Curso Elixir

Sección 14

Uso Changeset

Changesets nos permite filtrar, convertir, validar y definir restricciones al manipular estructuras. (https://hexdocs.pm/ecto/Ecto.Changeset.html).

Dentro del módulo lib/races/race.ex, agregamos `import Ecto.Changeset` y el siguiente función:

```
  def changeset(race, attrs) do
    race
    |> cast(attrs, [:year, :round, :circuitId, :name, :date, :time, :url])
    |> validate_required([:year, :round, :circuitId, :name, :date])
  end
```

En la función anterior definimos los campos que pueden ser mapeados así como la posibilidad de realizar validaciones (validate_required).

Recompilamos el código y verificamos que todo este bien (IEx.Helpers.recompile).

CRUD

Ahora agregaremos la funcionalidad de CRUD a el módulo race. Dentro de lib/races/race.ex agregamos:

```
import Ecto.Query, warn: false
alias App.Repo
```

y las funciones

```
  def get_race(id) do
    Repo.get!(Races.Race, id)
  end

  def create_race(attrs \\ %{}) do
    %Races.Race{}
    |> changeset(attrs)
    |> Repo.insert()
  end

  def update_race(%Races.Race{} = race, attrs) do
    race
    |> changeset(attrs)
    |> Repo.update()
  end
```

```
  def delete_race(%Races.Race{} = race) do
    Repo.delete(race)
  end
```

Recompilamos el código y verificamos que todo este bien (IEx.Helpers.recompile).

Pruebas

Cuando generamos un proyecto en Elixir con el comando mix, automaticamente se crea un acarpeta test. Para correr las pruebas ejecutamos el comando mix test dentro de la aplicación

```
cd App
mix test

..

Finished in 0.3 seconds
1 doctest, 1 test, 0 failures
```

El framework de pruebas que viene con Elixir es ExUnit e incluye todo lo que necesitamos para hacer pruebas a fondo de nuestro código.

```
defmodule AppTest do
  use ExUnit.Case
  doctest App

  test "greets the world" do
    assert App.hello() == :world
  end
end
```

- Assert

Usamos el macro assert para probar que la expresión es verdadera. En el caso que no lo sea, un error será lanzado y nuestras pruebas fallarán

- Refute

El macro refute es el contrario de assert probando que la expresión sea falsa.

Para más información sobre assets https://hexdocs.pm/ex_unit/master/ExUnit.Assertions.html#content

Dentro de la carpeta test, agregamos un nuevo archivo `races_test.exs` y agregamos algunas pruebas.

```
defmodule RacesTest do
  use ExUnit.Case
```

```
  alias Races.Race

  test "create_race" do
    params = %{year: 2020, round: 52, circuitId: 1, name: "Race Mexico", date:
"2020-05-20"}
    {:ok, race}= Race.create_race(params)

    assert race.raceId > 0
  end

  ...

end
```

DocTest

Este macro nos permite generar pruebas a partir de los ejemplos de código en la documentación de un módulo, función ó macro.

Agreguemos algunos doctest a las funciones de CRUD del módulo races

```
@doc """
  get_race.

  ## Examples

      iex> alias Races.Race
      iex> race = Race.get_race(1000)
      iex> race.name
      "Hungarian Grand Prix"

  """
```

y agregamos la línea `doctest Race` en races_test.exs, y corremos de nuevo las pruebas `mix test`