Meetup =

Elixir

|> nimbbo

|> UNITEC

Agenda

- Instalación Elixir
- Que es elixir?
- Características Principales del Lenguaje
- Demos

nimbbo

unitec
LAUREATE INTERNATIONAL UNIVERSITIES

Creador Elixir

plataformatec

- José Valim
- Fundador y Director Plataformatec
- Creador de elixir
- Miembro de la comunidad Ruby
- Facilitar el manejo de concurrencia en Ruby

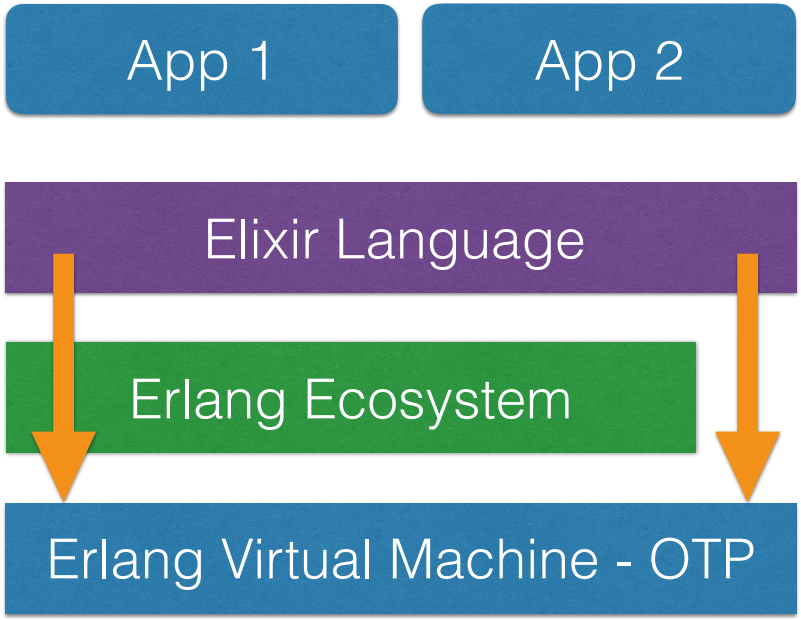
https://www.sitepoint.com/an-interview-with-elixir-creator-jose-valim/

Que es elixir?

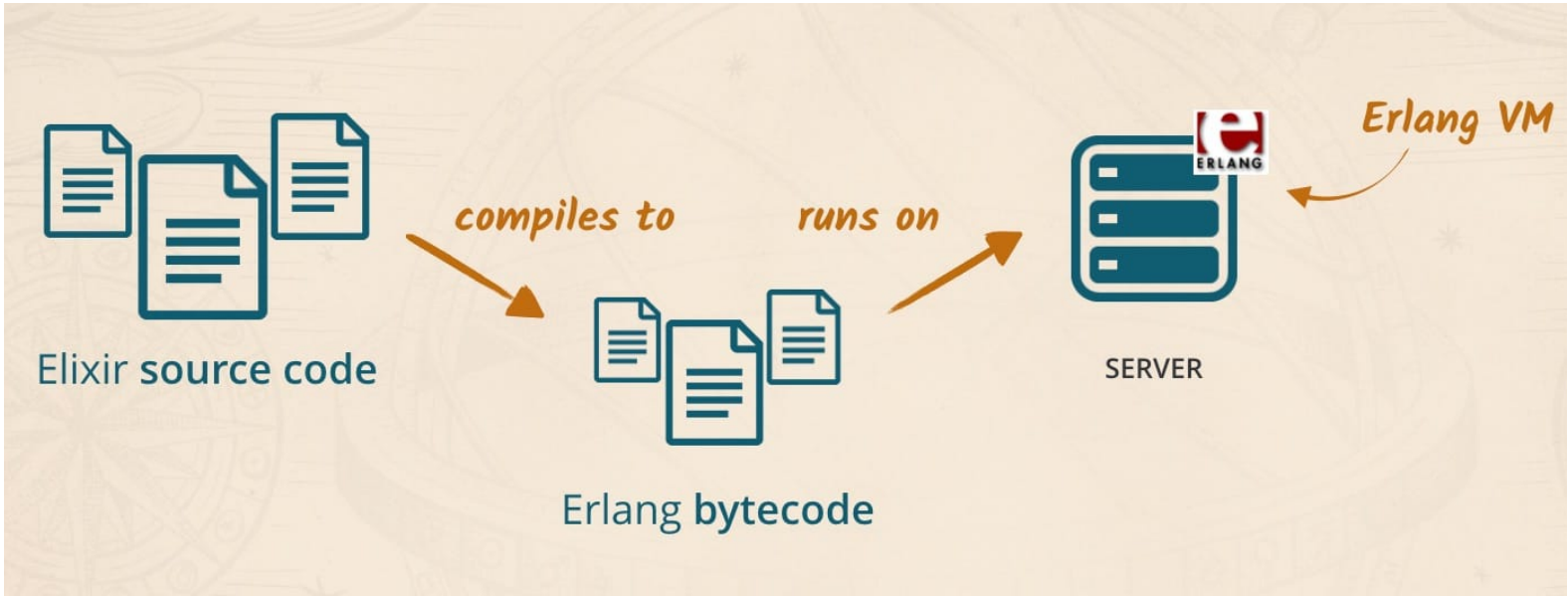
Elixir es un **lenguaje de programación funcional** diseñado para construir aplicaciones **escalables y administrables**.

