Rewriting a legacy application in Elixir

The good, the bad and the ugly

Raphael Costa
CTO @ Eficiência Fiscal

$ whoami

“Erlang + Ruby”

EFICIÊNCIA

F I S C A L

2600 hours

per year/company

100 Startups to Watch

- Not a “webscale” case
  - 1300 companies using our services
  - 900 active users

- Not a “webscale” case
  - 1300 companies using our services
  - 900 active users
- High customer-to-data ratio
  - 80 GB of database
  - 15 million items already processed

The Problem

The Problem

- Built on top of a legacy multi-tenant monolith

The Problem

- Built on top of a legacy multi-tenant monolith
- Hard to maintain and code rot spreading fast

The Problem

- Built on top of a legacy multi-tenant monolith
- Hard to maintain and code rot spreading fast
- Had paying users

The Problem

- Built on top of a legacy multi-tenant monolith
- Hard to maintain and code rot spreading fast
- Had paying users
- Small team (sometimes just me)

The Watcher

The Watcher

1. Make a request to an endpoint
2. Parse the response XML
3. Check if some data changed
4. Send a email with the changes
5. Repeat in 1 hour

The Watcher - Results

- Took 13 commits and 4 days
- First deployed Aug 18th, 2015
- Single commit one year later to refactor
- It’s still up and running to this day

The first endpoint

The first endpoint

1. Receive two text files
2. Parse and cross reference them
3. Put the built dataset through a lengthy audit process

The first endpoint

1. Receive two text files
2. Parse and cross reference them
3. Put the built dataset through a lengthy audit process
4. Make sure the thing runs until the end

Phoenix 1.0 – the framework for the modern web just landed

By Chris McCord · 2015-08-28 · v1.0.0

```
def index(conn, %{"tag_id" => tag_id}) do
  posts = Repo.all from p in Post,
            where: p.tag_id == ^tag_id,
          order_by: [desc: p.updated_at],
            select: p

  render conn, "index.html", posts: posts
end
```

```
def render("show.json", %{user: user, posts: posts}) do
  %{
    id: user.id,
    name: formatted_name(user),
    email: user.email,
    posts: render_many(posts, PostView, "show.json")
  }
end
```

```
1[||||||||95.2%]
2[||||||||70.7%]
3[||||||||94.6%]
4[||||||||71.7%]
M[|||6606/8192MB]
S[|||1381/3072MB]
```

After a year and a half of work, 2500 commits, and 30 releases, Phoenix 1.0 is here! With 1.0 in place, Phoenix is set to take on the world whether you're building APIs, HTML5 applications, or network services for native devices. Written in Elixir, you get beautiful syntax, productive tooling and a fast runtime. Along the way, we've had many success stories of companies using phoenix in production, and two ElixirConf's where we showed off Phoenix's progress.

