INFINUM

# Introduction to Elixir and Phoenix

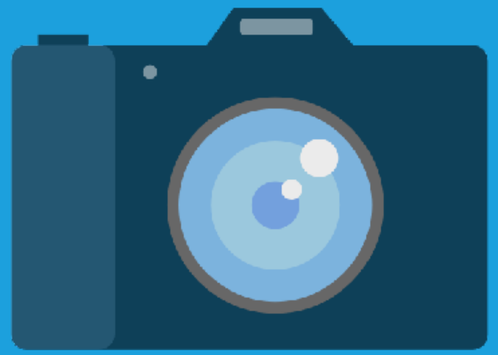
VLADIMIR ROSANČIĆ
RAILS/PHOENIX DEVELOPER

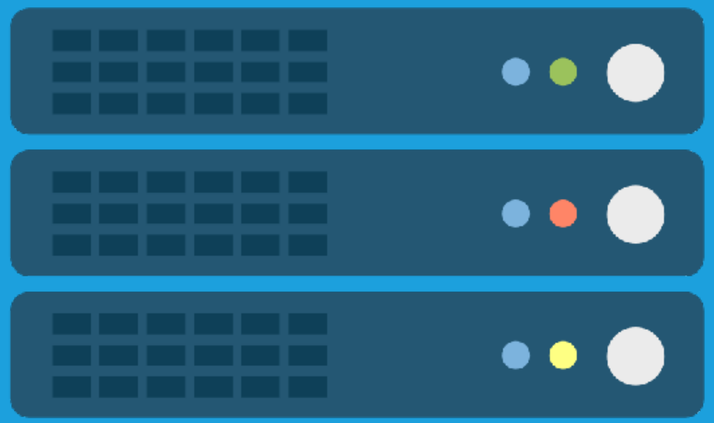
# STRUCTURE

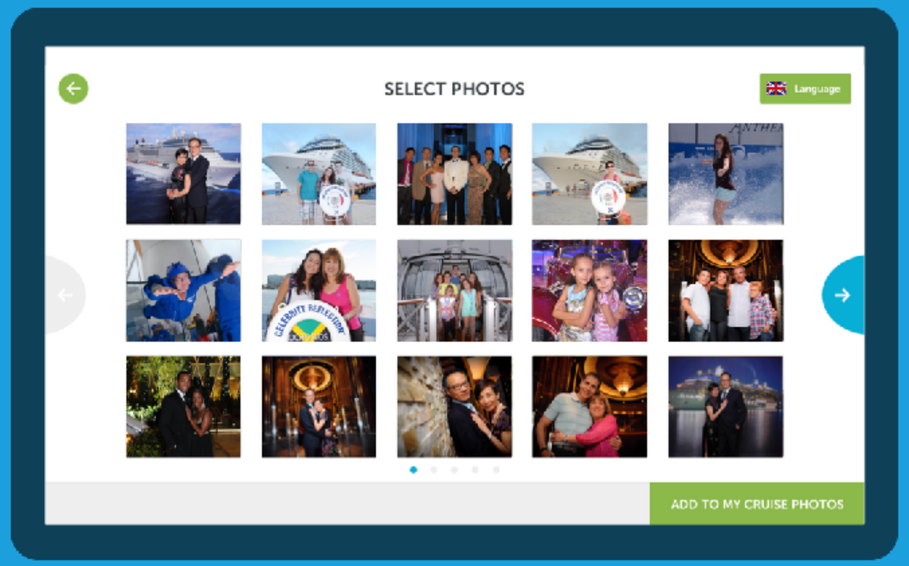
1. Decision
2. Elixir
3. Phoenix
4. Libraries

