Lesson 4: Phoenix Framework

Wannes Fransen & Tom Eversdijk

UC Leuven

2020

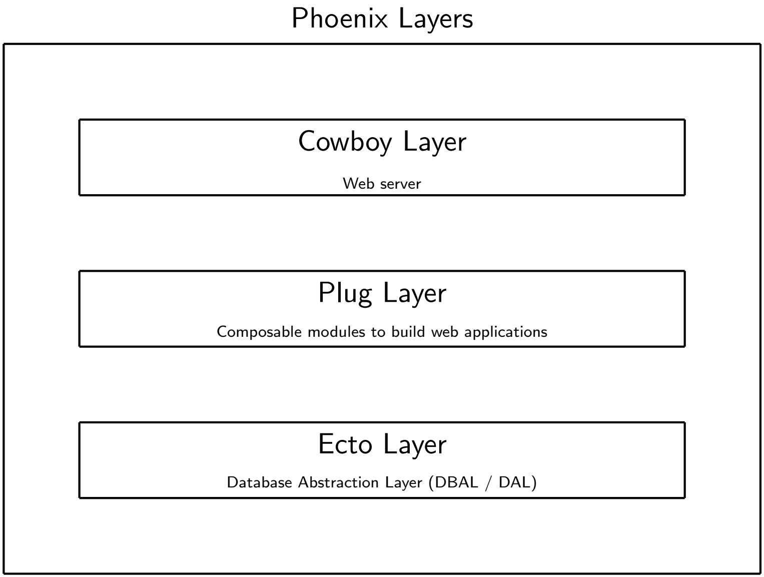
Why a framework?

- Avoid writing the same code over and over again
- Higher interopability between your code & boilerplate code
- Abstractions to e.g. sessions, security, databases, etc...
- Other developers can focus on the important bits

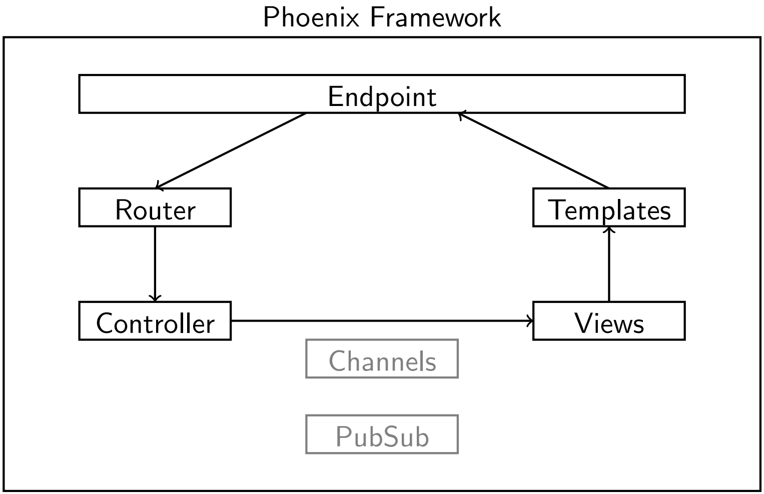
What is Phoenix?

- Web development framework written in Elixir
- Inspired by concepts of Ruby on Rails
- Server Side MVC framework
- High performance & made for realtime features
- Great language/framework for long connections

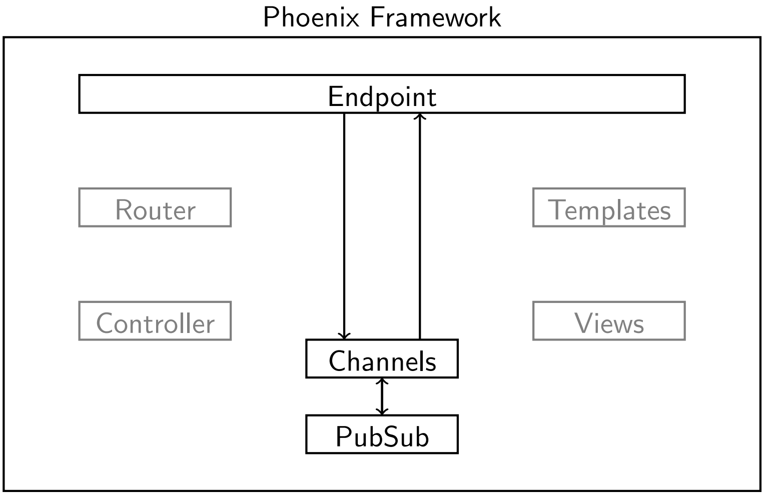
Phoenix layers


Phoenix Layers
Cowboy Layer
Web server
Plug Layer
Composable modules to build web applications
Ecto Layer
Database Abstraction Layer (DBAL / DAL)

Request lifecycle



Websocket communication (soft-realtime)



Libraries explanation 1/3

Endpoint

- start and end of the request lifecycle
- all aspects of requests up until the router takes over
- core set of plugs to apply to all requests

Router

- parses and dispatches requests to the correct controller/action
- helpers to generate route paths or urls to resources
- pipelines - applies groups of plugs to a set of routes

Controller

- provide functions, called actions, to handle requests
- action: prepare data and pass it into views
- action: invoke rendering via views
- action: perform redirects

Libraries explanation 2/3

Views - presentation layer

- render templates
- define template helper functions to decorate data

Templates

- files containing the contents that will be served in a response
- basic response structure, allow dynamic data to be inserted
- precompiled and fast

Libraries explanation 3/3

Channels

- manage sockets for easy realtime communication
- analogous to controllers, but allow bi-directional communication with persistent connections

PubSub

- underlies the channel layer and allows clients to subscribe to topics

There's already a great guide for this

[LINK]

Ecto - the concepts

- Repo module - Via the repository, we can create, update, destroy and query existing entries.
- Schemas - used to map data sources to Elixir structs.
- Changesets - way to filter and cast external parameters, as well to validate changes before applying them
- Query - queries written in Elixir syntax with specific DSL. Queries are by default secure, avoiding common problems. These can be created composable / piece by piece

Ecto SQL - not the same thing!

- This provides functionality for working with SQL databases in Ecto
- Migrations are an example of this

There's already a great guide for this

- Database is up to you (MySQL might be the easiest)
- Feel free to use generators such as:

```
mix phx.gen.schema User users \
name:string email:string \
bio:string number_of_pets:integer
```

[LINK]