2+yo Elixir App in Production

Chat to order

What is this talk about?

- My first Elixir paid job
- An architecture overview
- Experiences and lessons learned

Real world problem

- Restaurants order from Suppliers every day
- Find new supplier takes effort so let’s stick together forever
- Very similar products, tomatoes are not just tomatoes
- Same product but loads of diff packaging 🍌

Real world problem

- Restaurants order by voice phone mail at 11:30pm
- Suppliers listen to orders at 7am
- Suppliers ship what they’ve understand
- Use what you have on their kitchen
- Order => get the product => find out the price $$

What the App is For

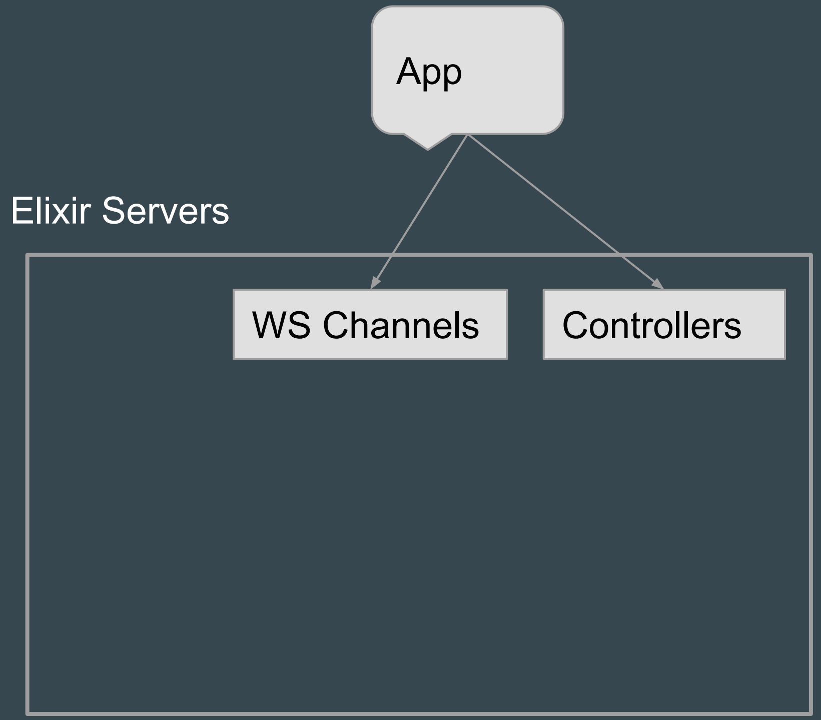
- Chat App with Ordering
- Chefs and sous-chefs <=> Suppliers
- App for the restaurant side
- Email/Sms integration on the supplier side
- Easy new supplier setup

Architecture Overview

- single git repo
- umbrella app (≈15)
- distillery releases (≈4)
- docker containers
- 3x AWS medium size
- CI with auto deploy

