



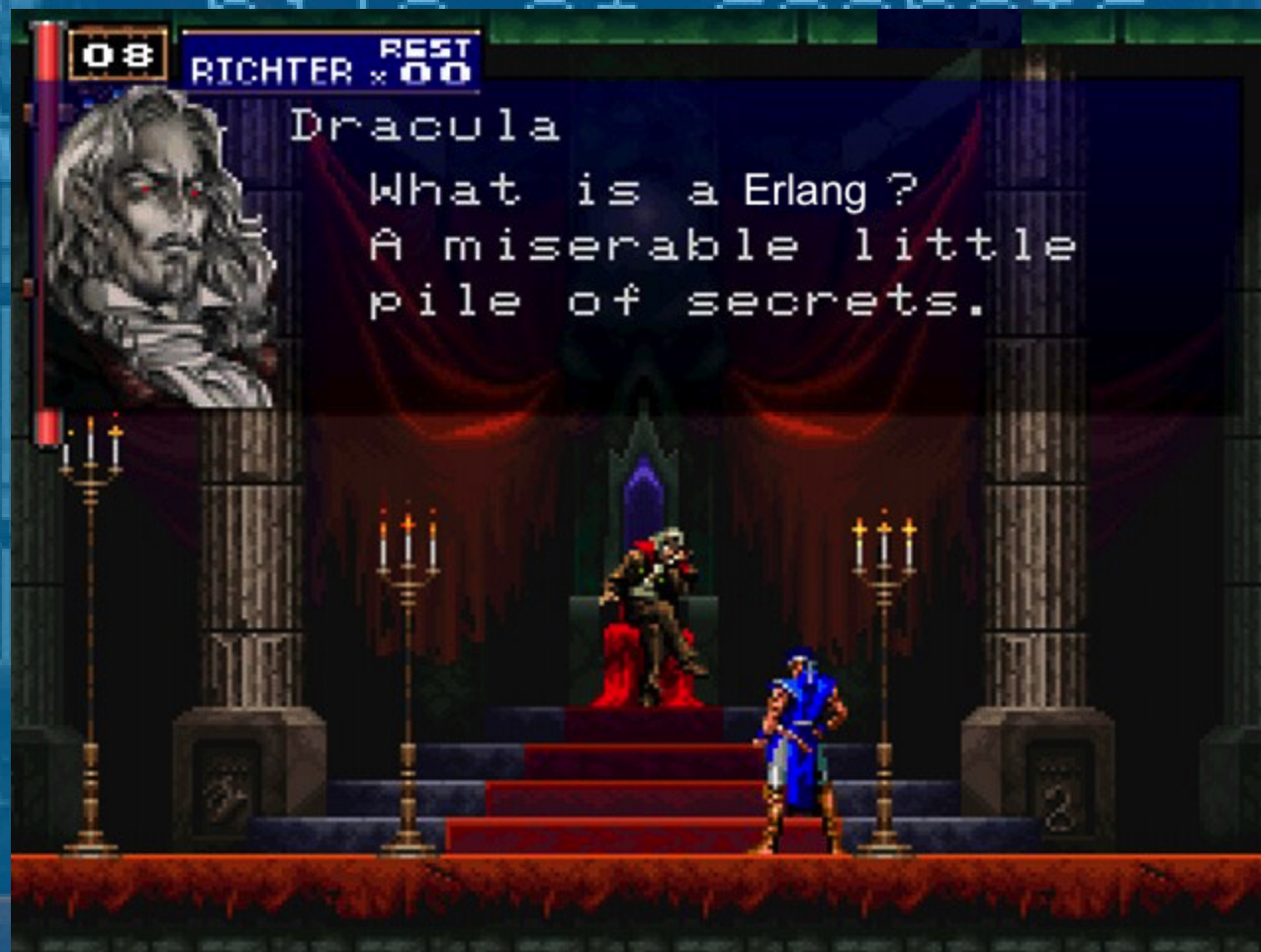
Enter the Erlang 🦖🦎 with LFE

Todo lo que necesitas para ser un crack 🐅




08

REST
RICHTER x 00

¿Qué es Erlang?



¿Por qué Erlang?

- Es una tecnología probada con + **30** años en la ~~trinchera~~ industria  →  → 
- Manejo de concurrencia de forma sana !
- Soft-real time server side
- Capacidad de Tolerancia a fallos
- OTP: Patrones de diseño reales (~~Gang of Four~~)
- La VM es más un OS que un interprete del bytecode

No tenés que saber **OTP** !