Elixir usa de base **Erlang VM**, muy conocido por correr **sistemas distribuidos, de alta disponibilidad y baja latencia**. Adicionalmente ha sido utilizado satisfactoriamente en el desarrollo de **aplicaciones web y entornos de embedded software**.

elixir - Erlang



- Hace uso de todo el ecosistema de Erlang sin penalización de performance
- Compila directo a bytecode utilizado por el Erlang VM



Compiling Elixir to Erlang VM

https://www.codeschool.com/blog/2017/02/22/why-elixir/

Instalación

- https://elixir-lang.org/install.html
- Opciones de instalación
  - MacOS X
  - Linux
  - Windows
  - Dockers
  - Nano
  - Raspberry Pi

José Valim dice que es:

- Un **lenguaje de programación funcional** enfocado en la productividad del programador.
- La concurrencia es el backbone de una aplicación desarrollada en Elixir.
- Nos permite pensar en transformar data utilizando funciones
- Es un lenguaje de programación extensible. Macros, protocolos.

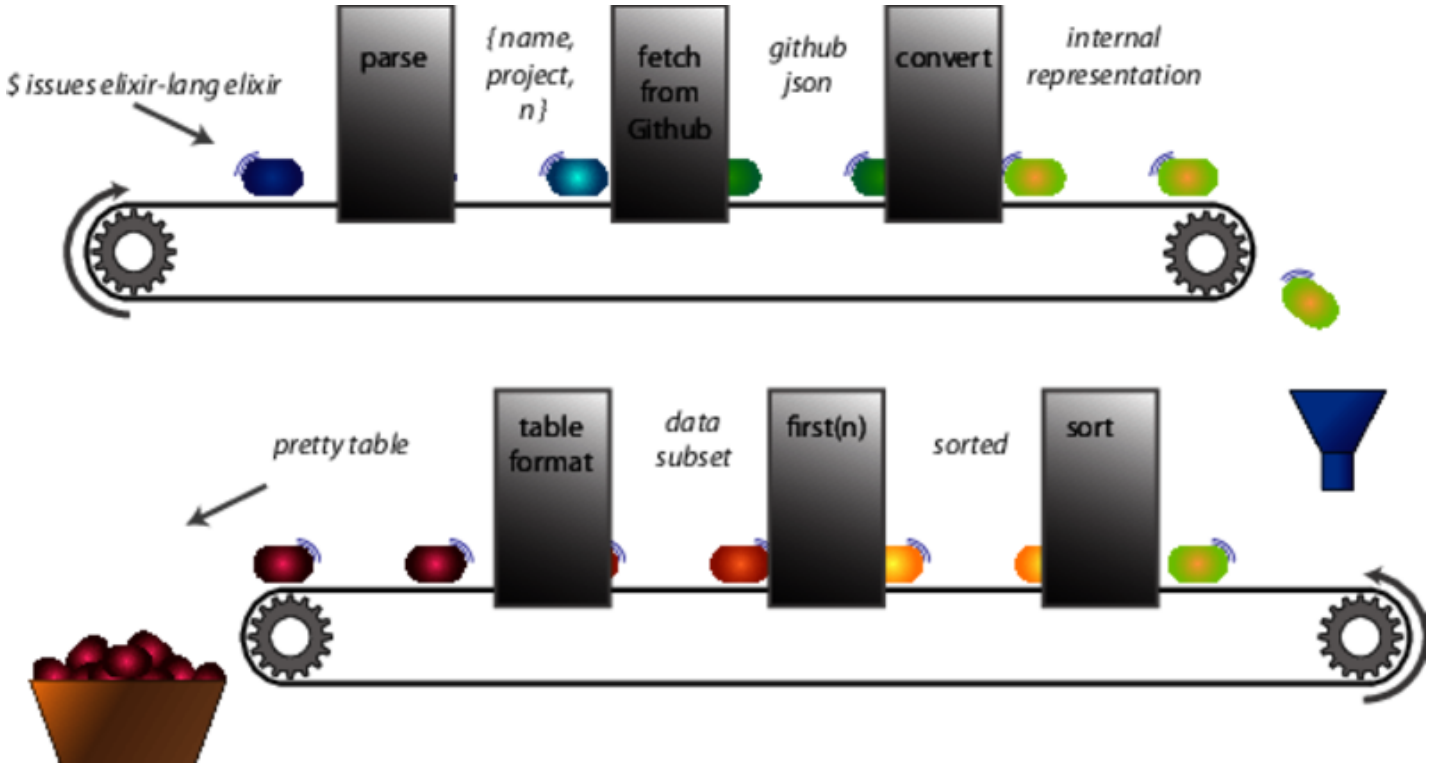
Palabras clave

- Lenguaje de programación funcional
- Transformación de data
- Concurrente - utilizando procesos