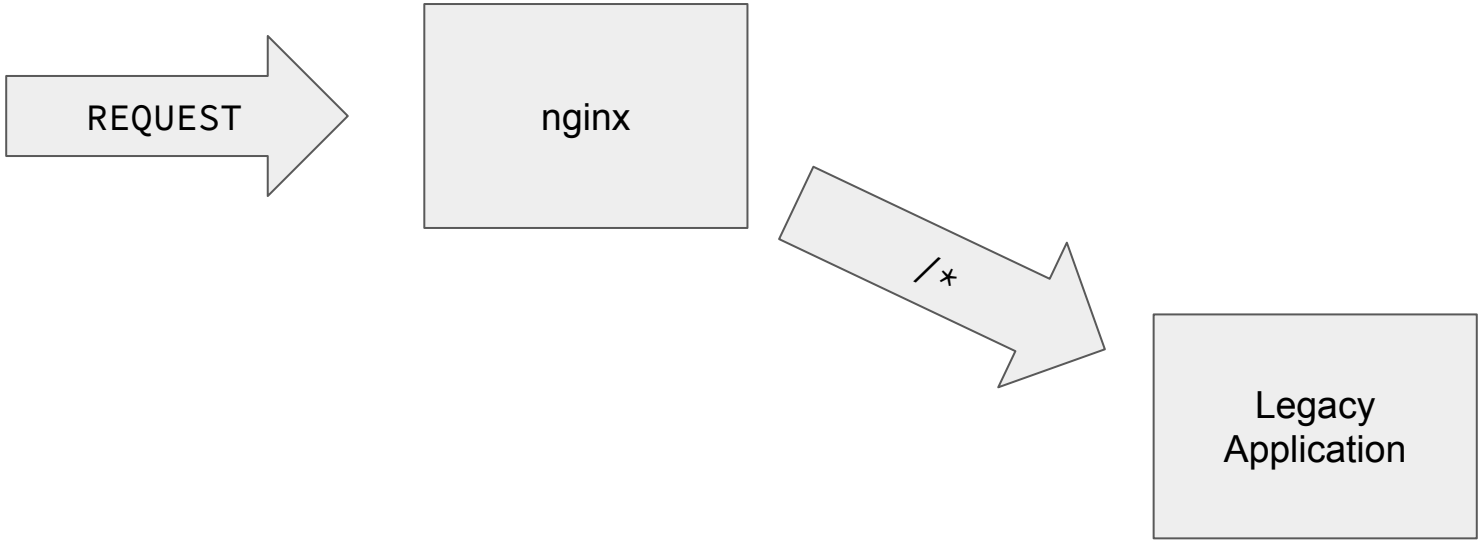
The first endpoint - Results

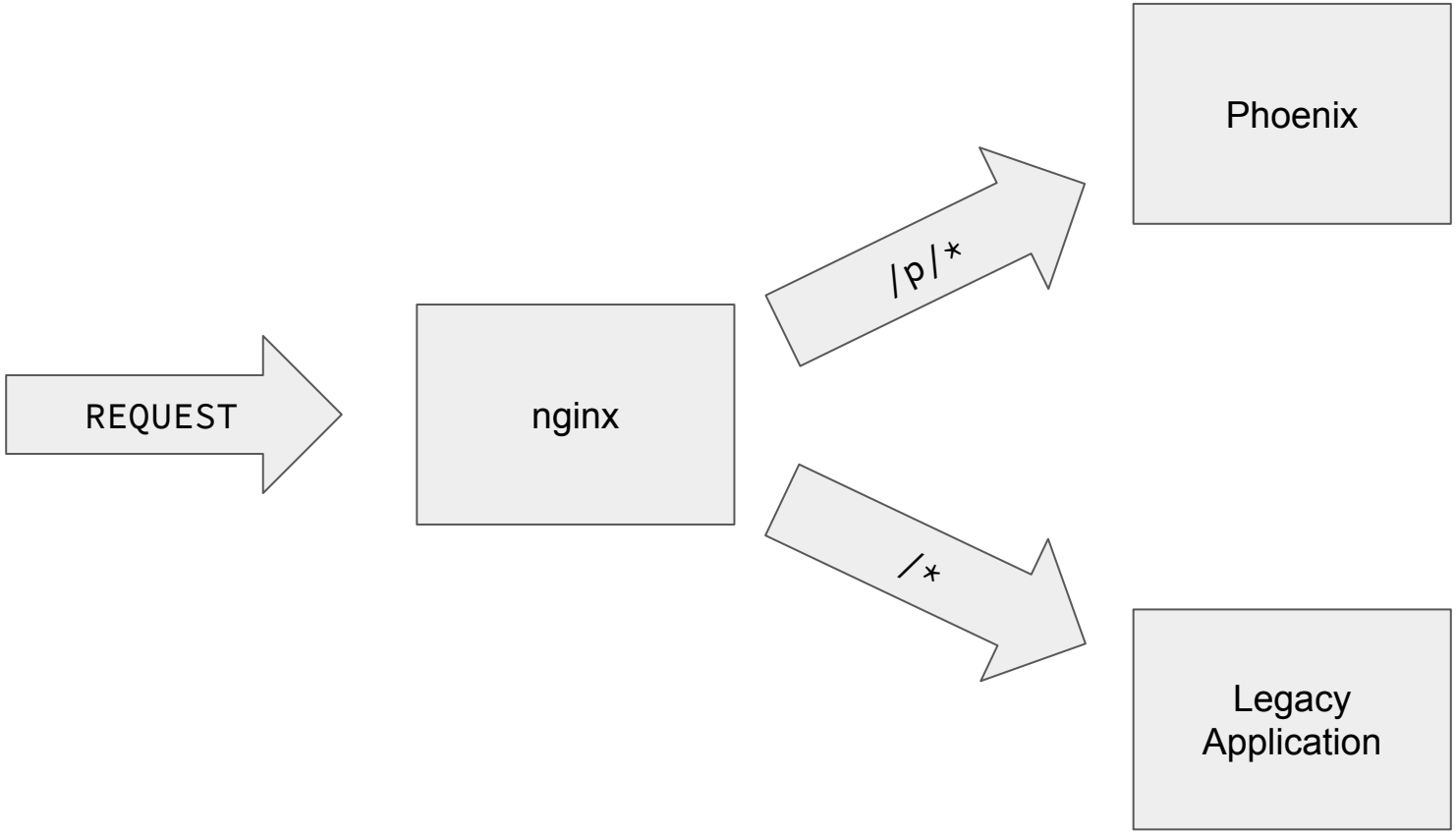
- A lot faster than anything else we had ever built
- Used websockets for live updates of the audit process

The first endpoint - Results

- A lot faster than anything else we had ever built
- Used websockets for live updates of the audit process
- The thing ran until the end!

