Aylen Reynoso

Se toma???

Un **elixir** (del latín «elixir», del árabe clásico *الإكسير* «al'iksīr» y del idioma griego *ξηρά*) es un líquido de sabor dulce utilizado con fines medicinales para curar enfermedades.

Agenda

- Historia de Elixir
- Runtime
- Manejo básico del lenguaje
- Aspectos avanzados
  - Procesos y Actores
- Trabajo Integrador
- Phoenix LiveView

Historia de Elixir

Caracteristicas

- Funcional
- BEAM based
  - Escalabilidad
  - Tolerancia a fallos
- Tools
  - OTP
  - Mix
  - Iex
  - ExUnit
- Compatible con Erlang

Origenes

Creador: Jose Valim

Año: 2012

- Problematica
  - Concurrencia
  - Thread Safety en OOP
  - Escalabilidad de app en Ruby on Rail
- Descubrimientos:
  - Estado explícito en lugar de implicito, ~~mutar~~ transformar
  - Erlang, principal use case construccion de Telecom Switches

La programación funcional está asociada a la concurrencia pero esto no es por diseño. Sucede que, haciendo explícitas las partes más complejas de un sistema, resolver problemas complicados como la concurrencia se vuelve mucho más sencillo.

–José Valim

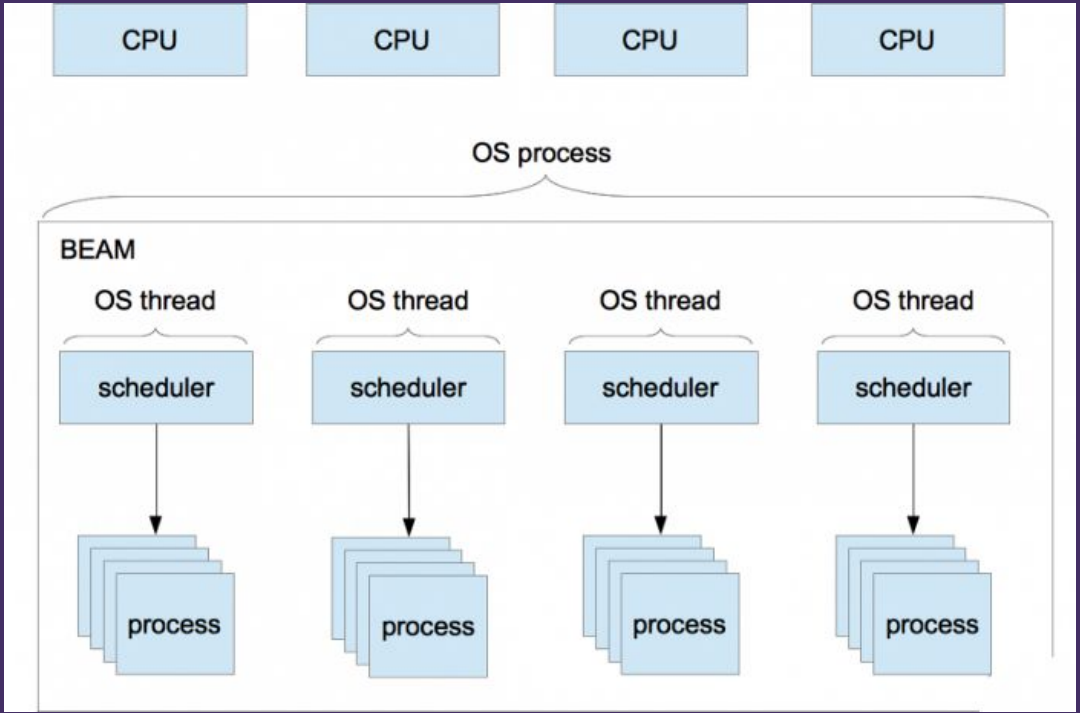
Inicialmente no estaba realmente interesado en la concurrencia como tal, estaba interesado en cómo construir sistemas fault-tolerant. Una característica de estos sistemas (switches) era que manejaban cientos de miles de llamadas telefónicas al mismo tiempo.

