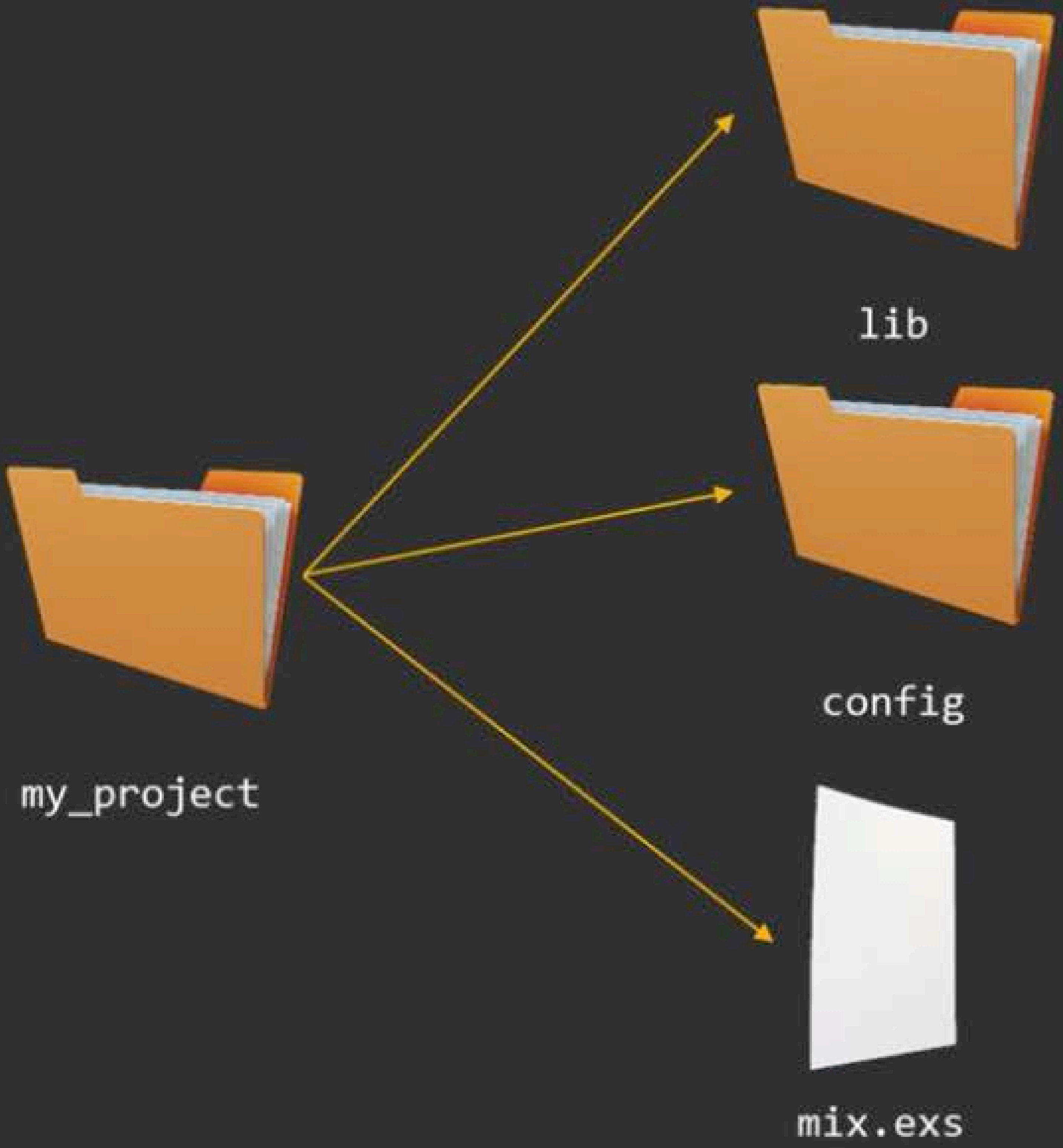
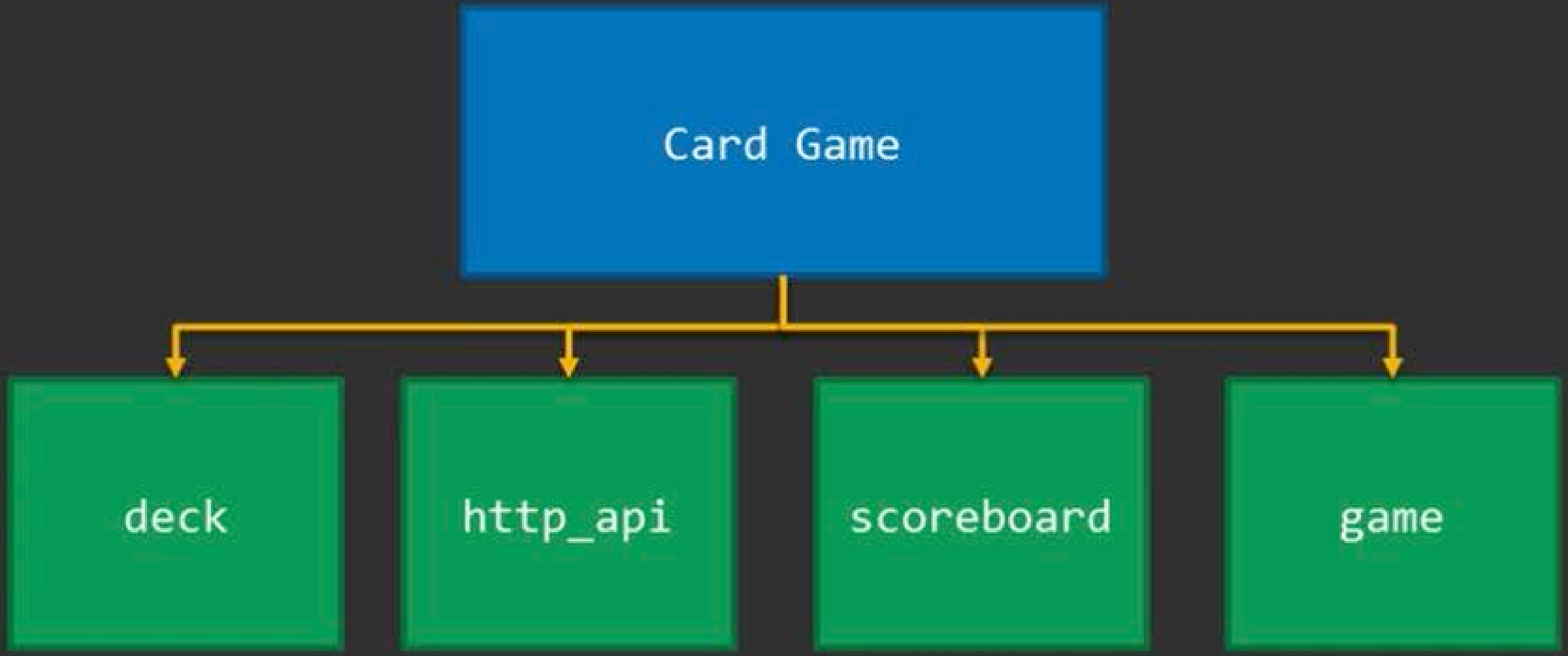