The migration begins





We had two problems

We had two problems

- The legacy application was too messy for the “just migrate the logic” strategy

We had two problems

- The legacy application was too messy for the “just migrate the logic” strategy
- The team was seeing the Phoenix app as secondary

Adjusting the plan

Adjusting the plan

- Use Phoenix as a reverse proxy to the legacy application

poteto / **terraform**

Watch 14 | Star 279 | Fork 7

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights

A simple plug for incrementally transforming an API into Phoenix. Check out the blog post: https://medium.com/@sugarpirate/rise-…

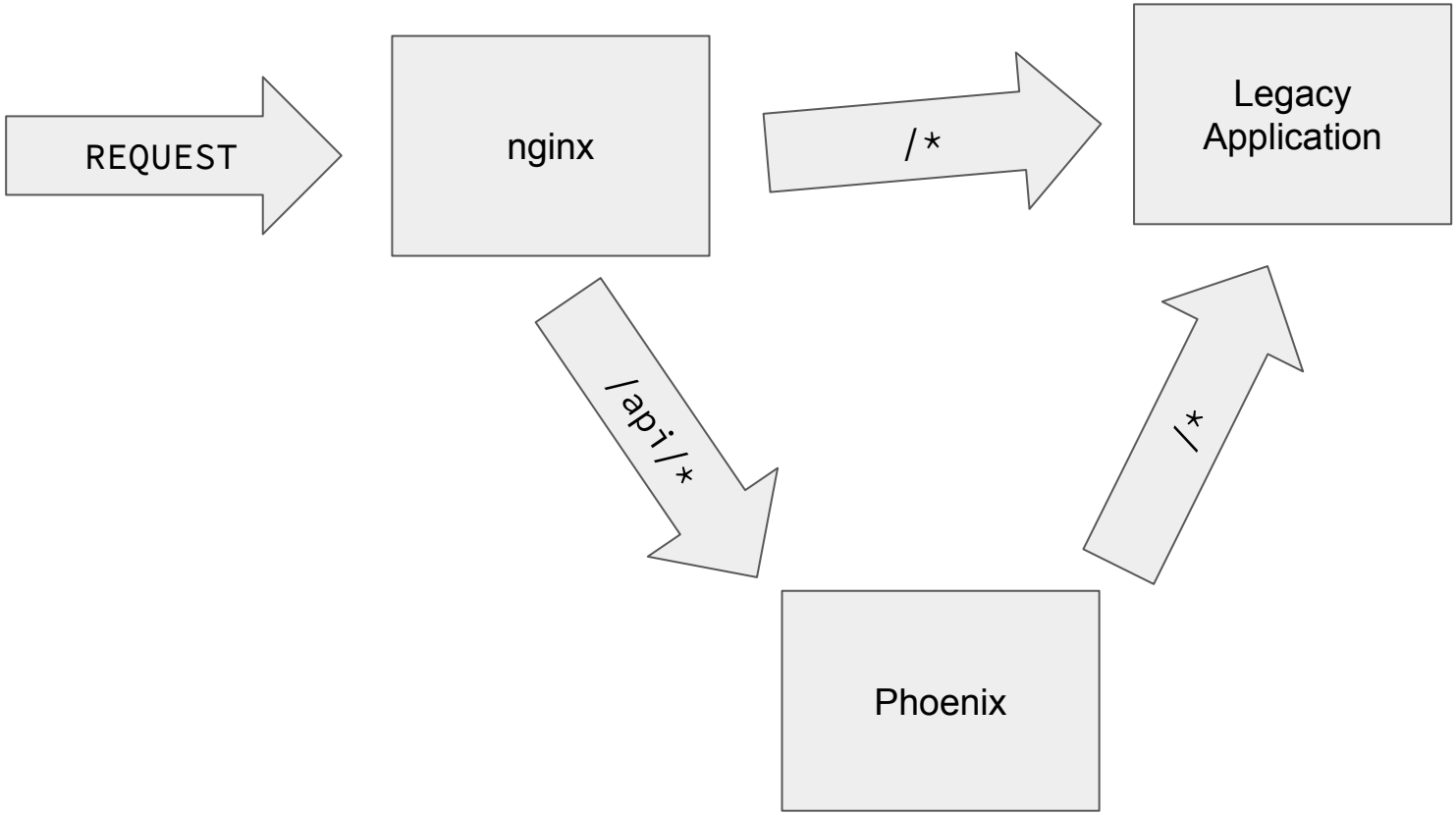
elixir phoenix plug reverse-proxy

24 commits | **1** branch | **0** releases | **4** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

bgmarx and **poteto** update version (#23) — Latest commit cc78b21 on Jun 5, 2017

config	Init	2 years ago
lib	Update deps and add moduledoc (#18)	11 months ago
test	Convert test to use Plug.Router in the Terraformer (#10)	2 years ago
.gitignore	Init	2 years ago
.tool-versions	Update deps and add moduledoc (#18)	11 months ago
LICENSE.md	Add license (#5)	2 years ago
README.md	update version (#23)	10 months ago
circle.yml	Update deps and add moduledoc (#18)	11 months ago