- Escalable - Alta Disponibilidad - Sistema Distribuido
- Mejora productividad del desarrollador - Sintaxis, extensible

Lenguaje programación funcional

- Elemento principal es la función
- No existen variables, solo constantes con valores inmutables
- Promueve el uso de funciones puras. No importa el orden de ejecución. Facilita ejecutar código de manera concurrente.
- No existen loops - solo recursion
- Funciones como parámetros
- Pattern Matching

http://theerlangelist.blogspot.com/2013/05/working-with-immutable-data.html



Transformación de la Data

Ejemplo: Libro Programming Elixir 1.2 by Dave Thomas

Demo

Transformando una lista CSV
Ejercicio Transformación

Setup

Creación aplicación base para DEMO

Procesos

Que es un proceso?

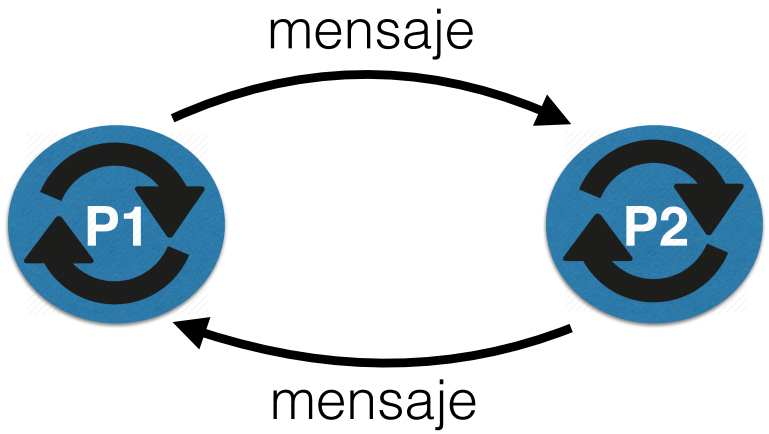
- Es la unidad básica para la concurrencia.
- No son procesos de sistema operativo
- Erlang VM sporta 134MM
- Es el actor en el “Actor Concurrency Model”

Cuando utilizar procesos

- Manejar estado
- Para ejecutar trabajos de manera concurrente
- Para manejar y aislar fallas
- Para comunicación distribuida

