Raxx.Kit

Lean mean web development

crowdhailer.me

Hi

name - Peter Saxton

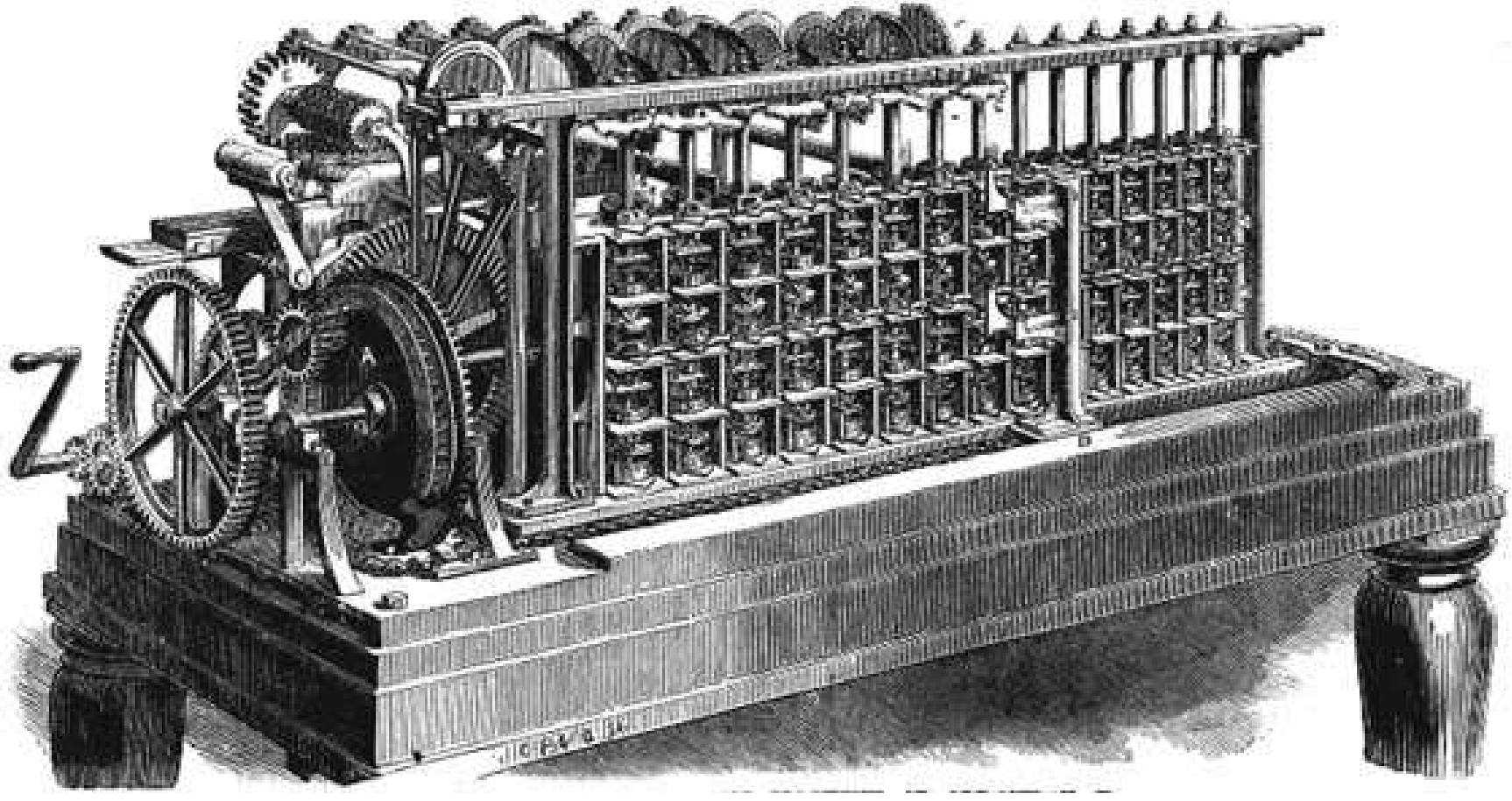
@internets - CrowdHailer

@works - `nil`

Raxx? ✋

Raxx.Kit? ✋

From the beginning



```
mix archive.install hex raxx_kit

mix raxx.new my_app
```

```elixir
defp deps do
  [
    {:ace, "~> 0.18.6"},
    {:raxx_logger, "~> 0.2.2"},
    {:raxx_view, "~> 0.1.7"},
    {:raxx_static, "~> 0.8.3"},
    {:raxx_session, "~> 0.2.0"},
    {:jason, "~> 1.0"},
    {:exsync, "~> 0.2.3", only: :dev},
    # ...
  ]
end
```

mix.exs

What is Raxx?