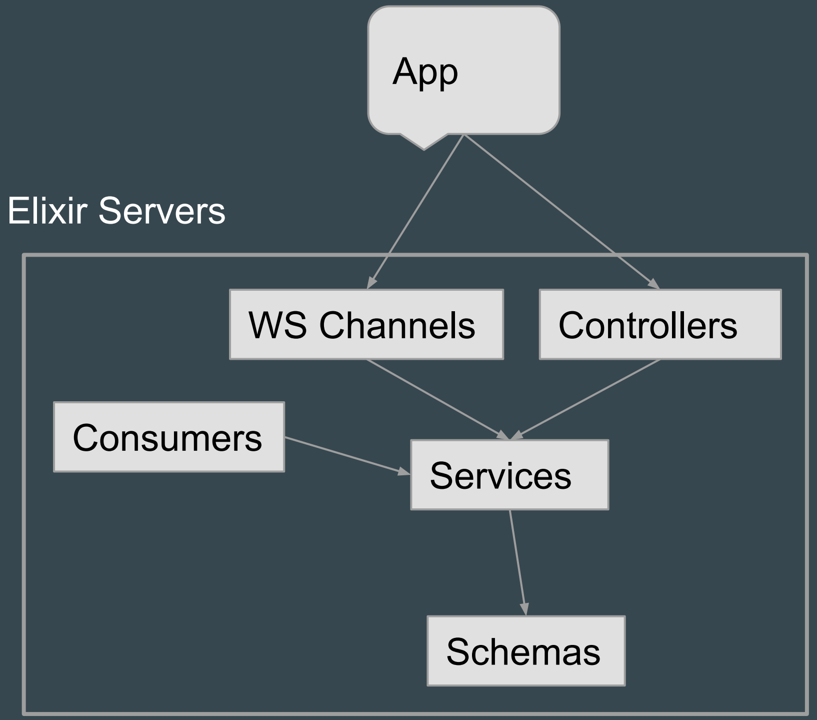
Architecture Overview

- Android/iOS/ReactNative
- Phoenix WS channels
- HTTP API



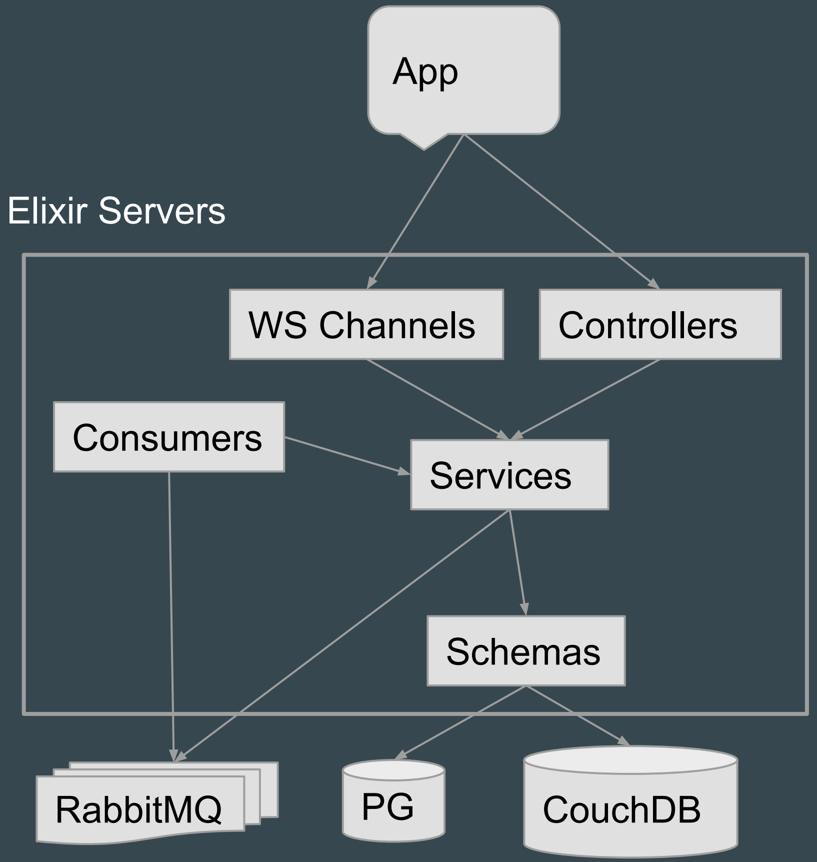
Architecture Overview

- Service modules
- Ecto or non-Ecto schemas
- RabbitMQ Consumers
- Retry a consumer mechanism