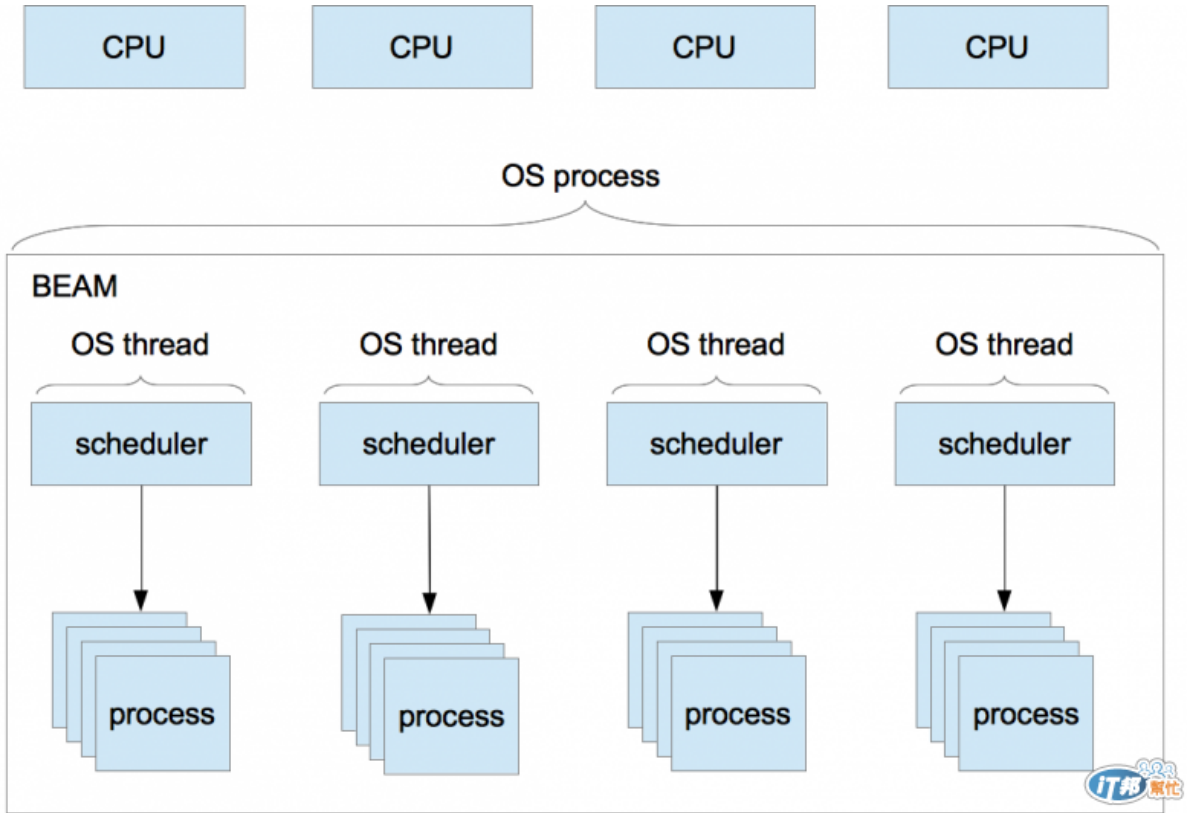
Como funcionan los procesos

- Es el actor en el “Actor concurrency model”
- Cada proceso ejecuta una tarea específica
- Para un proceso haga su tarea le debemos envíar un mensaje y responde con otro mensaje
- Internamente el mensaje recibido es “pattern matched”
- Los procesos no comparten información entre procesos