01

# WHY DID WE DECIDE TO USE ELIXIR/PHOENIX?






SELECT PHOTOS
Language
ADD TO MY CRUISE PHOTOS

300 GB photos per cruise
20M photos per year
80K passengers per week

# Results

Stable system
Great performance
Good codebase
Productivity at the high level


HAPPY DEVELOPER
imgflip.com

The Pragmatic Programmers

# Programming Elixir

Functional
|> Concurrent
|> Pragmatic
|> Fun

Dave Thomas

Foreword by
José Valim,
Creator of Elixir

*edited by Lynn Beighley*

The Pragmatic Programmers

Your Elixir Source

# Programming Phoenix

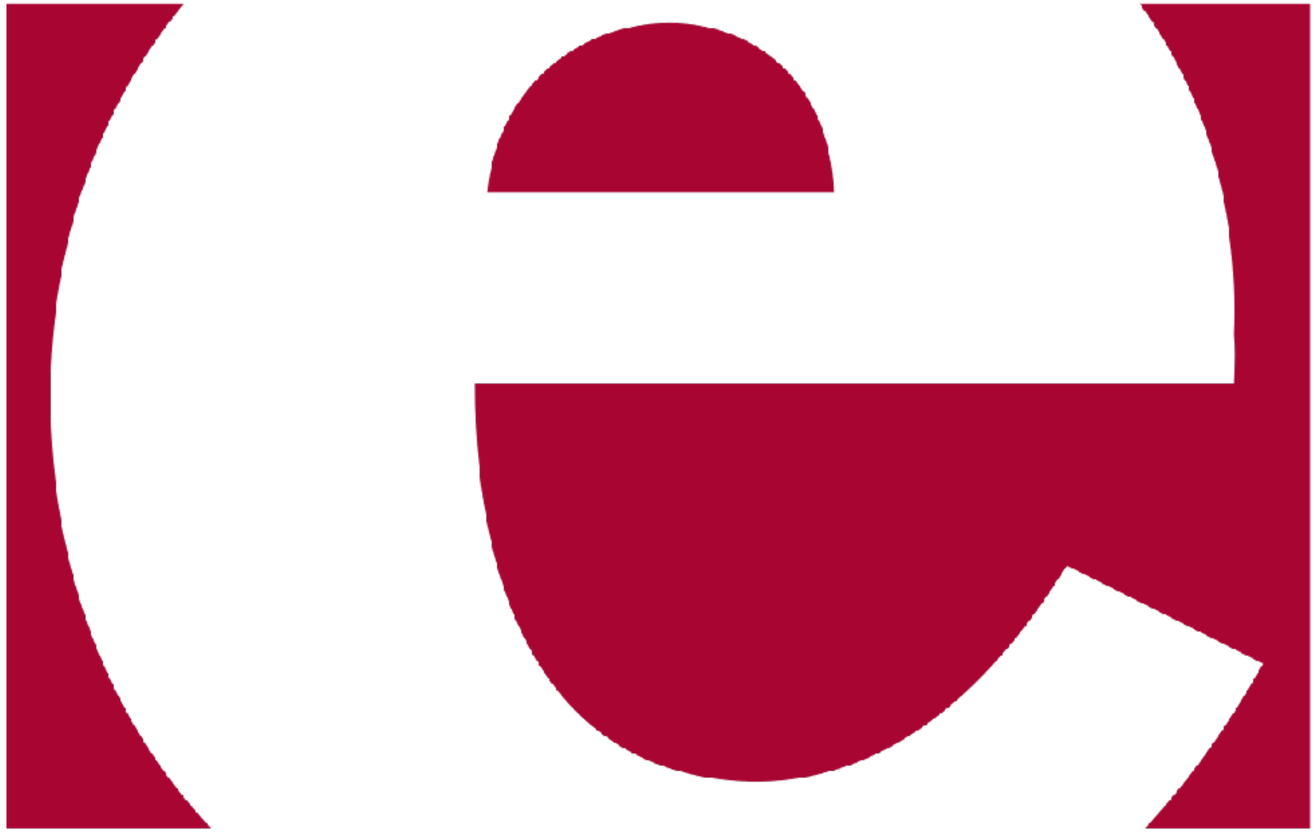
Productive |> Reliable |> Fast


Chris McCord,
Bruce Tate,
and José Valim

*edited by Jacquelyn Carter*

02

# ELIXIR



ERLANG

*Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability.*

highly scalable

# fault tolerant

# hot code swapping

distributed

bad coding experience

# WhatsApp

# Jose Valim

+


=

elixir

highly scalable

# fault tolerant

# hot code swapping

distributed

great coding experience

# Elixir is...

# ... functional

## OO:

```
passenger = new Passenger()

passenger.sail(cruise)

“PASSENGER”.downcase
```

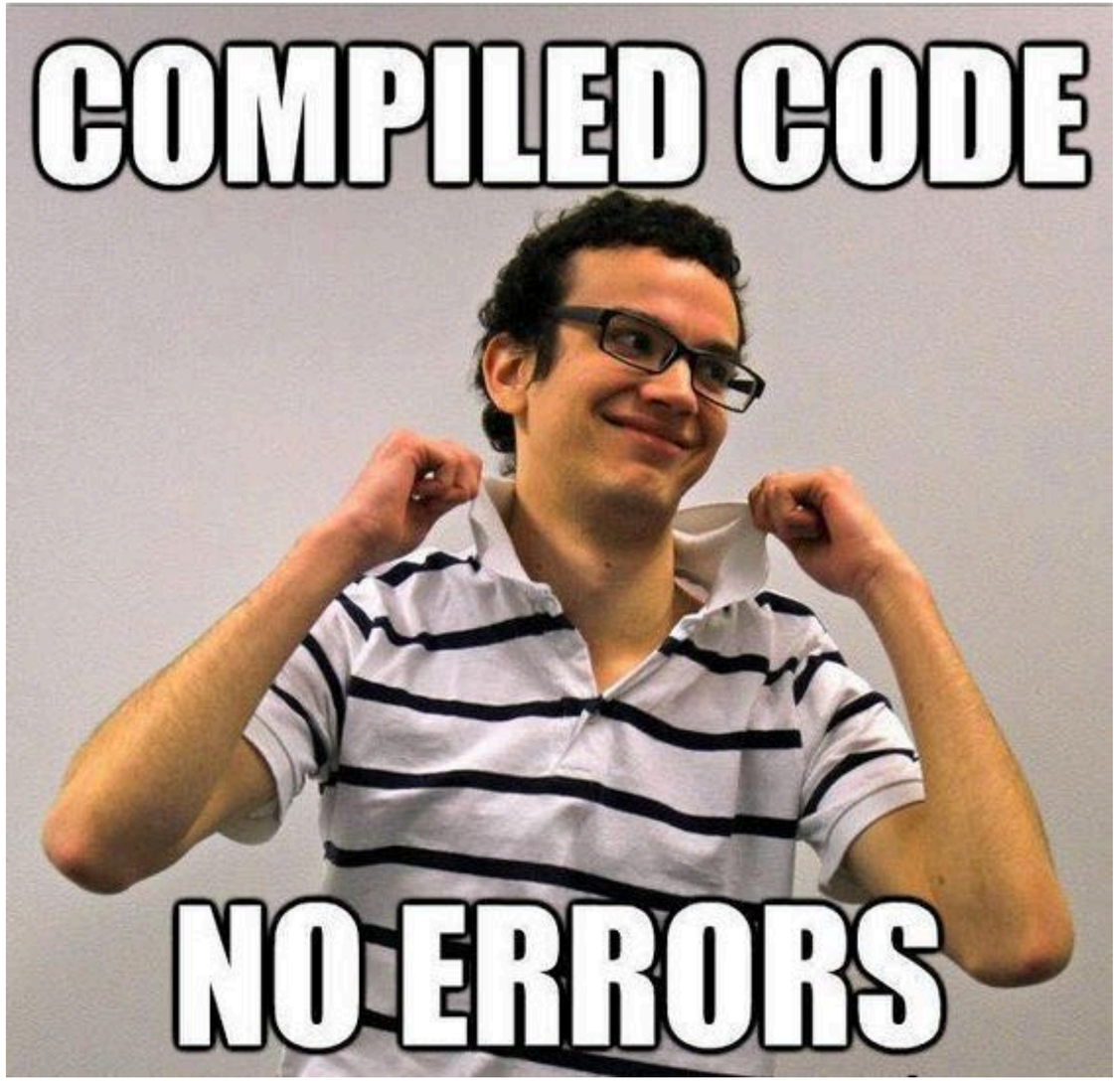
## FP:

```
passenger = Passenger.new()