Less than 1ms

overhead

Adjusting the plan

- Use Phoenix as a reverse proxy to the legacy application
- No more just porting the old solutions, but actually rethinking the logic

The Elixir Way

Adjusting the plan

- Use Phoenix as a reverse proxy to the legacy application
- No more just porting the old solutions, but actually rethinking the logic
- Model the application as an umbrella to support the background services

Switching the architecture

Switching the architecture

- Monolith
- Microservices

Switching the architecture

- ~~Monolith~~
- Microservices

Switching the architecture

- ~~Monolith~~
- ~~Microservices~~

Switching the architecture

- ~~Monolith~~
- ~~Microservices~~
- ???

Switching the architecture

- ~~Monolith~~
- ~~Microservices~~
- *Umbrellas*

The Results

The Results

- Team’s morale went up

The Results

- Team’s morale went up
- The code became easier to refactor and reason about

The Results

- Team’s morale went up
- The code became easier to refactor and reason about
- Things were moving smoothly and management was happy

Phoenix 1.3

ElixirConf 2017

OWLTURD.COM

Contexts FTW!

Our current status

Our current status

- Over 36k LOC (and growing)
- 3 developers + 2 interns (all using Elixir)
- 11 apps in our umbrella

Things that might help

Things that might help

- `credo`

rrrene / **credo**

Watch 54 | Star 2,390 | Fork 169

Code | Issues 42 | Pull requests 9 | Projects 1 | Wiki | Insights

A static code analysis tool for the Elixir language with a focus on code consistency and teaching. http://credo-ci.org/

elixir | code-analysis | static-analysis | linter | credo

1,428 commits | **16** branches | **62** releases | **93** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

rrrene Fix false positive on UnusedFunctionReturnHelper — Latest commit 517699f 2 days ago

assets	Add ExDoc	19 days ago
config	Move config to module_attributes	2 years ago
lib	Fix false positive on UnusedFunctionReturnHelper	2 days ago
test	Fix false positive on UnusedFunctionReturnHelper	2 days ago
.credo.exs	Remove NameRedeclaration checks	2 months ago
.formatter.exs	Reformat code with 120 chars per line	2 months ago
.gitignore	Add test_on_projects script	16 days ago
.template.check.ex	WIP: Store SourceFile fields source, lines and ast in ETS tables	11 months ago

credo

- Ensure a consistent codebase
- Find simple problems more easily
- Avoid discussions regarding style

Things that might help

- `credo`
- Property testing

whatyouhide / **stream_data**

Watch 20 | Star 223 | Fork 27

Code | Issues 8 | Pull requests 1 | Projects 0 | Wiki | Insights

Data generation and property testing for Elixir

elixir | property-based-testing | property-testing | data-generation | quickcheck

257 commits | **1** branch | **7** releases | **18** contributors

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

obrok and **whatyouhide** Fix some small mistakes in the documentation (#93) ... Latest commit `312d00c` 18 days ago

examples	Blame exceptions that are re-raised	4 months ago
lib	Fix some small mistakes in the documentation (#93)	18 days ago
test	Move keyword test to its own file	a month ago
.formatter.exs	Add a .formatter.exs to play with Elixir's new code formatter	6 months ago
.gitignore	First commit	11 months ago
.travis.yml	Remove a useless option when checking for formatted files on CI	2 months ago
CHANGELOG.md	Release v0.4.2	a month ago
README.md	Use property-based testing consistently	6 months ago

Property Testing

- Catching bugs in edge cases
- Test complex features much more easily

Things that might help

- `credo`
- Property testing
- Using Elixir's unique capabilities to solve problems

```
Enum.each(companies, &do_thing/1)
```

```
{:ok, supervisor} = Task.Supervisor.start_link()

supervisor
|> Task.Supervisor.async_stream_nolink(companies, &do_thing/1)
|> Stream.run()
```

Closing thoughts

The Pragmatic Programmers

Your Elixir Source

Adopting Elixir

From Concept to Production

Ben Marx, José Valim, Bruce Tate

edited by Jacquelyn Carter

Thank you!

raphael_vcosta

vidal.raphael@gmail.com

costaraphael