# TECNOLÓGICO NACIONAL DE MÉXICO
## INSTITUTO TECNOLÓGICO DE TIJUANA

### SUBDIRECCIÓN ACADÉMICA
### DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

**SEMESTRE:**
Agosto - Diciembre 2024

**CARRERA:**
Ingeniería en Sistemas Computacionales

**MATERIA:**
Programación lógica y funcional

**TÍTULO ACTIVIDAD:**
Preguntas teóricas

**UNIDAD A EVALUAR:**
I

**NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:**

Morales Calvo Ángel Omar - 21212000

**NOMBRE DEL MAESTRO (A):**
René Solis R.

## Cuestionario

### 1. Fundamentos de Erlang:

- **¿Qué características de Erlang lo hacen adecuado para aplicaciones de alta concurrencia y escalabilidad?**

Erlang es un lenguaje concurrente enfocado en la alta disponibilidad, las cuales son aspectos que en un principio habilitan a las aplicaciones de alta concurrencia y su escalabilidad. Esto es debido a que su runtime está diseñado para gestionar una gran cantidad de procesos, ya que los programas escritos en Erlang están construidos de varios procesos Erlang bastante ligeros, llegando a los cientos de estos, cada uno con su encapsulación de datos y la transmisión de mensajes.

Con este modelo de concurrencia, un proceso puede fungir como un supervisor de otros procesos, los cuales pueden actuar como trabajadores, o como supervisores de nivel inferior, asegurando que un programa pueda mantenerse en ejecución, o bien, escalar según se necesite.

- **Explica el modelo de actores en Erlang y cómo se aplica en la gestión de procesos concurrentes.**

El modelo de actores en las ciencias computacionales es un modelo matemático de computación concurrente que trata a un actor como un bloque de construcción fundamental de la computación concurrente.

Un actor es una entidad computacional que, en respuesta a un mensaje que recibe, puede concurrentemente:

- enviar un número finito de mensajes a otros caracteres;
- crear un número finito de nuevos actores;
- designar un comportamiento a emplear para el siguiente mensaje que recibe.

Al leer sobre el modelo de actores, podemos fácilmente relacionarlo con el modelo de concurrencia y el runtime de Erlang, pues un actor podría ser un proceso en la máquina virtual, que ha sido lanzado por cierta función. Cada proceso se comunica

por el envío de mensajes a otros procesos. Luego, cada proceso puede supervisar otros procesos en el árbol de supervisión bajo el que actúa, gestionar su ciclo de vida en caso de un crasheo, comunicar datos a otros procesos, etcétera, justo como un actor.

Aunque el cómo trabaja Erlang es bastante similar al modelo de actores, los diseñadores de Erlang han comentado que aprendieron sobre dicho modelo después de haber diseñado el lenguaje, por lo que Erlang no implementa el modelo de actores conscientemente, no obstante, esto no quiere decir que Erlang no lo implementa conceptualmente, aunque hay ciertas diferencias esperadas.

**2. Características de Cowboy:**

- **¿Cuáles son las principales ventajas de usar Cowboy como servidor HTTP en aplicaciones Erlang?**

Cowboy provee un framework ligero con el que se puede generar un stack HTTP completo en una base de código relativamente pequeña. De acuerdo con el creador, está optimizado para baja latencia y bajo uso de memoria. Además, su arquitectura es orientada a eventos y asíncrona, permitiéndole realizar una gran cantidad de operaciones de forma concurrente.

- **Describe cómo Cowboy maneja las conexiones concurrentes de manera eficiente.**

Debido a que utiliza operaciones I/O sin bloqueo y el modelo de concurrencia de pase de mensajes de Erlang, Cowboy puede manejar múltiples peticiones concurrentemente sin bloquear el proceso servidor principal.