Passenger.sail(passenger, cruise)

String.downcase(“PASSENGER”)
```

... compiled


COMPILED CODE
NO ERRORS

... dynamic

```
age = 37

name = “John Doe”
```

# … immutable

```
user = %{email: "test@example.com"}

SomeModule.change_email(user, "new@example.com")

user.email # => "test@example.com"
```

```
user = %{email: "test@example.com"}

user = %{email: "new@example.com"}

user.email # => "new@example.com"
```

# Syntax

```elixir
defmodule CruiseApp.Passenger do
  alias CruiseApp.Cruise

  def sail(passenger, cruise) do
    IO.puts "#{passenger.name} is going to sail on the #{ship_name}."
  end

  defp ship_name(cruise) do
    Cruise.get_ship_name(cruise)
  end
end

CruiseApp.Passenger.sail(passenger, cruise)
# Vladimir Rosancic is going to sail on the Harmony of the Seas.
```

# Basic data types

```
1
3.14
:atom
"string"
```

```
{1, 2, 3}
[1, 2, 3]
```

```
%{"key1" => "value", "key2" => 2}
%Passenger{name: "Vlado", age: 28}
```

# Piping |>

```
arg1 |> fun(arg2, arg3)

fun(arg1, arg2, arg3)
```

```
params
|> Passenger.new()
|> Passenger.sail(cruise)
|> Passenger.debark()
```

```
Passenger.debark(
  Passenger.sail(
    Passenger.new(params),
    cruise
  )
)
```

