Curos Elixir

Actividad 6

Mix new

```
$ mix new app --sup

* creating README.md
* creating .formatter.exs
* creating .gitignore
* creating mix.exs
* creating lib
* creating lib/app.ex
* creating lib/app/application.ex
* creating test
* creating test/test_helper.exs
* creating test/app_test.exs

Your Mix project was created successfully.
You can use "mix" to compile it, test it, and more:

    cd app
    mix test

Run "mix help" for more commands.
```

Mix deps

Edita la función deps en el archivo `app/mix.exs`

```
    defp deps do
    [
      {:ecto_sql, "~> 3.2"},
      {:myxql, ">= 0.0.0"}
    ]
    end
```

```
$ mix deps.get
Resolving Hex dependencies...
Dependency resolution completed:
New:
  connection 1.0.4
  db_connection 2.2.2
  decimal 1.8.1
  ecto 3.4.4
  ecto_sql 3.4.3
  myxql 0.4.0
  telemetry 0.4.1
* Getting ecto_sql (Hex package)
* Getting myxql (Hex package)
* Getting db_connection (Hex package)
* Getting decimal (Hex package)
* Getting connection (Hex package)
* Getting ecto (Hex package)
* Getting telemetry (Hex package)
```

Base de datos

- Crear base de datos:

```
> create database f1db;
```

- Cargar la base de datos:

```
$ mysql -u user -p f1db < DataBase/f1d.sql
```

Repo

- Creamos el repositorio para poder comunicarnos con la base de datos

```
$ mix ecto.gen.repo -r App.Repo

* creating lib/app
* creating lib/app/repo.ex
* creating config/config.exs
Don't forget to add your new repo to your supervision tree
(typically in lib/app/application.ex):

    {App.Repo, []}

And to add it to the list of ecto repositories in your
configuration files (so Ecto tasks work as expected):

    config :app,
      ecto_repos: [App.Repo]
```

https://hexdocs.pm/ecto/Mix.Tasks.Ecto.Gen.Repo.html#module-command-line-options

- Modificamos el conector de Repo `lib/app/repo.ex` por defaul repo tiene el conector de Postgres

```
defmodule App.Repo do
  use Ecto.Repo,
    otp_app: :app,
    adapter: Ecto.Adapters.MyXQL
end
```

- Configuramos el módulo App.Repo como supervisor dentro del árbol de supervisor de nuestra aplicación en `lib/app/application.ex`. Esto iniciará el proceso de Ecto cuando nuestra aplicación inicie.

```
...

    def start(_type, _args) do
    # List all child processes to be supervised
    children = [
      App.Repo,
    ]

  ...
```

- Modificamos las credenciales de acceso a la base de datos `config/config.exs`

```
  config :app, App.Repo,
  database: "f1db",
```

```
    username: "root",
    password: "",
    hostname: "localhost"
```

Esquemas

- Crea una carpeta `lib/races` y dentro de un archivo `race.ex`
- Creamos el módulo race.ex

```
defmodule Races.Race do
  use Ecto.Schema

  @primary_key {:raceId, :id, autogenerate: true}

  schema "races" do
    field :year, :integer
    field :round, :integer
    field :circuitId, :integer
    field :name, :string
    field :date, :date
    field :time, :time
    field :url, :string
  end
end
```

Probar persistencia

Dentro del proyecto

```
$ iex -S mix

iex> alias App.Repo
App.Repo
iex> alias Races.Race
Races.Race

iex> Repo.get!(Race, 1)

17:53:26.528 [debug] QUERY OK source="races" db=0.3ms decode=0.9ms queue=2.2ms idle=163.4ms
SELECT r0.`raceId`, r0.`year`, r0.`round`, r0.`circuitId`, r0.`name`, r0.`date`, r0.`time`, r0.`url` FROM `ra
%Races.Race{
  __meta__: #Ecto.Schema.Metadata<:loaded, "races">,
  circuitId: 1,
  date: ~D[2009-03-29],
  name: "Australian Grand Prix",
  raceId: 1,
  round: 1,
  time: ~T[06:00:00],
  url: "http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix",
  year: 2009
}

iex> Repo.get(Race, 0)

17:54:09.984 [debug] QUERY OK source="races" db=1.3ms queue=0.1ms idle=1625.9ms
SELECT r0.`raceId`, r0.`year`, r0.`round`, r0.`circuitId`, r0.`name`, r0.`date`, r0.`time`, r0.`url` FROM `ra
nil
```

https://hexdocs.pm/ecto/Ecto.Repo.html#content