-Joe Armstrong

Runtime

Erlang/Elixir Runtime

El runtime de Elixir es una instancia de BEAM!!!



El mundo es paralelo. Si queremos escribir programas que se comporten como se comportan otros objetos en la realidad entonces estos programas tendrán una estructura concurrente... Las personas funcionan como entidades independientes que se comunican enviando mensajes. Así es como funcionan los procesos en Erlang, los programas están compuestos por pequeños procesos que hablan el uno con el otro - como la gente

-Joe Armstrong

Estructura de un programa en Elixir

- Procesos livianos y aislados
- Comunicación a través de mensajes
- Arboles de supervision
- Fail fast approach
- Organización modular

Herramientas

Por qué Elixir?

OTP: Open Telecom Platform Set de librerías que vienen con Erlang usadas para construir apps robustas y tolerantes a fallos.

Mix: Brinda tasks para crear compilar testear aplicaciones, maneja dependencias, etc.

ExUnit: framework para unit test.

Phoenix: framework para desarrollo web.

Y mucho mas...

Básico

Los conocidos

```
iex> 1          # integer
iex> 0x1F       # integer
iex> 1.0        # float
iex> true       # boolean
iex> "elixir"   # string
iex> [1, 2, 3]  # list
iex> {1, 2, 3}  # tuple
```

Strings

```
iex> "Hola Mundo"
"Hola Mundo"
iex> 'Hola Mundo'
'Hola Mundo'
iex> 'hello' == "hello"
false
```

```
iex> IO.puts "hello\nworld"
hello
world
:ok
iex> byte_size("holä")
5
iex> String.length("hola")
4
```

Listas

```
iex> list = [1, 2, 3]
iex> hd(list)
1
iex> tl(list)
[2, 3]
iex> length list
3
```

```
iex> list ++ [4, 5, 6]
[1, 2, 3, 4, 5, 6]

iex> [1, true, 2, false, 3, true] -- [true, false]
[1, 2, 3, true]

iex> [0 | list]
[0, 1, 2, 3]
```

Tuplas

```
iex> tuple = {:ok, "hello"}
{:ok, "hello"}

iex> elem(tuple, 1)
"Hello"

iex> tuple_size(tuple)
2
```

```
iex> tuple = {:ok, "hello"}
{:ok, "hello"}

iex> put_elem(tuple, 1, "world")
{:ok, "world"}

iex> tuple
{:ok, "hello"}
```

Funciones anónimas

```
iex> add = fn a, b -> a + b
end
iex> add.(1, 2)
3
iex> is_function(add)
true
```

```
iex> x = 42
42
iex> (fn -> x = 0 end).()
0
iex> x
42
```

Atomos y Alias

```
iex> :hola
:hola
iex> :mundo
:mundo
iex> :hola == :hola
true
iex> :hola == :mundo
false
```

```
iex> true == :true
true
iex> is_atom(false)
true
iex> is_boolean(:false)
true
iex> is_atom(Hello)
true
```

Módulos y Funciones

```
iex> defmodule Math do
...>   def sum(a, b) do
...>     a + b
...>   end
...> end
iex> Math.sum(1, 2)
3
```

```
iex> fun = &Math.sum?/1
&Math.sum?/2
iex> is_function(func)
true
iex(6)> fun.(1,2)
3
```

Keyword List

```
iex> list = [{:a, 1}, {:b, 2}]
[a: 1, b: 2]
iex> list == [a: 1, b: 2]
true
```

```
iex> list ++ [c: 3]
[a: 1, b: 2, c: 3]
iex> [a: 0] ++ list
[a: 0, a: 1, b: 2]
iex> new_list = [a: 0] ++ list
[a: 0, a: 1, b: 2]
iex> new_list[:a]
0
```