**3. Uso en la Industria:**

- **Menciona tres empresas que utilizan Erlang/Cowboy en su infraestructura y explica brevemente cómo lo emplean.**
    1. WhatsApp: de acuerdo a la información pública, alrededor de 2014, Erlang era utilizado en el lado del servidor para desarrollar nuevas características, mantener las existentes, y soportar el sistema completo, lo que implica el envío de mensajes de usuarios. Esto debido a que Erlang mejora la escalabilidad relativa a su modesta huella en el hardware.
    2. RabbitMQ: Utiliza Erlang como su lenguaje principal con el que se implementa su protocolo de cola de mensajes, debido a sus capacidades nativas de manejar una gran cantidad de operaciones concurrentes.
    3. bet365: Utiliza Erlang para manejar cientos de miles de usuarios concurrentes que se encuentran realizando apuestas.
- **¿En qué tipos de aplicaciones es menos común utilizar Cowboy y por qué?**

De acuerdo a las capacidades de Cowboy, no parece que realmente haya un área en el que no pueda dar una solución factible, el problema recae en el propio lenguaje de programación que se implementa. Recientemente, Heroku anunció que migraría su router basado en Erlang a Go, principalmente por múltiples dependencias que habían quedado desactualizadas o sin soporte por los autores, lo que evitaba que se actualizará la aplicación.

Por lo tanto, el principal problema es que Erlang comienza a ser un lenguaje para ciertos nichos, mas no quiere decir que Cowboy sea inutil no tenga área de aplicación, pues ha demostrado proveer las suficientes capacidades para realizar algo funcional.

Aunque algunas empresas que inicialmente utilizaban Erlang, han migrado a otros lenguajes, el caso más interesante es Ericsson, pues es el lugar en donde surgió Erlang, y en el que alguna vez fue el lenguaje principal, hoy hay más equipos que principalmente desarrollan en C.

**4. Integración con Otros Frameworks:**

- **¿Cómo se integra Cowboy con el framework Phoenix en Elixir?**

Cowboy es el servidor HTTP predeterminado que utiliza Phoenix en Elixir. Phoenix utiliza Cowboy para manejar solicitudes HTTP de forma concurrente y eficiente. Al ser ambos proyectos basados en el ecosistema de Erlang/OTP, se integran de manera fluida. Phoenix proporciona una capa de abstracción que facilita el trabajo con Cowboy, permitiendo a los desarrolladores centrarse en escribir código de aplicaciones web en lugar de preocuparse por la gestión de conexiones HTTP de bajo nivel.

Durante el arranque de una aplicación Phoenix, Cowboy se inicia como parte del sistema de supervisión de OTP, escuchando las conexiones entrantes en los puertos configurados. Phoenix configura Cowboy para manejar diferentes tipos de solicitudes, como las peticiones HTTP/HTTPS, y también se encarga de administrar rutas, middlewares y ciclos de vidas.

- **Explica la relación entre Phoenix Channels y Cowboy en el manejo de WebSockets.**

Phoenix Channels permite el manejo de comunicaciones bidireccionales y en tiempo real, utilizando WebSockets para la transmisión de mensajes entre el servidor y los clientes conectados. Cowboy es el servidor que subyace en la implementación de Phoenix Channels y maneja la apertura y gestión de las conexiones WebSocket de manera eficiente, gracias a su arquitectura orientada a eventos y su capacidad de manejar múltiples conexiones concurrentes sin bloqueo.

**5. Desafíos y Consideraciones:**

- **¿Cuáles son los principales desafíos al aprender y utilizar Cowboy para nuevos desarrolladores?**

Aunque Cowboy es un servidor HTTP eficiente, el lenguaje Erlang ofrece suficiente reto para ser desafiante familiarizarse con el modelo de concurrencia basado en múltiples procesos ligeros, que a su vez el paradigma y el patrón de diseño resulta bastante distintivo de otros lenguajes conocidos.

Además, otro desafío es la comunidad relativamente pequeña que impacta en la disponibilidad de recursos actualizados, y la creación de nuevos paquetes y dependencias que podrían resolver cierto problema en el framework, a comparación de otros más populares como Django.

- **Discute cómo la tolerancia a fallos de Erlang beneficia a las aplicaciones desarrolladas con Cowboy.**

Erlang fue diseñado con la tolerancia a fallos en mente, lo cual beneficia directamente a las aplicaciones construidas con Cowboy. El sistema de supervisión jerárquica de Erlang permite que si un proceso falla, sea reiniciado automáticamente sin afectar al resto de la aplicación. Esto significa que las aplicaciones web que usan Cowboy pueden manejar fallos inesperados sin sufrir tiempos de inactividad prolongados.