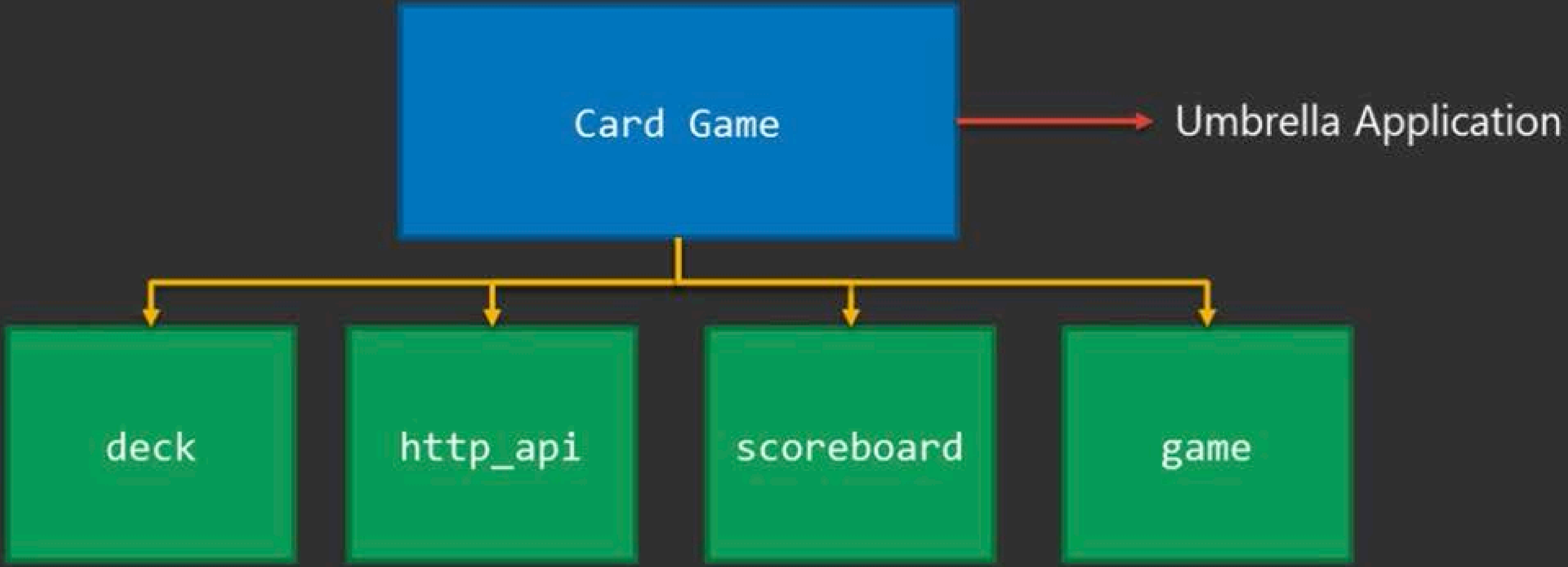
Umbrella Applications

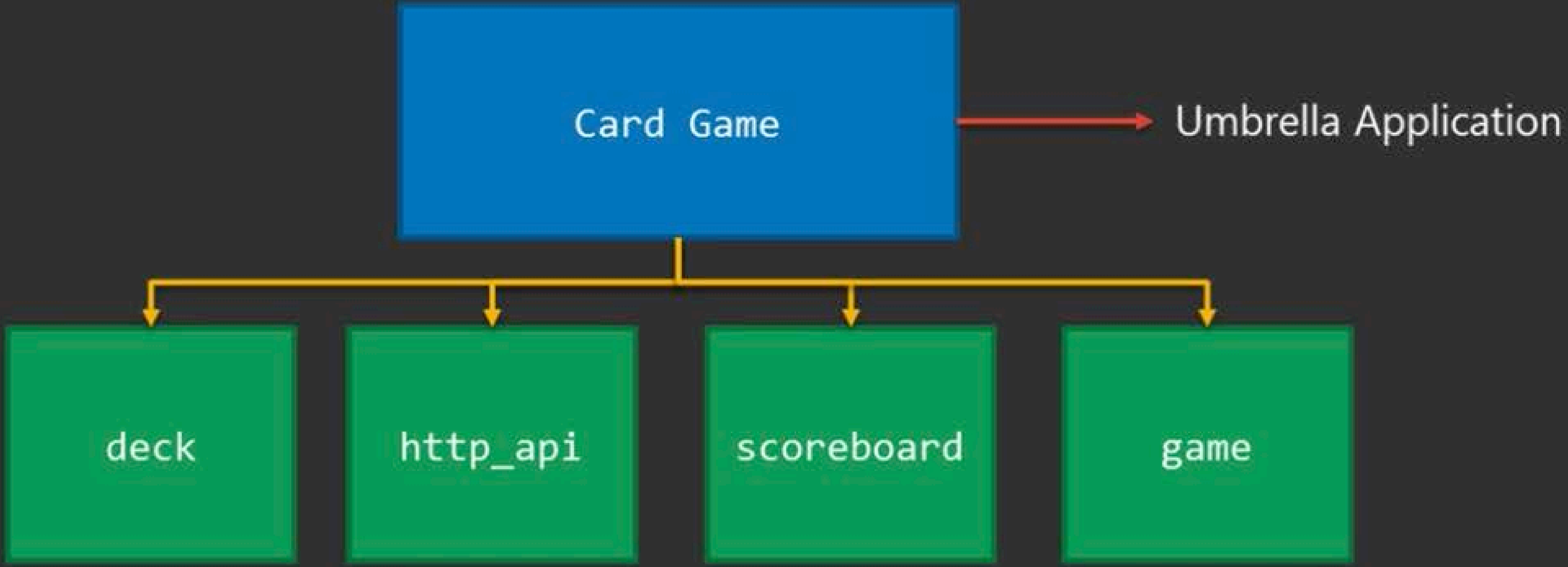
In this Video, we