Demo

Ejercicio de proceso
Ejercicio aplicación simple - sin concurrencia



Concurrency - Erlang VM

http://ithelp.ithome.com.tw/articles/10161709

OTP - Open Telecom Platform

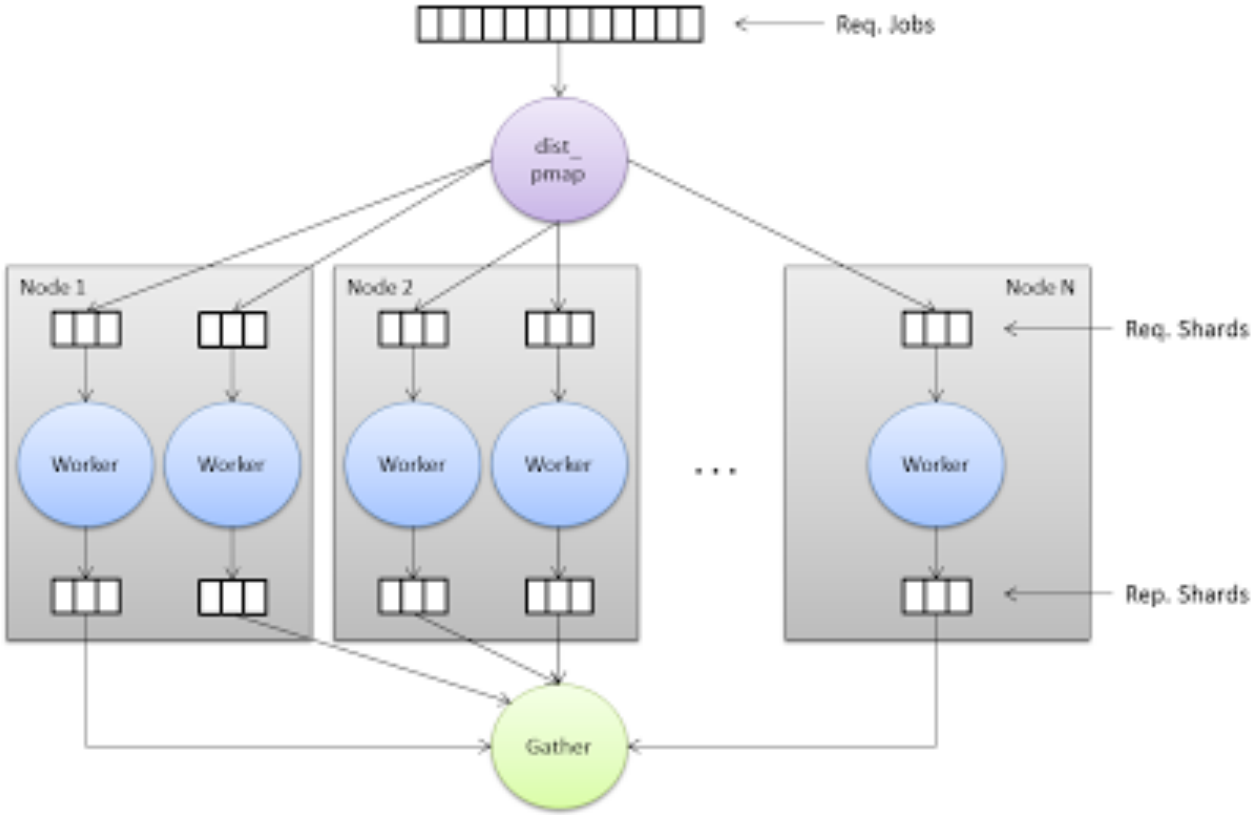
- Framework para construir aplicaciones fault-tolerant, escalables y distribuidas.
- Incluye
  - Erlang Runtime
  - OTP components
  - Base de Datos

Abstracciones de procesos Elixir - OTP

- **Task** - ejecuta un trabajo en particular de manera async (Task.async/Task.await)
- **Agent** - nos permite mantener el estado (“variable”)
- **Supervisor** - behavior - es un proceso que supervisa otro proceso.
- **GenServer** - behavior - es un proceso que mantiene el estado y permite ejecutar código de manera asíncrona

Demo

Ejercicio haciendo concurrencia con Task
Ejercicio de concurrencia con pmap
Ejercicio de concurrencia con lista de workers