Si sabés un poco de Lisp, podés aprender el ecosistema sobre la marcha 🧠📚

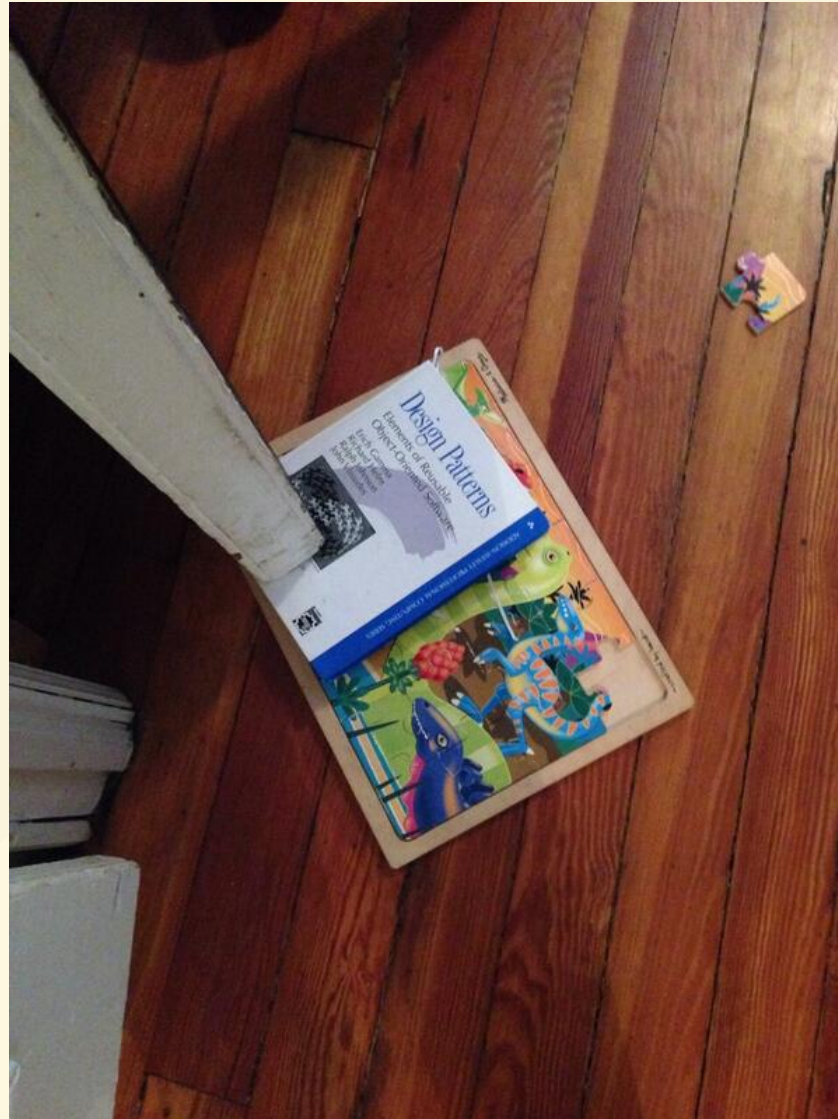


It's a UNIX system.
I know this.

Erlang es piola 100 ! para soft real time 🕒



- Real 🕒 -> Perder un deadline es una falla total del sistema
- Soft 🍦 Real 🕒 -> La utilidad de un resultado se degrada después del deadline 📉, pero sigue siendo útil. En sistemas de streaming se valora fluidez del servicio.

OTP: Patrones de diseño posta



El Actor Model

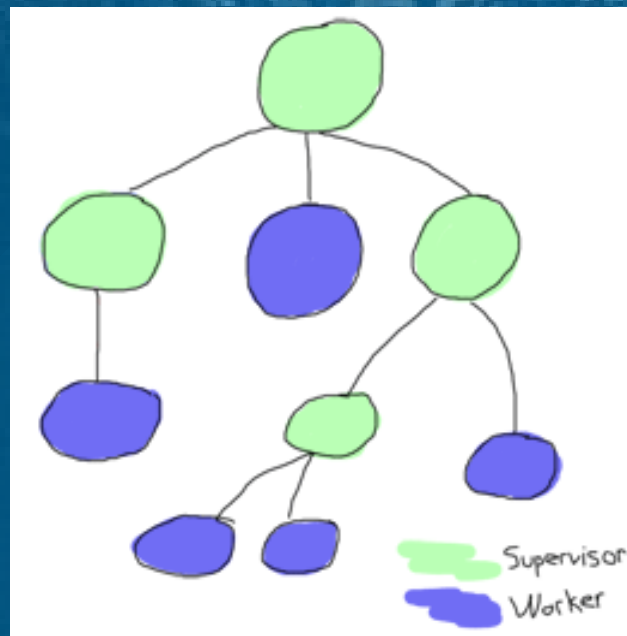
En Erlang los actores son procesos livianos aislados de la beam (NO son hilos del SO), no comparten memoria o no deberían (?)

- Se comunican por msg  (mutabilidad)
- Tiene su propio **mail box** 
- No locks/mutexs para la concurrencia 🙌 100 !
- Cada actor tiene su propio garbage collector
- Cualquier parecido con la POO es pura coincidencia(?)

El Supervisor

El supervisor es un **behaviour** responsable de arrancar, parar y monitorear sus procesos hijos.

Siempre es un nodo del arbol de procesos, en cambio los procesos worker son hoja o terminales.



El Gen Server aka Microservicios

Es el patrón para escribir servidores genericos en erlang.

