Del Colegio a la Programación:

Un Viaje de Desaprendizaje y Transformación Educativa

Luis Miguel Triana Rueda

Socorro, Santander, Colombia

Del Colegio a la Programación: Un Viaje de Desaprendizaje y Transformación Educativa

Diciembre de 2025. Fin de semestre. Un amigo de otra universidad me pregunta sobre biología, tema para su examen del día siguiente:

"Luis, ¿te acuerdas de las Leyes de Mendel?"

Me quedé en blanco. Las había estudiado. Las había memorizado para el examen ICFES. Saqué buena nota. Pero ahí, tres años después, no podía explicarle ni la primera ley.

Busqué en Google. Ahí estaban: Primera Ley (Principio de uniformidad), Segunda Ley (Principio de segregación) y Tercera Ley (Principio de distribución independiente).

Esa misma noche, un estudiante de mi tutoría me escribe:

"Luis, ¿cómo valido que un email tenga formato correcto en Spring Boot?"

Le respondí sin pensarlo:

"Usa @Email de Bean Validation. Lo añades al DTO así: @Email(message = 'Email inválido') private String email; Spring lo valida automáticamente cuando llega la petición. Si falla, devuelve 400 Bad Request con el mensaje. ¿Quieres también validar que no sea null? Combínalo con @NotBlank..."

Cuarenta minutos después, seguíamos hablando. Le expliqué validaciones, manejo de errores, @ControllerAdvice, todo. Sin notas. Sin buscar en Google.

Ahí me di cuenta: estudié biología durante años, pero no recuerdo las Leyes de Mendel. Aprendí validaciones hace seis meses construyendo SLAS (Sistema de Liquidación de Aportes a Seguridad Social), y puedo explicarlas dormido. ¿La diferencia? Mendel lo memoricé para un examen. Validaciones las aprendí resolviendo un problema real. Llevaba 12 años estudiando mal.

El Método del Colegio: Memorizar para Aprobar

A lo largo de mi aprendizaje durante este año, me choqué con muchas formas de aprender que no había conocido hasta que intenté llevar el aprendizaje de programación y el aprendizaje universitario al mismo tiempo. Personalmente, cuando estudié en bachillerato, veíamos entre cinco y seis materias en una jornada de 6:30 a.m. a 1:00 p.m., todos los días de la semana. Muchas clases consistían en memorizar temas de forma superficial. Teníamos aproximadamente 11 materias, las cuales se enfocaban en bloques de dos horas para materias con contexto extendido como cálculo y ciencias políticas.

Especialmente en el último año, nos dieron materias para tener un buen desempeño en el examen Saber 11 o ICFES. Siempre me pregunté por qué no se profundizaba un poco más en las materias que veíamos, y es porque teníamos que cubrir un horario estipulado. Se entiende: el sistema lo exige. Al ver tantas materias, era imposible profundizar en una o centrarse en una sola rama, como, por ejemplo, las ciencias (física, biología, química).

¿Cómo era el método de estudio?

El método que yo usaba (y la mayoría de mis compañeros) era simple: memorizar para aprobar. No importaba si entendías. Importaba si recordabas.

Ejemplo concreto

Lunes, 7:00 a.m. Quiz de física sobre cuerpos rígidos. Problema: la última clase de ese tema había sido el miércoles pasado, cuatro días atrás. ¿Nos dejaron taller para practicar? Sí. ¿Me sirvió? No mucho, porque necesitaba que alguien me explicara de nuevo antes de poder practicar.

Mi estrategia:

• Domingo en la noche: leer apuntes tres veces

• Memorizar fórmulas clave ($\tau = r \times F$, $I = \int r^2 dm$)

• Rezar que no preguntaran algo que no estudiamos

• Pasar con 3.8 o 4.0

• Una semana después: olvidar el 80%

Lo peor no era olvidar. Lo peor era la sensación de "estoy estudiando, ¿pero realmente estoy aprendiendo?"

Intenté profundizar por mi cuenta alguna vez. Busqué videos sobre momento de inercia, leí sobre aplicaciones reales. ¿Resultado? Perdí tres horas en temas que nunca cayeron en el examen. Aprendí la lección: en el colegio, profundizar era "tiempo perdido". Solo estudia lo que va en el examen.

