarás lo ya programado.

Eficiencia en el espacio: ¿qué ocupa menos espacio, la foto de un círculo negro de 5 centímetros de diámetro o la descripción "círculo negro de 5 centímetros de diámetro"? Aunque la capacidad de memoria de las

computadoras sigue creciendo a la tasa de la Ley de Moore, cuanto menos información innecesaria mejor funciona todo. El ejemplo más habitual es el del Excel: aprendí que los datos nunca tienen que repetirse. Deben estar en un solo lugar y con referencias de cualquier otro archivo a ese dato original. Hasta hoy lo uso cuando publico: muchos me dicen que debería tener un blog, sin entender que para mí no hay que repetir la información en la web. O publico en LinkedIn o en un blog. Y para mí los blogs están muertos.

Poka Yoke: cuando entendí que había un término japonés para "a prueba de boludos", no podía creerlo. Los usuarios no son programadores. Nuestros clientes no piensan como nosotros, tenemos que hacerles la vida más fácil. Eliminemos ese formulario, pongamos un simbolito en los autos que diga de qué lado se carga el combustible y dejemos de pedir el "sexo" cuando solo necesitamos el número de documento. Lo del combustible ya está, buscá la flechita en el indicador de combustible en el tablero :-)

Obediencia absoluta: en 1982 aprendí a programar; algunos se acordarán de la tortuguita del lenguaje Logo. Al inicio de este capítulo repliqué, en pseudo-código (un lenguaje de programación inventado), parte de un juego que hice en esa época. No me fue fácil detectar, a mis 12 años, el error que cometí. Les aseguro que la tortuguita no hacía lo que yo esperaba: hacía lo que yo escribí. Me enojé, apagué la computadora y le pedí ayuda a un profesor. Todos los programas hacen lo que le ordenó quien lo ideó, textualmente, sin dudar. Mientras la inteligencia artificial no pase cierto punto esto será así: computadoras, celulares y robots obedeciendo ciegamente.

Corolario I: cuando estudié mecanografía, me enseñaron que "O" y "0" son lo mismo. Para una computadora es pecado mortal.

Corolario II: Para cuidarnos de que esa obediencia nos dañe, Isaac Asimov creó estas Tres Leyes de la Robótica:

1. Un robot no hará daño a un ser humano ni, por inacción, permitirá que un ser humano sufra daño.

2. Un robot debe hacer o realizar las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley.

3. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley.

Corolario III: La energía es clave. Si desenchufás o desconectás, deja de andar.

Corolario IV: Mr. Spock era genial.

Capítulo 56 de *Soy Solo* (mayo de 2018).

Escuchalo con un comentario al final:
leopiccioli.com/libros/soysolo/56