Universidad del Quindío

Conceptos Iniciales

Juan Manuel Rojas Rodriguez
Jean Harwol Ariza Rendon
Manuela Puerta Acosta
Programación 3
Jhan Carlos Martinez

Armenia, Quindio
Agosto 2025

Primer seguimiento: Este documento es la evidencia de sustentación, sobre la búsqueda de conceptos iniciales que vamos a desarrollar en el transcurso del curso.

1. Elixir se ejecuta sobre Erlang¿Que es Erlang y que características tiene?

   Erlang es un lenguaje de programación y también un entorno de ejecución diseñado por Ericsson,(es una compañía multinacional sueca dedicada a ofrecer soluciones de telecomunicaciones y transformación digital) en los 80's. Pensado principalmente para sistemas que ocupen alta concurrencia(procesos ejecutándose al mismo tiempo), alta disponibilidad(sistemas que se recuperen rápido) alta tolerancia a fallos.

   Características de Erlang:

Paralelismo masivo con procesos livianos: no usa hilos, sino procesos internos.

programación funcional: basado en funciones puras, reduce errores concurrentes

Permite actualizar código de un sistema en producción sin detenerlo

distribución por naturaleza permite la ejecución de código en varias máquinas, como un solo sistema.

2. ¿Qué ventajas tiene usar elixir en lugar de erlang? Hablé de:

- Distribuido, concurrente, resiliente, velocidad, fácil de usar, actualización de codigo en vivo, árbol de supervisión
- Metaprogramación, DSL (Domain Specific language)
- NIF(Native Implementos functions)

1. Distribuido

Erlang: Nació para sistemas distribuidos (telco, redes) y tiene soporte nativo y sólido para nodos distribuidos.

Elixir: Ofrece lo mismo que Erlang en este aspecto porque hereda todo de la BEAM, pero con sintaxis más amigable y tooling moderno para conectar nodos y hacer despliegues (por ejemplo, con mix releases).

Ventaja Elixir → Misma potencia distribuida de Erlang, pero con una curva de aprendizaje menos intimidante.

2. Concurrente

Erlang: Procesos ligeros, mailbox, y actor model bien establecidos.

Elixir: Igual de concurrente que Erlang, pero con abstracciones como Task, GenServer más expresivos y pipe operator (|>) que hace el código más legible al encadenar transformaciones de datos.

Ventaja Elixir → Mejor ergonomía para manejar concurrencia gracias a su sintaxis y librerías de alto nivel.

3. Resiliente

Erlang: Filosofía "Let it crash" con árboles de supervisión robustos.

Elixir: Exactamente la misma base, pero con documentación más moderna, ejemplos claros y tooling como ExUnit para pruebas, facilitando mantener esa resiliencia en desarrollo.

Ventaja Elixir → Misma solidez, pero más accesible para nuevos equipos.

4. Velocidad

Erlang y Elixir: Idéntica velocidad en ejecución, porque ambos se compilan a bytecode BEAM.

La diferencia está en la velocidad de desarrollo: en Elixir escribir código es más rápido para muchos programadores por su sintaxis parecida a Ruby.

Ventaja Elixir → No es más rápido en ejecución, pero sí en productividad y legibilidad.

5. Fácil de usar

Erlang: Sintaxis más “arcaica” y menos intuitiva para programadores modernos.

Elixir: Sintaxis limpia y expresiva, inspirada en Ruby, más amigable para principiantes y equipos que vienen de lenguajes OO.

Ventaja Elixir → Mucho más cómodo para leer, escribir y mantener código.

6. Actualización de código en vivo

Erlang: Soporta hot code swapping nativo.

Elixir: También lo soporta, pero su ecosistema y herramientas modernas (Phoenix, Mix) lo integran mejor para aplicaciones web y servicios modernos.

Ventaja Elixir → Misma capacidad, pero más fácil de integrar con entornos de desarrollo actuales.

7. Árbol de supervisión

Erlang: Patrón fundamental para la tolerancia a fallos.

Elixir: Mismo patrón, pero con una sintaxis más clara y fácil de leer en supervisores y workers.

Ventaja Elixir → Misma potencia, menos fricción para definir y entender la jerarquía.

8. Meta programación

Erlang: Tiene macros limitadas, menos enfoque en meta programación.

Elixir: Soporta macros muy poderosas y permite generar código en tiempo de compilación, lo que habilita crear APIs y comportamientos muy adaptados al dominio.

Ventaja Elixir → Mucho más fuerte en meta programación

9. DSL (Domain Specific Language)

Erlang: Posible, pero tedioso por la sintaxis.

Elixir: Su sistema de macros permite crear DSLs elegantes y legibles (ej. Ecto para consultas, Phoenix Router, ExUnit).

Ventaja Elixir → Perfecto para crear DSL claros.

10. NIF (Native Implemented Functions)

Erlang: Soporta NIFs para código en C.

Elixir: Igual soporte, pero con librerías y ejemplos modernos que simplifican la integración.

Ventaja Elixir → Misma potencia, mejor documentación y tooling.

3. Entonces. ¿por qué deberías aprender Elixir?

   a. Fácil concurrencias: Al estar construido en Erlang/OPT facilita los sistemas de telecomunicaciones para un optimo manejo de millones de conexiones simultaneas sin caerse, principalmente útil para apps en tiempo real como chats, videojuegos online, streaming, etc..

   b. Futuro y nicho rentable: Si bien, no es el lenguaje mas popular entre programadores las empresas suelen pagar bien debido a que se requiere personal que maneje un alto nivel de concurrencia, tolerancia a fallos y sistemas distributivos

   c. Posee una sintaxis limpia y legible, aunque adopta un paradigma funcional, lo que evita errores y facilita su utilización.

¿Proyectos en los que puede ser ideal elixir?

creación de Apis, microservicios y sistemas distribuidos, procesamiento de datos de streaming, aplicaciones SaaS(software como servicio) Ejemplos: Discord, plataformas de trading, start Ups tecnológicas

Prompt utilizado en IA:

Quiero aprender sobre Erlang y Elixir. Explícame:

Qué es Erlang, incluyendo sus características principales y al menos 5 diferencias importantes frente a otros lenguajes de programación.

Qué es Elixir, en qué se diferencia de Erlang y cuál es mejor en distintos contextos.

Razones para estudiar Elixir en vez de Erlang.

Ejemplos de proyectos para los que es más útil Elixir y para cuáles es más útil Erlang, con ejemplos reales.