# Pattern matching

```
name = "John Doe"
```

```elixir
{status, message} = {:ok, "Success"}

status # => :ok
message # => "Success"
```

```
{:ok, message} = {:ok, "Success"}

{:ok, message} = {:error, "Failure"}
# ** (MatchError) no match of right
# hand side value: {:error, "Failure"}
```

```elixir
case File.read(path) do
  {:ok, file} -> do_something(file)
  {:error, message} -> message
end
```

```elixir
def sail(%{name: name, age: age}) do
  ...
end
```

```elixir
def sail(_, %{name: "Voyager"}) do
  IO.puts "Voyager currently does not sail"
end

def sail(passenger, ship) do
  ...
end
```

```ruby
def sail(%{name: _ , age: age}, _) when age < 18 do
  IO.puts "You must sail with your parents"
end

def sail(passenger, ship) do
  ...
end
```

# Concurrency

03

# PHOENIX

# Chris McCord

# Plug

```elixir
def some_plug_function(conn, opts) do
  transform conn
  return conn
end
```

```elixir
def put_headers(conn, _opts) do
  Plug.Conn.put_resp_header(
    conn,
    "some_header",
    "some_value"
  )
end
```

```
%Plug.Conn{
  method: ...,
  host: ...,
  params: ...,
  req_headers: ...,
  request_path: ...,
  resp_headers: ...,
  resp_body: ...,
  resp_cookies: ...,
  status: ...
}
```

conn → endpoint → router
→ controller → conn

# Endpoint

```elixir
defmodule MyApp.Endpoint do
  use Phoenix.Endpoint, otp_app: :my_app

  plug Plug.Static,
    at: "/", from: :my_app, gzip: false,
    only: ~w(css fonts images js favicon.ico robots.txt)

  if code_reloading? do
    plug Phoenix.CodeReloader
  end

  plug Plug.RequestId
  plug Logster.Plugs.Logger, formatter: Logster.JSONFormatter

  plug Plug.MethodOverride
  plug Plug.Head

  plug MyApp.Router
end
```

# Router

```elixir
defmodule MyApp.Router do
  use MyApp, :router

  pipeline :browser do
    plug :accepts, ["html"]
    plug :fetch_session
    plug :protect_from_forgery
    plug :put_secure_browser_headers
  end

  pipeline :api do
    plug :accepts, ["json"]
    plug Guardian.Plug.EnsureAuthenticated
  end

  scope "/", MyApp do
    pipe_through :browser

    get "/", HomeController, :index
    get "/passengers", PassengerController, :index
    get "/passengers/:id", PassengerController, :show
  end

  scope "/api", MyApp.Api do
    pipe_through :api

    post "/passenger", PassengerController, :create
  end
end
```

# Controller

```elixir
defmodule MyApp.Api.PassengerController do
  use MyApp.Web, :controller

  alias MyApp.CreatePassenger
  alias MyApp.PassengerSerializer
  alias MyApp.ErrorSerializer

  def create(conn, params) do
    case CreatePassenger.run(params) do
      {:ok, passenger} ->
        conn
        |> put_status(201)
        |> json(PassengerSerializer.serialize(passenger))
      {:error, message} ->
        conn
        |> put_status(422)
        |> json(ErrorSerializer.serialize(error))
    end
  end
end
```

# View

```
<div class="container">
  <div class="passengers">
    <%= for passenger <- @passengers do %>
      <h3><%= full_name(passenger) %></h3>
      <p><%= passenger.description %></p>
    <% end %>
  </div>
</div>
```

```elixir
defmodule MyApp.PassengerView do
  use MyApp, :view

  def full_name(passenger) do
    "#{passenger.first_name} #{passenger.last_name}"
  end
end
```

# Data Layer


Ecto.Repo

Ecto.Schema

Ecto.Changeset

Ecto.Query

```
from p in Passenger,
  join: s in assoc(p, :ship),
  where: p.age > 18 and
    s.name == "Harmony of the Seas",
  select: %{
    name: p.name,
    email: p.email,
    age: p.age
  }
```

# Channels

# 04 LIBRARIES WE USE

# GUARDIAN

## Authentication

https://github.com/ueberauth/guardian

# JA SERIALIZER

## JSON API Serialization / Deserialization

https://github.com/vt-elixir/ja_serializer

# BAMBOO

## Email Library

https://github.com/thoughtbot/bamboo

# EQX

## Sidekiq-like background job library

https://github.com/akira/exq

# POLICY WONK

## Authorization

https://github.com/boydm/policy_wonk

# QUANTUM

## Cron-like job scheduler

https://github.com/c-rack/quantum-elixir

# CREDO

## Static code analysis tool

https://github.com/rrrene/credo

INFINUM

# Thank you!

Visit infinum.co or find us on social networks:

infinum.co infinumco infinumco infinumco infinum