Architecture Overview

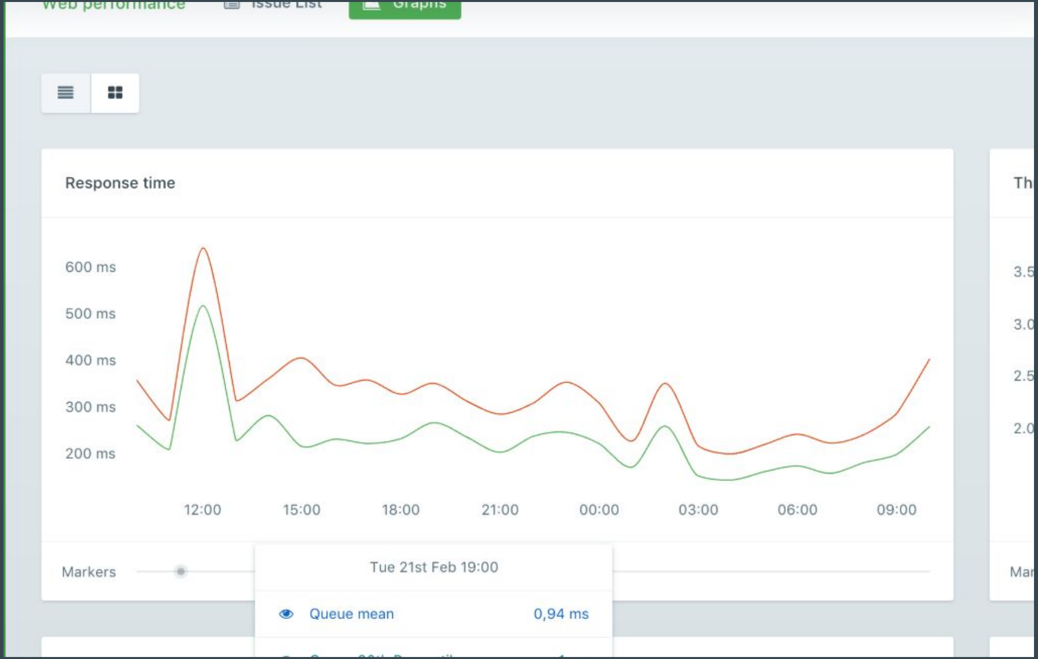
- Regular PG setup, great performance
- RabbitMQ json messages with validation
- legacy CouchDB db accessed by:
    - by bunch of node js apps
    - by android apps via http
    - several performance issues



Lessons learned

Monitoring

- Appsignal => https://appsignal.com/



Tue 21st Feb 19:00

Queue mean 0,94 ms

PostgreSQL performance

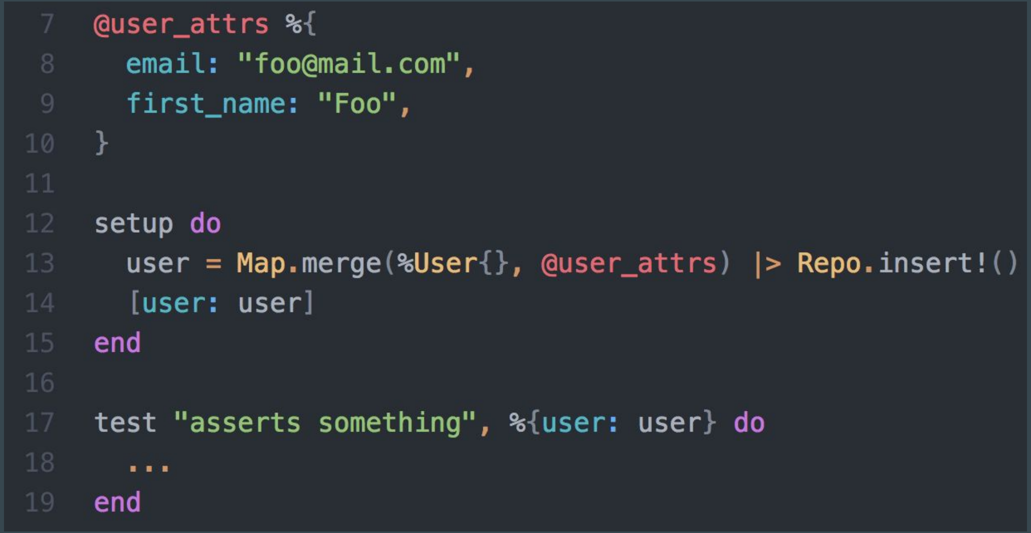
- Small performance issues, high hit => big issues
- High hit

Business Felling

- Backend was a slow team of 2 people
- Frontend was fast with 6 people

Before Factories

- Module attrs with values
- Use setup to build data
- attrs, setup and test are far apart