Raxx is stable

```
{:raxx, "~> 1.0"}
```

In production since 2017

Next door right now.

Raxx is an Interface

Raxx

```
defmodule MyApp.WWW do
  use Raxx.SimpleServer

  def handle_request(_request, _state) do
    %Raxx.Response{status: 200, headers: [], body: "Hello, World!"}
  end
end
```

erlang/OTP

```
defmodule MyServer do
  use GenServer

  def handle_call(:request, _from, state) do
    {:reply, :response, state}
  end
end
```

Raxx is an Interface

```
defmodule MyApp.WWW.Upload do
  use Raxx.Server # ! Raxx.SimpleServer

  def handle_head(%{path: ["upload"] body: true}, _) do
    {:ok, io_device} = File.open("my/path")
    {[], {:file, device}}
  end

  def handle_data(data, state = {:file, device}) do
    IO.write(device, data)
    {[], state}
  end

  def handle_tail(_trailers, state) do
    response(:see_other)
    |> set_header("location", "/")
  end
end
```

Raxx is a Toolkit

```
defmodule MyApp.WWW.Greetings do
  use Raxx.SimpleServer

  def handle_request(request, _state) do
    query = Raxx.get_query(request)

    Raxx.response(:ok)
    |> Raxx.set_header("content-type", "text/plain")
    |> Raxx.set_body("Hello, World!")
  end
end
```

Raxx is imported by default.

Raxx is an Ecosystem

```
defmodule MyApp.WWW.Greetings do
  use Raxx.SimpleServer

  alias Raxx.Session

  use Raxx.View,
    arguments: [:user],
    template: "path/to/greetings_template.html.eex"

  def handle_request(request, %{session_config: conf}) do
    {:ok, session} = Session.extract(request, conf)

    response(:ok)
    |> render(session.user)
  end
end
```

Raxx is an Ecosystem

```
Ace.HTTP.Service.start_link(
  {MyApp.WWW, config},
  port: 8080,
  # ...
)
```

Ace - HTTP/(1 & 2) server to run Raxx applications.

What is Raxx.Kit?

Cowboy -> Ace

Plug -> Raxx

Phoenix -> ???

```
mix raxx.new my_app
  --node-assets
  --ecto
  --docker
  --api
```

- project structure
- live reloading
- templates
- static assets
- testing
- docker integration (`--docker`)
- compiled assets (`--node-assets`)
- database setup (`--ecto`)

Capability
Convenience
Convention

Capability

the power or ability to do something

- Ace
- EExHTML

Convenience

the state of being able to proceed with something without difficulty

- Raxx
- Raxx.Session
- Raxx.View *(works with Plug)*

Convention

behaviour that is considered acceptable or polite to most members of a society

- Raxx.Kit

Principles

Less is More

> The cheapest, fastest, and most reliable components are those that aren't there.

Gordon Bell

Remove unnecessary features

Old

```
@route_name :users
route ["users"], request do
  :GET ->
    # ...
  :POST ->
    # ...
end
```

New

```
def handle_request(%{path: ["users"], method: :GET}, _) do
  # ...
end

def handle_request(%{path: ["users"], method: :POST}, _) do
  # ...
end
```

Delegate

PARCEL

Blazing fast, zero configuration web application bundler

webpack

MODULE BUNDLER

rollup.js

Delegate

```
Supervisor.child_spec({Task, fn() ->
  System.cmd("npm", ["run", "watch:js"], cd: "lib/my_app/www")
end}, id: :watch_js),

Supervisor.child_spec({Task, fn() ->
  System.cmd("npm", ["run", "watch:css"], cd: "lib/my_app/www")
end}, id: :watch_css),
```