Escalable - Alta disponibilidad - Sistema Distribuido

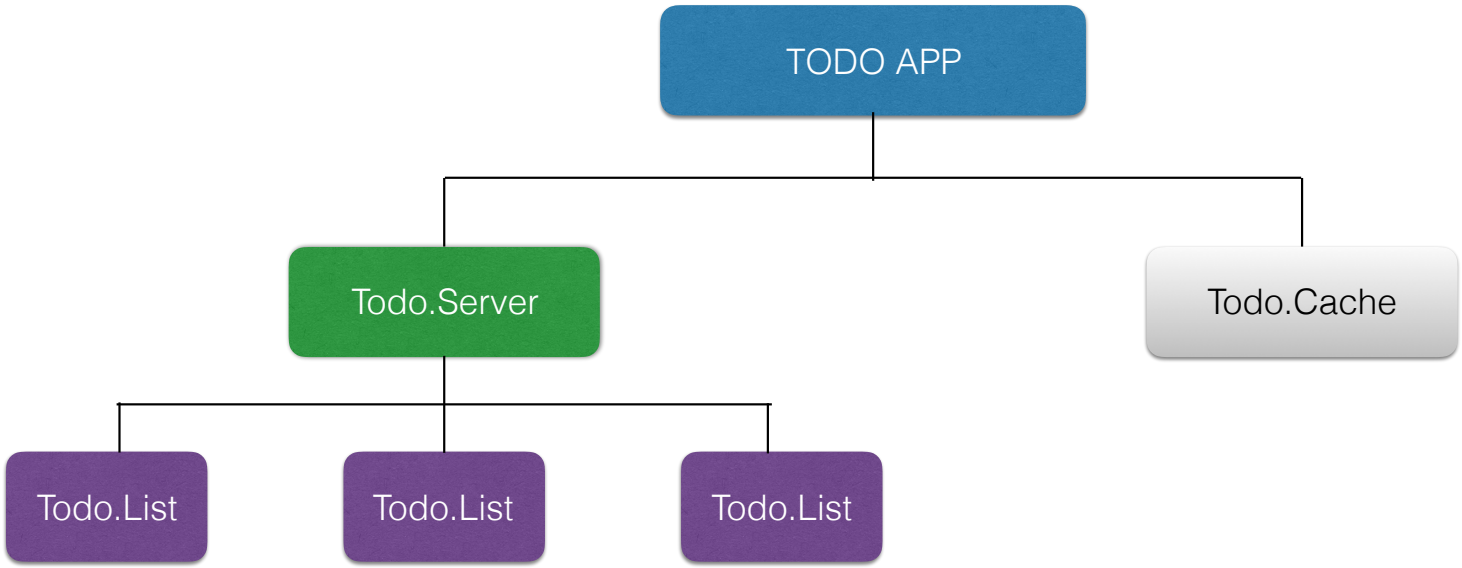
http://cabol.github.io/posts/2013/11/24/programming-languages-and-multicore-crisis.html

