Bet (F)unFair

Betting is only a problem if you stop before winning big

featuring friend of the group mr ‘acc’

Álvaro Ferrero Díez
Borja Martinena Cepa


Phoenix
Framework
TM

`handle_call` is just a function

and other bad code decisions

```
defp place_bet(kind, user_id, market_id, stake, odds, exchange_name) do
    %{status: market_status} = Repo.get(Models.Market, market_id)
    case market_status do
      :active ->


        case Repo.get_user(user_id, exchange_name) do
          {:ok, %{id: id}} ->
            case Repo.add_bet(%Models.Bet{
                   bet_type: kind,
                   user: id,
                   market: market_id,
                   original_stake: stake,
                   odds: odds,
                   status: :active,
                   remaining_stake: stake
                 }) do
              {:ok, %Models.Bet{id: id}} ->
                # Please don't do this at home
                handle_call({:user_withdraw, user_id, stake}, __MODULE__, exchange_name)

                {:reply, {:ok, id}, exchange_name}

              {:error, error} ->
                {:reply, error, exchange_name}
            end

          error ->
            {:reply, error, exchange_name}
        end
      _ -> {:reply, {:error, "Market not active"}, exchange_name}
    end
  end
```

```elixir
defp settle_market_bet(bet, {success, count, result}) do
    setup( ... )

    acc =
      cond do
        {bet.bet_type, result} == {:lay, false} or {bet.bet_type, result} == {:back, true} ->
          do_stuff( ... )

        true ->
          do_other_stuff( ... )
      end

    update_bet_status()
    acc
  end
```

Final project structure

```
lib
├── bet_unfair
│   ├── application.ex
│   ├── models.ex
│   ├── repo.ex
│   └── server.ex
└── bet_unfair.ex
```

What we learned
(but didn't do)

- Defining and limiting the basic operations that you want to perform on your database helps.
- Understanding the algorithm you are trying to implement before actually implementing it helps too.
- Leveraging the power of reusable code is better than simply writing new code for every new bit of functionality.

Q & A