--node-assets

Opt in

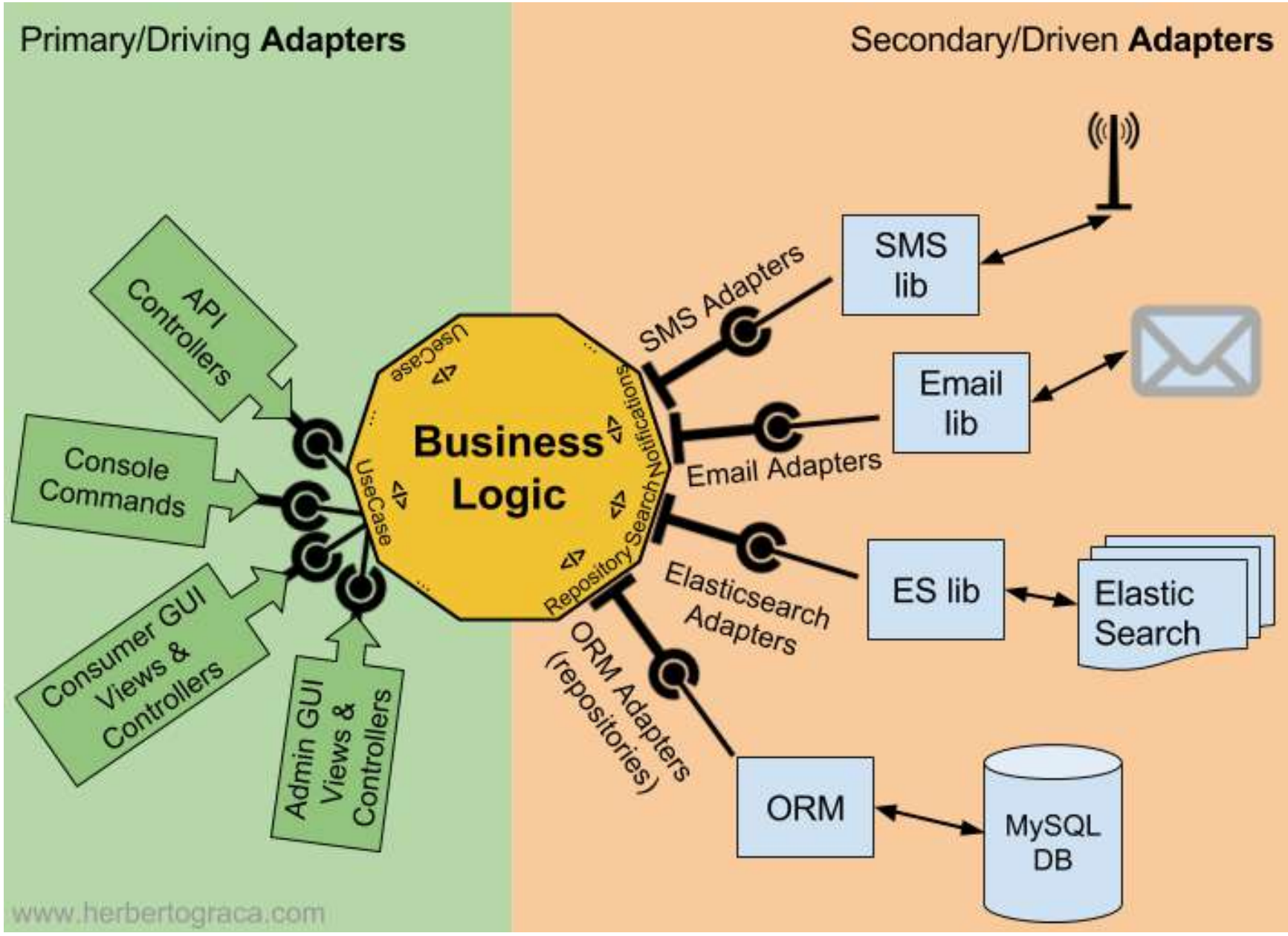
Without a database

```
mix raxx.new my_app

mix phx.new my_app --no-ecto
```

With a database

```
mix raxx.new my_app --ecto

mix phx.new my_app
```

Mind your own business
View
Controller


Primary/Driving Adapters
Secondary/Driven Adapters
API Controllers
Console Commands
Consumer GUI Views & Controllers
Admin GUI Views & Controllers
Business Logic
UseCase
UseCase
Repository
Search
Notifications
SMS Adapters
Email Adapters
Elasticsearch Adapters
ORM Adapters (repositories)
SMS lib
Email lib
ES lib
Elastic Search
ORM
MySQL DB
www.herbertograca.com

- SQL
- NoSQL
- EventSourcing/CQRS
- Memory Image
- Hexagonal Architecture
- Onion Architecture
- Bring your own

Raxx supports all of these architectures...