Demo

Ejecución de Código Distintos Nodos

Demo OTP Application

Application TODO



http://learnelixir.tv

Phoenix Framework

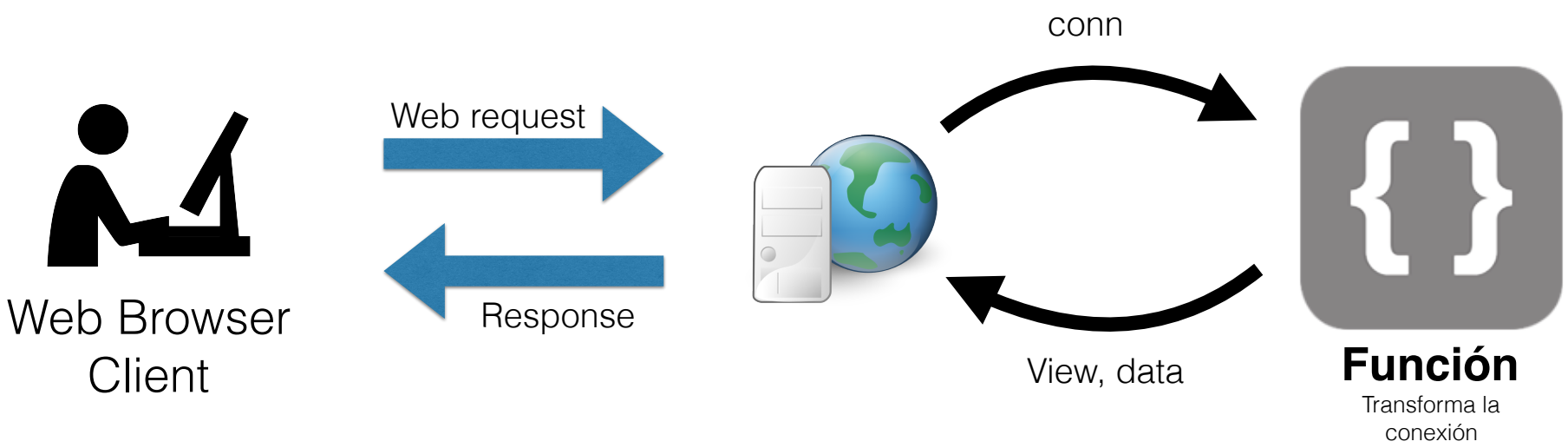
Phoenix Framework

- Chris McCord - Creador
- Desarrollado en Elixir
- Web Development Framework
- Modelo - Vista - Controlador (MVC)
- Muy parecido a Ruby on Rails
- Ecto <-> Active Record
- Base Datos Postgresql por defecto
- Cowboy Web Server -> Erlang

Instalación Phoenix

- http://www.phoenixframework.org/docs/installation
- Instalación PostgreSQL

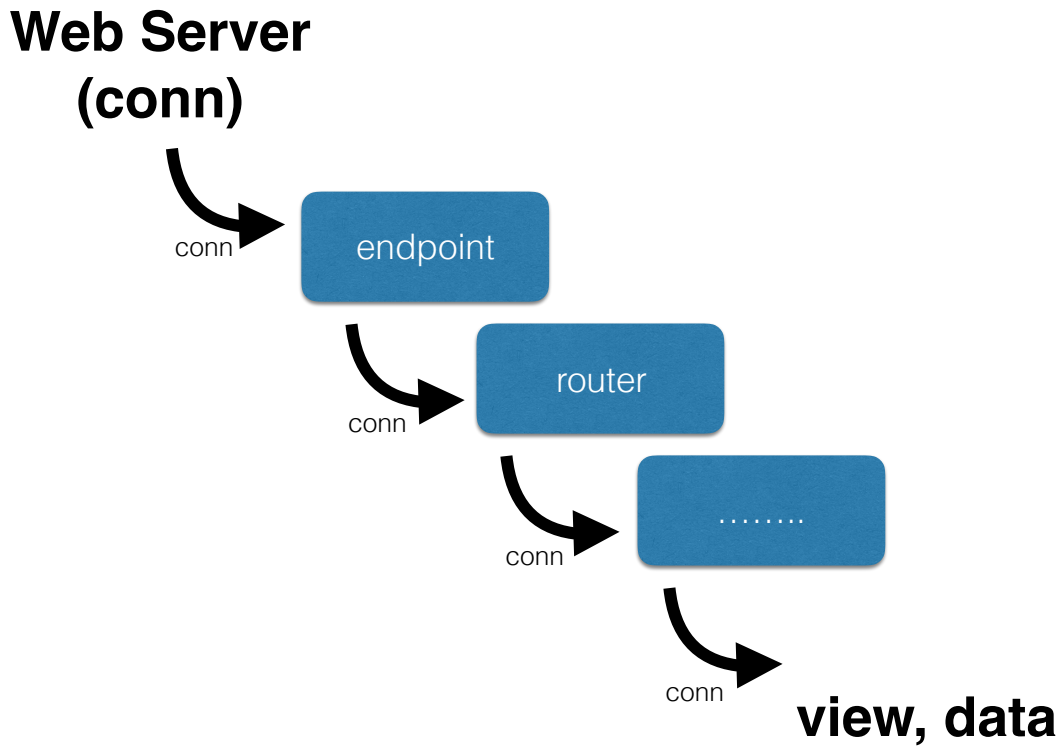
Concepto general



Como funciona?

- Plug - es una función que recibe una conexión y devuelve una conexión. Nos permite transformar la conexión.

connection
    |> endpoint
    |> router
    |> pipelines
    |> controller
    |> common_services
    |> action
    |> view
    |> template



Que es un conn?

- Es una estructura de datos (map) de elixir que contiene todos los datos pertinentes al request y response y que es transformada a través de los Plugs.
  - Request Fields (host, method, port, req_headers)
  - Fetchable Fields (params)
  - Response Fields (resp_body, resp_headers)
  - Connection Fields (assigns - user data, state)

Demo

Creación aplicación básica Phoenix Framework