are going to take a look at…

- What is an Umbrella application
- Creating umbrella applications
- Pitfalls

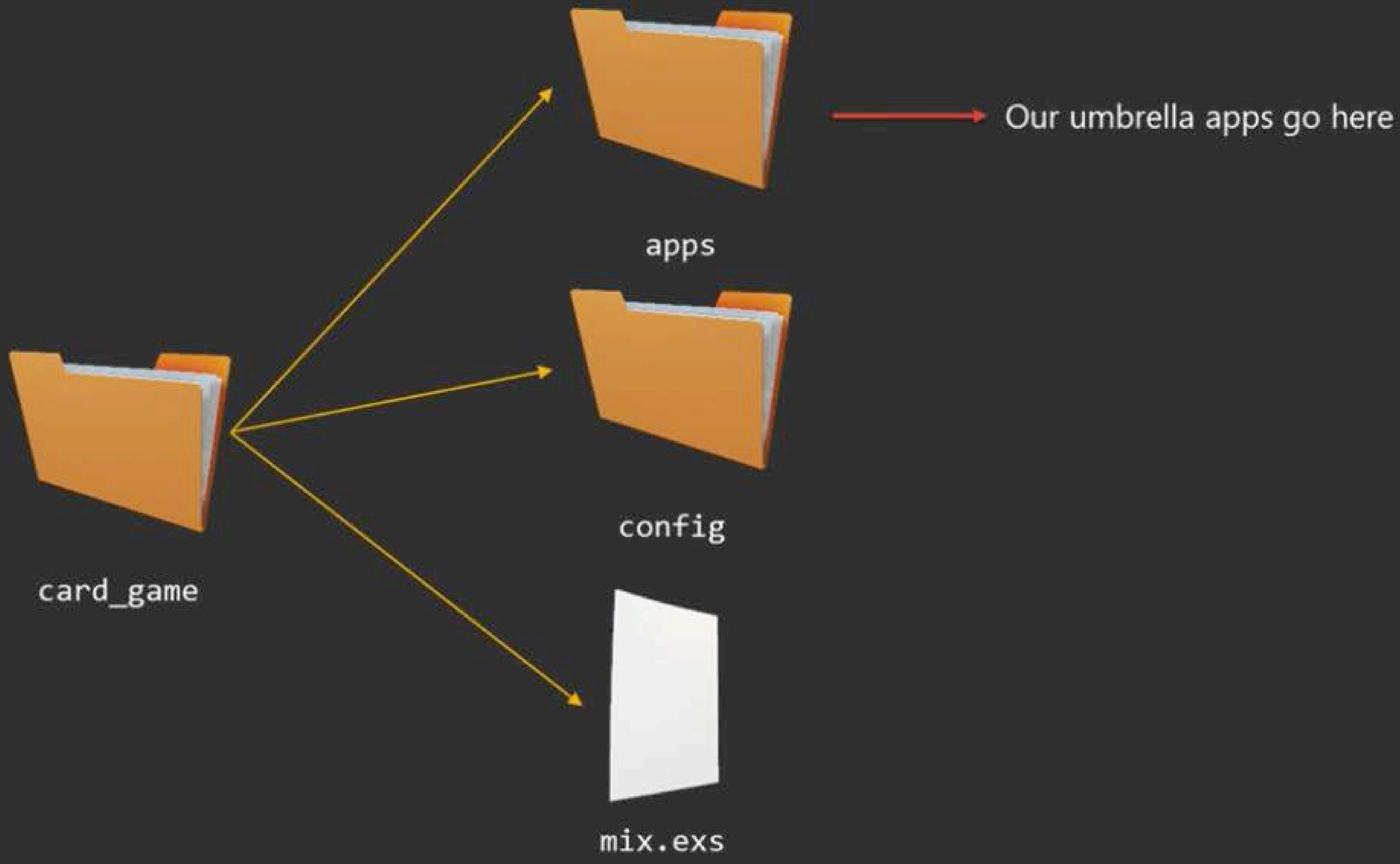

lib
config
my_project
mix.exs

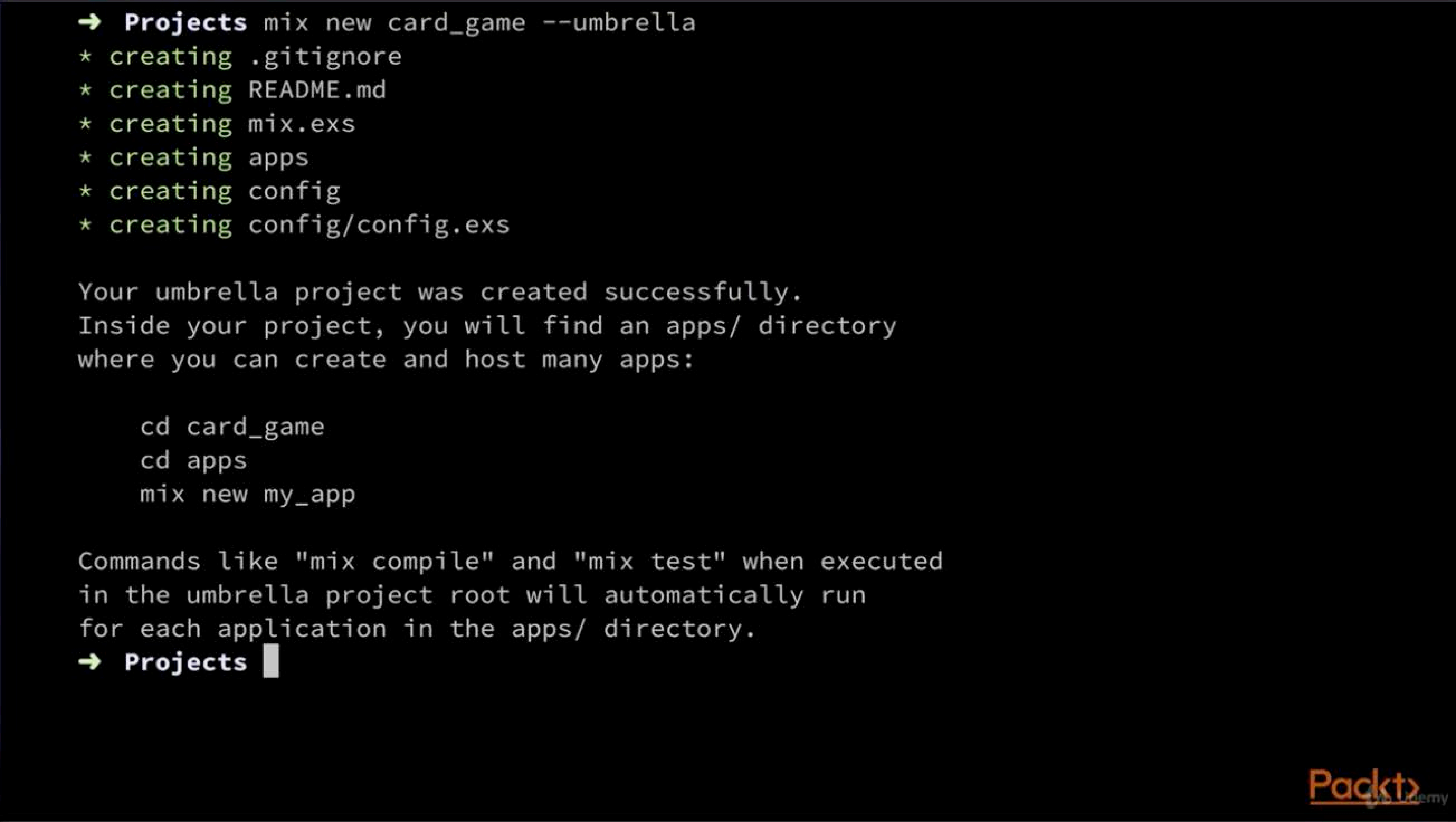