Mapas

```
iex> map = %{:a => 1, 2 => :b}
%{2 => :b, :a => 1}
iex> map[:a]
1
iex> map[2]
:b
iex> map[:c]
nil
```

```
iex> %{} = %{:a => 1, 2 => :b}
%{2 => :b, :a => 1}
iex> %{:a => a} = %{:a => 1, 2
=> :b}
%{2 => :b, :a => 1}
iex> a
1
```

Estructuras de control

Case

```
iex> case {1, 2, 3} do
...>   {4, 5, 6} ->
...>     "This clause won't match"
...>   {1, x, 3} ->
...>     "This clause will match and bind x to 2 in this clause"
...>   _ ->
...>     "This clause would match any value"
...> end
```

Estructuras de control

Cond

```
iex> cond do
...>   2 + 2 == 5 ->
...>     "This will not be
true"
...>   2 * 2 == 3 ->
...>     "Nor this"
...>   1 + 1 == 2 ->
...>     "But this will"
...> end
 "But this will"
```

If y Unless

```
iex> if true do
...>   "This works!"
...> end
"This works!"

iex> unless true do
...>   "This will never be
seen"
...> end
nil
```

Aspectos avanzados

Pattern Matching

Asignación de múltiples variables

```
iex> {a, b, c} = {:hello, "world", 42}
{:hello, "world", 42}
iex> a
:hello

iex> [first, second | rest] = [1,2,3,4]
iex> rest
[3,4]
iex> second
2
```

Match

```
iex> {a, b, c} = {:hello, "world"}
** (MatchError) no match of right hand side value: {:hello, "world"}

iex> 5 = 2 + 2
```

Rebound de variables

```
iex> x = 1
1

iex> 1 = x
1

iex> 2 = x
** (MatchError) no match of
right hand side value: 1
```

Operador pin

```
iex> x = 2
2

iex> ^x = 1
**(Match Error)

iex> x
2
```

Procesos

Los Procesos en Elixir

```
iex> spawn fn -> 1 + 2 end
#PID<0.43.0>
iex> send self(), {:hello, "world"}
{:hello, "world"}
iex> receive do
...>   {:hello, msg} -> msg
...>   {:world, _msg} -> "won't match"
...> end
"world"
```

Alias/Nombre

```
iex> Process.register(pid, :kv)
true
iex> send :kv, {:get, :hello, self()}
{:get, :hello, #PID<0.41.0>}
iex> flush()
:world
:ok
```

Actores

Abstracciones

Abstracciones para procesos:

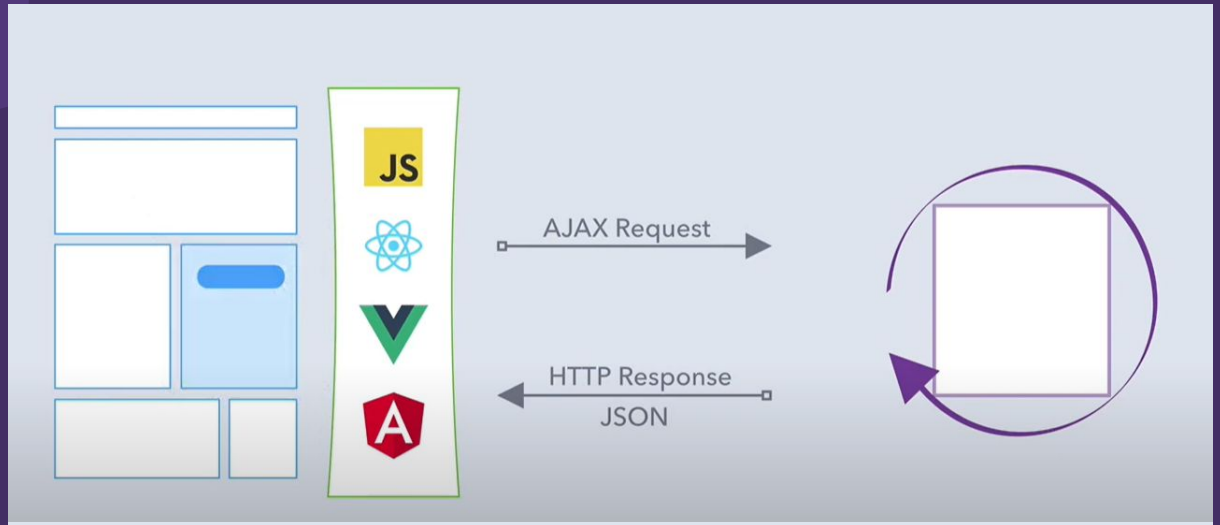
- Agent: abstracción de estado
- GenServer: relación client-server
- Task: unidades de computo