La idea es separa la funcionalidad del manejo de la concurrencia del servidor a travez de callbacks.

handle_call: llamadas síncronas (esperan respuesta)

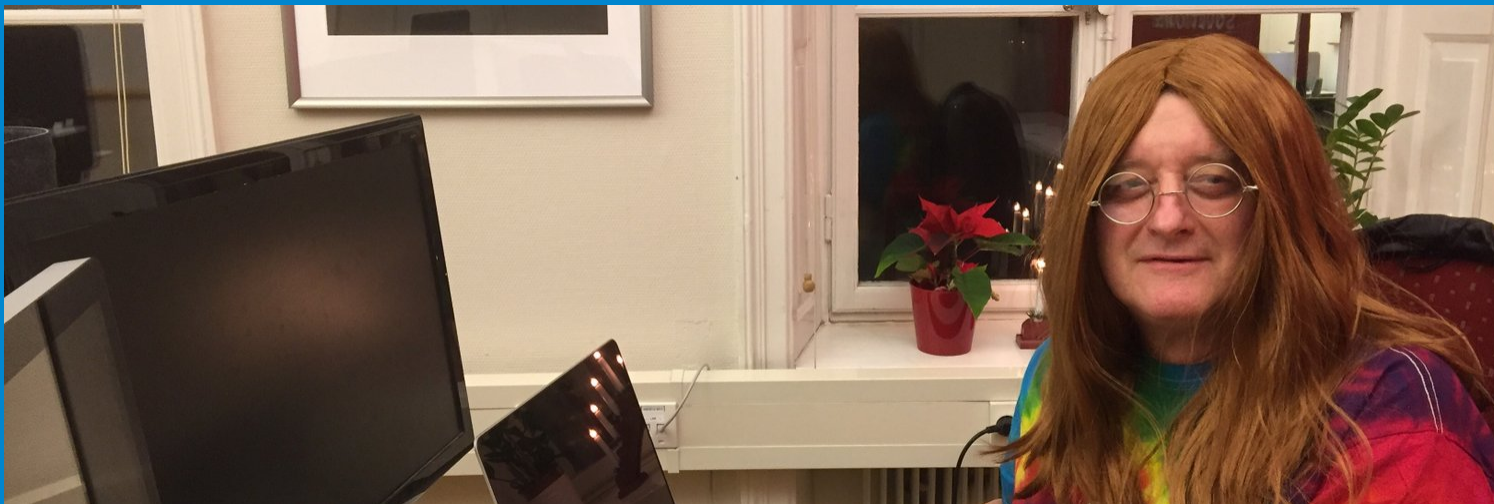
handle_cast: llamadas asíncronas (Sin esperar respuesta)

Lo qur uno programe dentro de cierta llamada va a ser servido por un proceso de la beam.

`(lisp (flavoured (erlang)))`

LFE es un dialecto de LISP creado por [Roberto Virding](#) sobre la Erlang VM.

Es un Lisp2+, LFE tiene diferentes namespaces. Podes tener una fun `help` y una var `help`



Números

Los enteros pueden ser tan grandes como quieras o te quedás sin memoria, lo que suceda primero 😊💧

Hay de punto flotante, pero a nadie le importa (?)
Tampoco hay **'nan** ni **'infinity**, los tenes que fabricar.

```
lfe> ( / 1000000444 991)
1009082.1836528758
lfe> (/ 1.0 0.0)
exception error: error in arithmetic expression
    in (erlang : / 1.0 0.0)
```

Cadenas

las cadenas en Erlang son **listas** 🍏 🎩 ... de enteros 😊

```
lfe> (== "Ceci n'est pas une " (99 104 97 238 110 101))  
"Ceci n'est pas une chaîne"
```

```
lfe> (lfe_io:format "Maximale ascii est: ~c. "  
    (list (lists:max "Ceci n'est pas une chaîne")))  
Maximale ascii est: î. ok
```


```
erlang> io:format("Maximale ascii est: ~c. ",  
    [lists:max("Ceci n'est pas une chaîne")]).  
Maximale ascii est: î. ok
```

Átomos

son **enums** que se representan así mismos, los átomos empiezan con comilla simple '.

```
lfe> (erlang:is_atom 'desinflamante')  
true
```

```
erlang> erlang:is_atom(desinflamante).  
true
```

- **'true** tiene un valor **truthy** y el **'false** **falsey**
- No hay **null** , pero podés definir el átomo **'null**, **'undefined**, **'none**, **'nothing**, **'lol**, **'ahre**

Binarios -> blob

Ejemplo de Exercism

```
(defmodule leap
  (export all))

(defun leap-year
  ((year) (when (== 0 (rem year 400)))
    'true)
  ((year) (when (== 0 (rem year 100)))
    'false)
  ((year) (when (== 0 (rem year 4)))
    'true)
  ((_year)
    'false))
```

Ejemplo de Exercism

```
(defmodule rna-transcription
  (export (to-rna 1)))

(defun to-rna-char
  ([#\G] #\C)
  ([#\C] #\G)
  ([#\T] #\A)
  ([#\A] #\U))

(defun to-rna (dnaList)
  (lists:map #'to-rna-char/1 dnaList))
```

Erlang viene con 3 DBs build-in

Las mas usadas son ETS y Mnesia

Módulos



Módulos

**En Erlang los módulos contienen
las funciones y estas tienen**

Dieselpunk



