Fecha: 22 de noviembre de 2021

COMISIÓN DE SELECCIÓN DE ANALISTAS

Convocatoria pública de 26 de enero de 2021, para la provisión de siete plazas vacantes de Analista con destino en la Dirección de Tecnologías de la Información y de las Comunicaciones de la Secretaría General del Congreso de los Diputados

Cuarto Ejercicio

Nombre:	Firma:
Apellidos:	
DNI:	

Instrucciones:

1. No abra este cuestionario hasta que le sea indicado.
2. No escriba ni haga ninguna marca o alteración de los códigos de barras impresos en cada hoja del cuestionario.
3. El enunciado del ejercicio consta de una única página.
4. El tiempo de realización de este ejercicio es de **45 minutos**.
5. Rellene todos los datos de la portada y entréguela cuando se le solicite, antes de comenzar la realización del ejercicio.
6. Puede utilizar todas las hojas en blanco que considere necesario para la realización del ejercicio. Ponga en cada una de ellas el número de orden.
7. Al finalizar el ejercicio, deberá entregar las hojas de respuestas y el enunciado del ejercicio, los cuales serán grapados entre sí y guardados en un sobre cerrado.

Fuente: https://www.infoworld.com/article/3639050/complexity-is-killing-software-developers.html

Complexity is killing software developers

The growing complexity of modern software systems is slowly killing software developers. How can you regain control, without losing out on the best these technologies have to offer?

"Complexity kills," Lotus Notes creator and Microsoft veteran Ray Ozzie famously wrote in a 2005 internal memo. "It sucks the life out of developers; it makes products difficult to plan, build, and test; it introduces security challenges; and it causes user and administrator frustration."

If Ozzie thought things were complicated back then, you can't help but wonder what he would make of the complexity software developers face in the cloud-native era.

The shift from building applications in a monolithic architecture hosted on a server you could go and touch, to breaking them down into multiple microservices, packaged up into containers, orchestrated with Kubernetes, and hosted in a distributed cloud environment, marks a clear jump in the level of complexity of our software. Add to that expectations of feature-rich, consumer-grade experiences, which are secure and resilient by design, and never has more been asked of developers.

"There is a clear increase in complexity when you move to such a pervasive microservices environment," said Amazon CTO Werner Vogels during the AWS Summit in 2019. "Was it easier in the days when everything was in a monolith? Yes, for some parts definitely."

Or, as his colleague, head of devops product marketing at AWS, Emily Freeman, said in 2021, modern software development is "a study in entropy, and it is not getting any more simple."

On the other hand, complex technologies have never been easier to consume off the shelf, often through a single API—from basic libraries and frameworks, to image recognition capabilities or even whole payments stacks. Simply assemble and build your business logic on top. But is it really that simple?

"It has never been more difficult to be a software developer than it is today," said Nigel Simpson, a consultant and former director of enterprise technology strategy at Walt Disney. "While we've seen an up-leveling of capabilities that enable developers to do more by using high-level frameworks for application development and machine learning, this comes at a cost. The explosion of choice and the pace of development make it challenging for developers to keep up with the zeitgeist[1], with many developers getting caught in the headlights."

Essential vs. accidental complexity

Justin Etheredge, cofounder of the software agency Simple Thread, helpfully differentiates between essential and accidental complexity. He told InfoWorld, "Essential is the complexity in the business domain you are working in, the fact that enterprises are extremely complicated environments, so the problems they are trying to solve are inherently complex. The other area is accidental; this is the complexity that comes with our tooling and what we layer on top when solving a problem."

[1] *Zeitgeist*: «espíritu del tiempo» o «espíritu de la época»