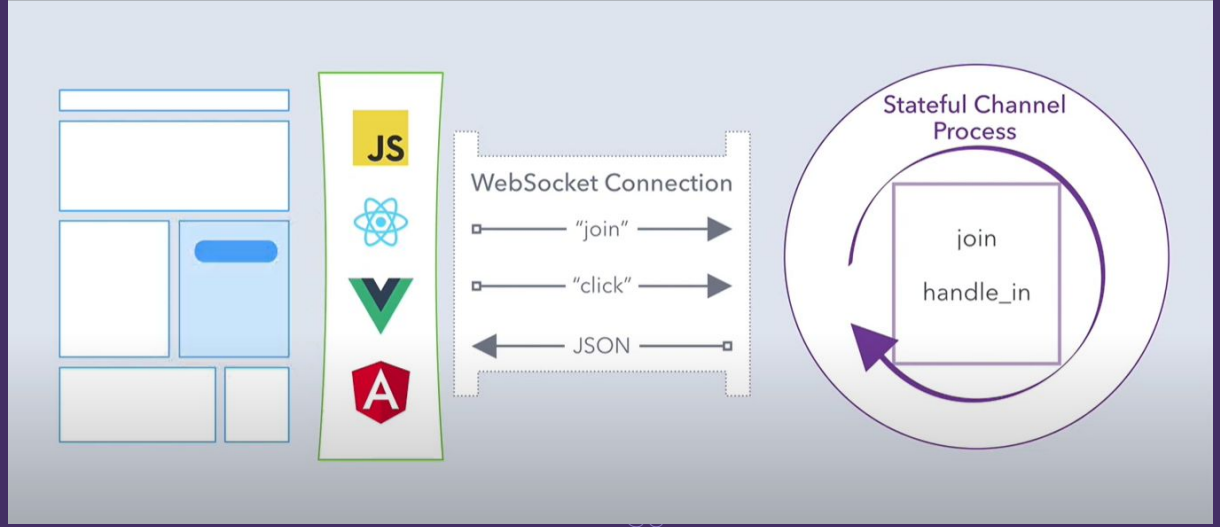
Componentes:

- Supervisor
- Application

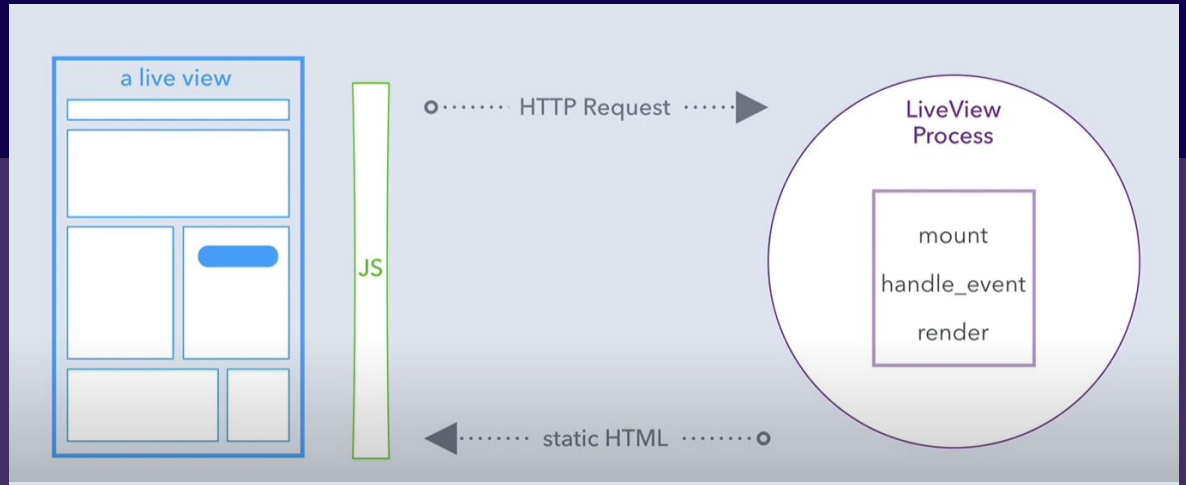
$ALLY

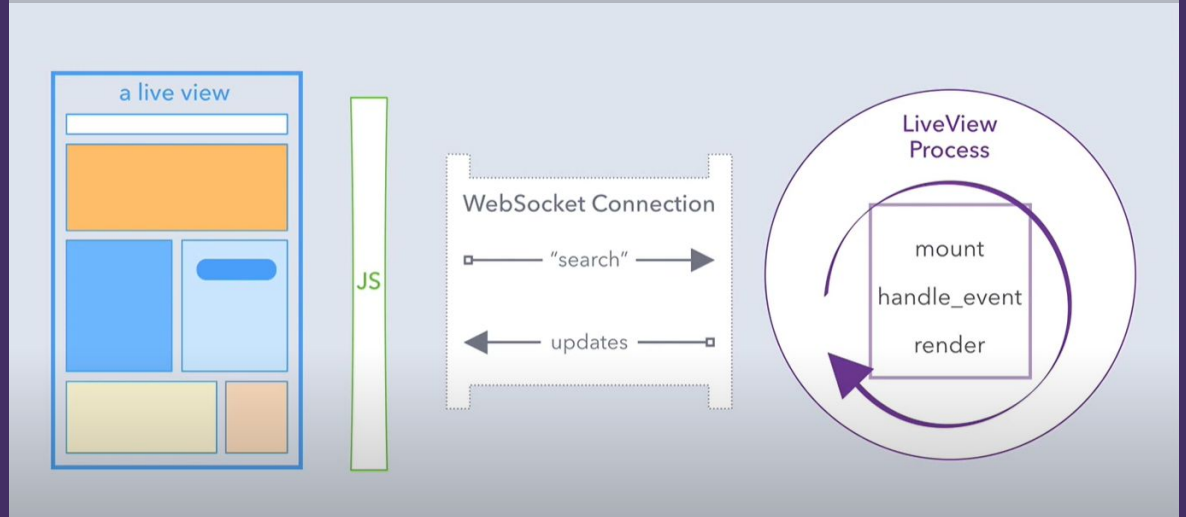
Phoenix LiveView


JS
AJAX Request
HTTP Response
JSON


JS
WebSocket Connection
"join"
"click"
JSON
Stateful Channel
Process
join
handle_in

Cómo funciona LiveView?