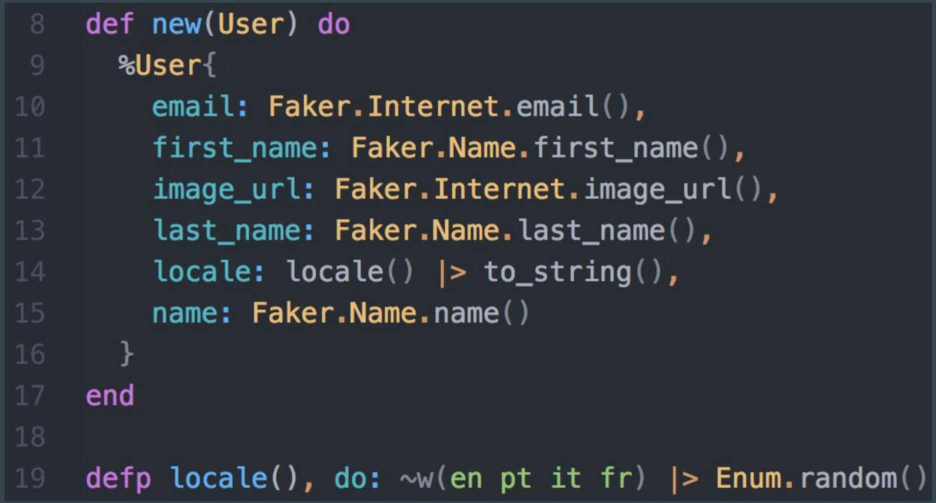
```
@user_attrs %{
  email: "foo@mail.com",
  first_name: "Foo",
}

setup do
  user = Map.merge(%User{}, @user_attrs) |> Repo.insert!()
  [user: user]
end

test "asserts something", %{user: user} do
  ...
end
```

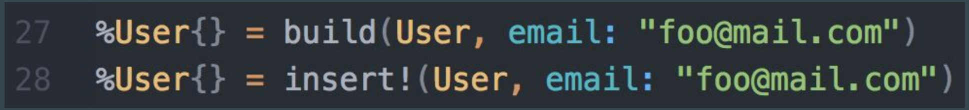
Factories guides for the project

- Keep it simple
- A single factory definition per schema
- Random data all the time
- Build all regular attributes
- Build all belongs_to relation
- Everything else defined on tests



```elixir
def new(User) do
  %User{
    email: Faker.Internet.email(),
    first_name: Faker.Name.first_name(),
    image_url: Faker.Internet.image_url(),
    last_name: Faker.Name.last_name(),
    locale: locale() |> to_string(),
    name: Faker.Name.name()
  }
end

defp locale(), do: ~w(en pt it fr) |> Enum.random()
```



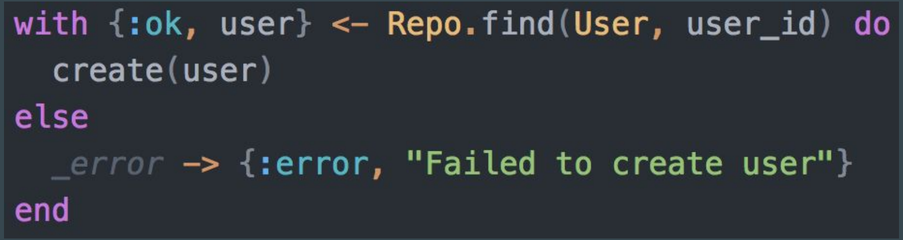
```elixir
%User{} = build(User, email: "foo@mail.com")
%User{} = insert!(User, email: "foo@mail.com")
```

Other Lessons learnt

- You may not need an umbrella app yet ☔
- Check for updates regularly
- Tests are simpler

Expectations and Reality

- Elixir vs Ruby syntax
- A new way to code
- Open source libraries



```
with {:ok, user} <- Repo.find(User, user_id) do
  create(user)
else
  _error -> {:error, "Failed to create user"}
end
```

OMHO

- It’s faster to maintain code
- It’s simpler
- Elixir is great

Thank you very much =>!