- SQL
- NoSQL
- EventSourcing/CQRS
- Memory Image
- Hexagonal Architecture
- Onion Architecture
- Bring your own

Raxx supports all of these architectures...

because it assumes none of them

Libraries > Frameworks

```
children = [
  {MyApp.WWW, [[port: 4000]]}
  {MyApp.API, [[port: 4001]]}
  {MyApp.Admin, [[port: 4002]]}
]
```

lib/my_app/application.ex

```
setup do
  {:ok, service} = MyApp.WWW.start_link(%{}, port: 0)
  # etc
end
```

test/my_app/www_test.ex

Functional cohesion

Functional cohesion (best)

Functional cohesion is when parts of a module are grouped because they all contribute to a single well-defined task of the module.

https://en.wikipedia.org/wiki/Cohesion_(computer_science)

Functional cohesion

```
my_app/
├── www/
    ├── actions/
        ├── login.ex
        ├── login.html.eex
```

Functional cohesion

```
defmodule MyApp.WWW.Login do
  use Raxx.SimpleServer
  use Raxx.View, arguments: []

  def handle_request(%{method: :GET}, _state) do
    response(:ok)
    |> render()
  end

  def handle_request(request = %{method: :POST}, _state) do
    # ...
  end
end
```

Future plans

Stability


5 Open
94 Closed
Author
Labels
Projects
Milestones
Assignee
Sort
Raxx.Logger 1.0 Roadmap
help wanted
#179 opened 13 days ago by CrowdHailer
0 of 2
Raxx.View 1.0 Roadmap
help wanted
#178 opened 13 days ago by CrowdHailer
2 of 5
Raxx.Session 1.0 Roadmap
help wanted
#177 opened 13 days ago by CrowdHailer
1 of 6
Benchmark different Raxx.Stack state structures and switch to the fastest one
#147 opened on 2 Nov 2018 by nietaki
1
Add line numbers from template to view.
bug
#132 opened on 9 Sep 2018 by CrowdHailer

Better erlang support

```
handle_request(_Request, _State) ->
  Response = raxx:response(ok),
  raxx:set_body(Response, <<"Hello, World">>).
```

New conventions - Forms

```elixir
defmodule MyApp.API.CreatePost.Form do
  use Ecto.Schema

  @primary_key false
  embedded_schema do
    field(:pass_token, :binary_id)
    field(:idempotency_key, :string)
  end

  defp cast(raw) do
    keys = Map.keys(Map.from_struct(struct(__MODULE__)))
    Ecto.Changeset.cast(__MODULE__, raw, keys)
    |> Ecto.Changeset.apply_action(:insert)
  end
end
```

New conventions - Forms

```
defmodule MyApp.API.CreatePost do
  use Raxx.SimpleServer
  alias MyApp.API.CreatePost.Form

  def handle_request(request, _state) do
    {:ok, payload} = Jason.decode(request.body)
    {:ok, data} = Form.cast(payload)
  end
end
```

New conventions - Error Handling

```
defmodule MyApp.API.CreatePost do
  use Raxx.SimpleServer
  require OK

  def handle_request(request, _state) do
    OK.try do
      payload <- Jason.decode(request.body)
      data <- Form.cast(payload)

    after
      response(:created)

    rescue
      error ->
        MyApp.API.format_error(error)
    end
  end
end
```

add supports for QUIC #128

Open **haoxinst** opened this issue on 13 Dec 2018 · 0 comments

haoxinst commented on 13 Dec 2018

https://tools.ietf.org/html/draft-ietf-quic-http-13

CrowdHailer added **help wanted** **wontfix** labels on 13 Dec 2018

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close issue Comment

More Buzzwords

```
mix raxx.new my_app
  --graphql
  --event_sourcing
  --kubernetes
```

Questions?

@CrowdHailer