Estadisticas

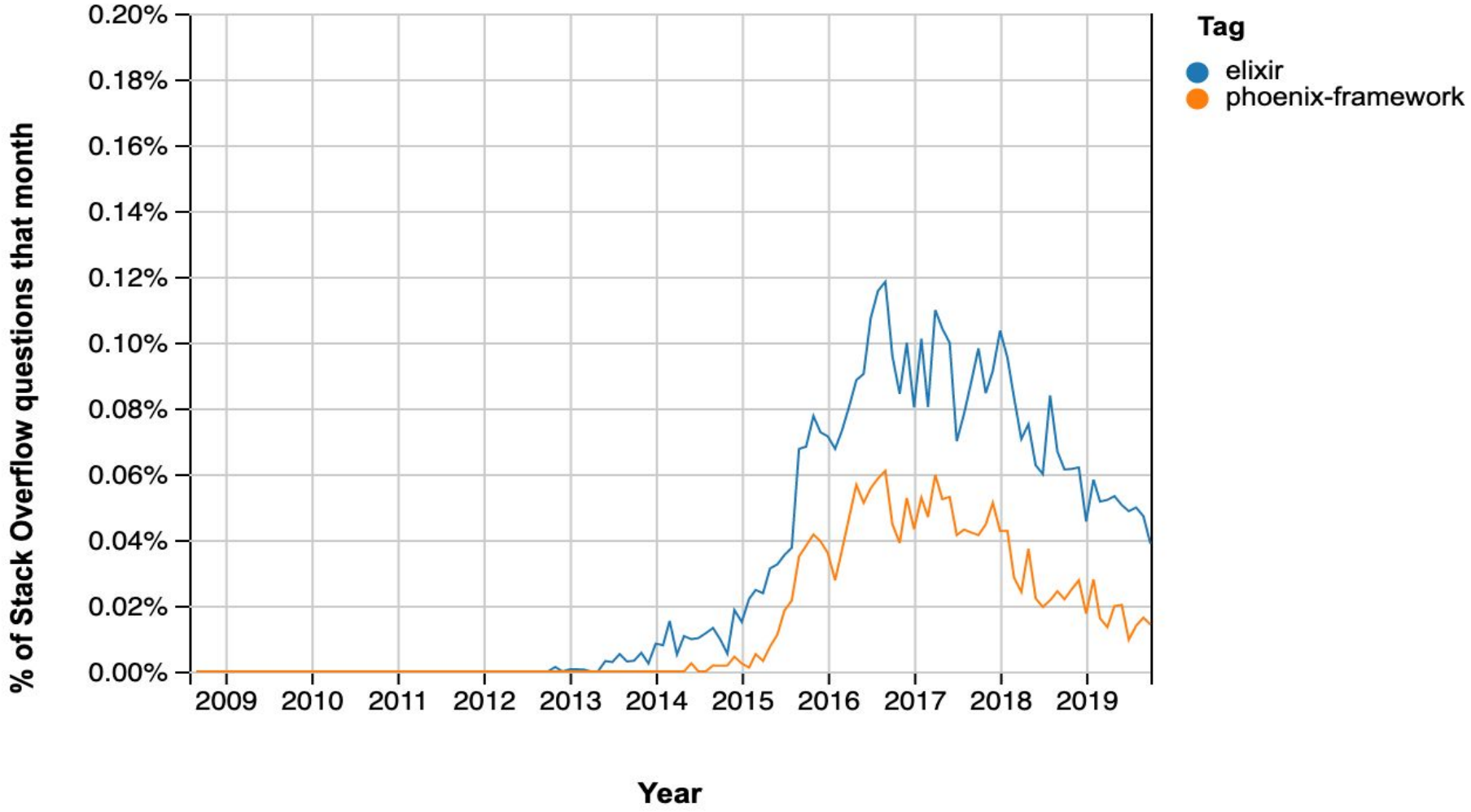
0.465%

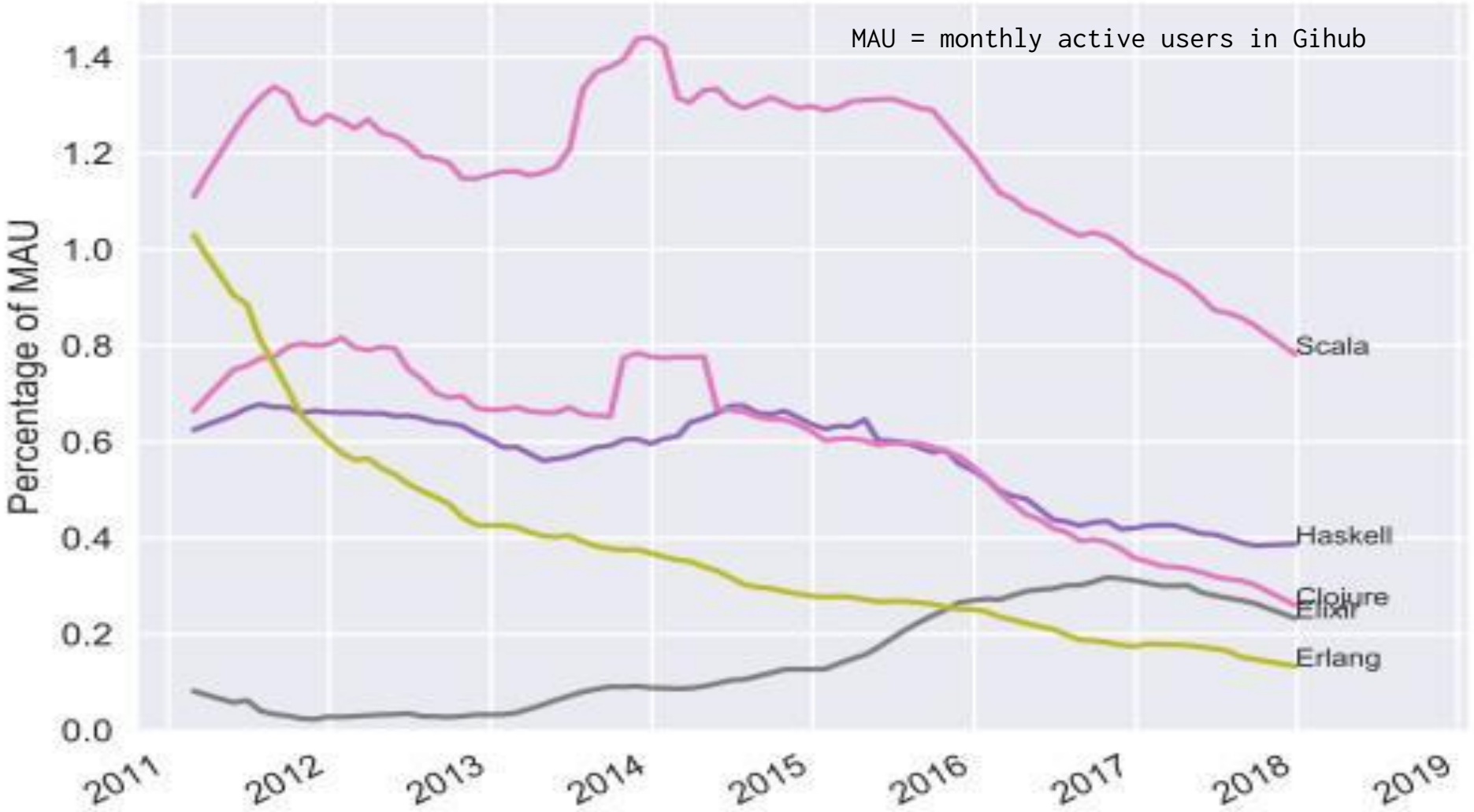
Porcentaje de pull request en github segun

https://madnight.github.io/githut/#/pull_requests/2019/3

7mo

Mas amado como lenguaje segun Stackoverflow - 2017


0.20%
0.18%
0.16%
0.14%
0.12%
0.10%
0.08%
0.06%
0.04%
0.02%
0.00%
% of Stack Overflow questions that month
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Year
Tag
elixir
phoenix-framework


MAU = monthly active users in Gihub
Percentage of MAU
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2011
2012
2013
2014
2015
2016
2017
2018
2019
Scala
Haskell
Clojure
Elixir
Erlang

Compañias que usan Elixir

Pinterest

They use it to handle the routing of the events generated in their system.

Financial Times

SquareEnix

SquareEnix is using Elixir for authenticating players, in-game communication and the CMS (Content Management System).

Moz Pro

They decided to replace their traditional MySQL databases with a distributed indexing data system built in Elixir..

Toyota

Toyota Connected launched its Mobility Service Platform (MSPF) which connects their cars, allowing them to send real-time events.

Discord

Discord, a leading provider of chat for gaming have built their systems in Elixir.

Gracias!

Preguntas?