La Transición a la Programación: Un Cambio de Paradigma

Ahora, aprendiendo una carrera tecnológica como la ingeniería en desarrollo de software, se me dificultó pasar de un método de estudio basado en la memorización a entender conceptos aplicados a un contexto tangible.

Ejemplo: Validaciones en Spring Boot

Aprender a usar el *starter* de *validation* para validaciones de entrada de datos a una API está aplicado a un contexto que soluciona un problema real, lo cual genera una sensación de necesidad de conocerlo para evitar errores en nuestros proyectos. Este es un enfoque aproximadamente 70% práctico y 30% teórico.

¿Hasta Dónde Debo Conocer del Tema?

En el colegio, era lo que enseñaran los docentes. Muchas veces intenté profundizar por mi cuenta y tocaba temas que nunca se abordaban en el examen o nunca se sentaban unas bases,

lo cual era tiempo "perdido" para estudiar las demás materias, para poder llevar un buen promedio y no ser juzgado por las demás personas en muchos casos.

En la universidad y aprendiendo de forma autodidacta la programación, es importante conocer el tema y los muchos temas que rodean la tecnología: las ramas que se pueden explorar, las herramientas, etc. Pero se genera la mala sensación de deber conocer todo para poder empezar, y entré en parálisis por análisis.

La Parálisis por Análisis

Cuando empecé a aprender Spring Boot en serio, vi la cantidad de cosas que "debía saber": Spring Core (IoC, DI, Beans), Spring MVC (Controllers, Views), Spring Data JPA (Repositories, Entities), Spring Security (Auth, JWT, OAuth), Spring Boot Actuator, Testing, Profiles...

Mi cerebro de colegio gritó: "¡Debes estudiar todo antes de empezar!" Me bloqueé. Pasé dos semanas leyendo tutoriales, sin escribir una línea de código.

La Salvación: Un Proyecto Real

¿Qué me salvó? Un proyecto real: SLAS. Necesitaba:

1. Calcular aportes de salud (matemática básica)

2. Validar datos de entrada

3. Guardar en base de datos

No necesitaba OAuth. No necesitaba microservicios. No necesitaba Kafka. Aprendí:

• @RestController → porque necesitaba

endpoints

• @Valid + Bean Validation → porque necesitaba validar

• Spring Data JPA → porque necesitaba guardar datos

En el colegio: "Estudia todo primero, aplica después"

En programación: "Aprende lo justo, aplica ahora, profundiza cuando lo necesites"

El Rol de la Investigación y la Profundización

En este caso, no está mal profundizar; al contrario, es súper necesario para poder destacar en la industria que avanza tan rápido como la tecnología. Solo es saber profundizar en el momento adecuado y cuando sea necesario. Entra aquí también el rol investigativo: saber buscar y en dónde buscar, como:

• Documentación oficial

• Blogs especializados

• Tutoriales

• Inteligencia artificial

El cambio más drástico, el que me sigue costando hoy, es aprender hasta dónde llegar en cada versión de un proyecto. Mi cerebro de colegio me dice: "No entregues hasta que sea perfecto". Pero en desarrollo: "Entrega algo funcional hoy, mejóralo mañana".

Reflexión Final

No digo que el sistema educativo colombiano sea inútil. Me enseñó disciplina, constancia y a trabajar bajo presión. Pero me enseñó a estudiar para exámenes, no para crear. Cuando llegué a programación, tuve que desaprender 12 años de hábitos:

• Memorizar todo → Entender contexto

• Saber todo antes → Aprender justo a tiempo

• Profundizar = perder tiempo → Profundizar = diferenciarte

• Copiar soluciones → Resolver problemas reales

Mi Apuesta Personal

No se pueden descuidar ambos mundos. Las universidades son centros de conocimiento que siempre van a ser necesarios. En mi caso, intento aplicar todo lo que veo en la universidad de alguna manera en código: ejercicios de cálculo, álgebra lineal, costos y presupuestos; los intento traducir a código. Así repaso la universidad y le doy un contexto al código que hago.

Mensaje para Otros Estudiantes

Si estás en bachillerato o universidad y sientes lo mismo: no estás loco. El sistema te enseñó a estudiar de una forma. La industria te pide otra. No tienes que elegir entre ser "buen estudiante" o "buen desarrollador", pero sí tienes que ser consciente de que son habilidades diferentes. Y está bien que te cueste el cambio. A mí me sigue costando. Lo importante: ya lo identificaste. Ese es el primer paso.