Card Game
deck
http_api
scoreboard
game


Card Game
Umbrella Application
deck
http_api
scoreboard
game

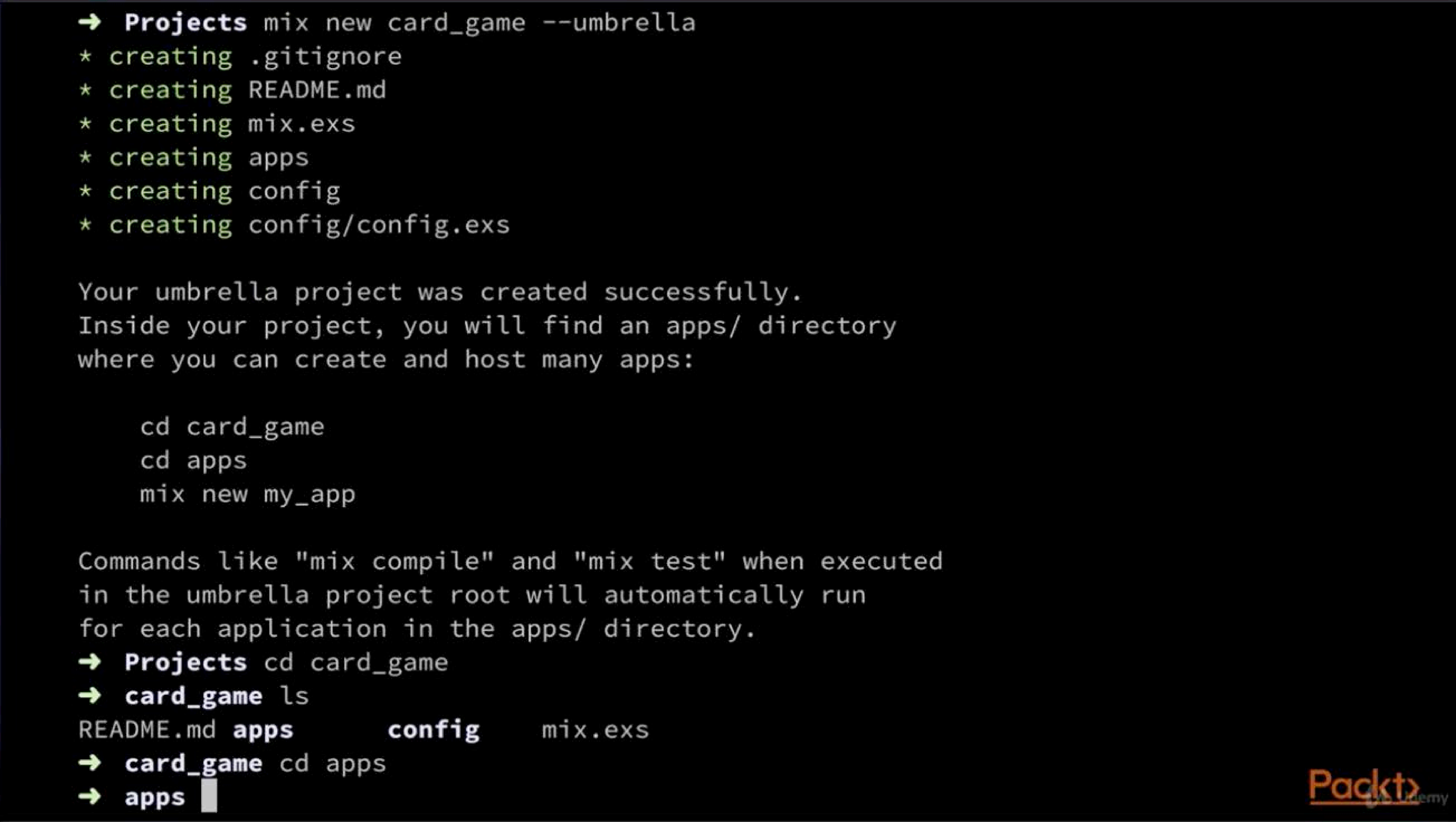


```
mix new card_game --umbrella
```

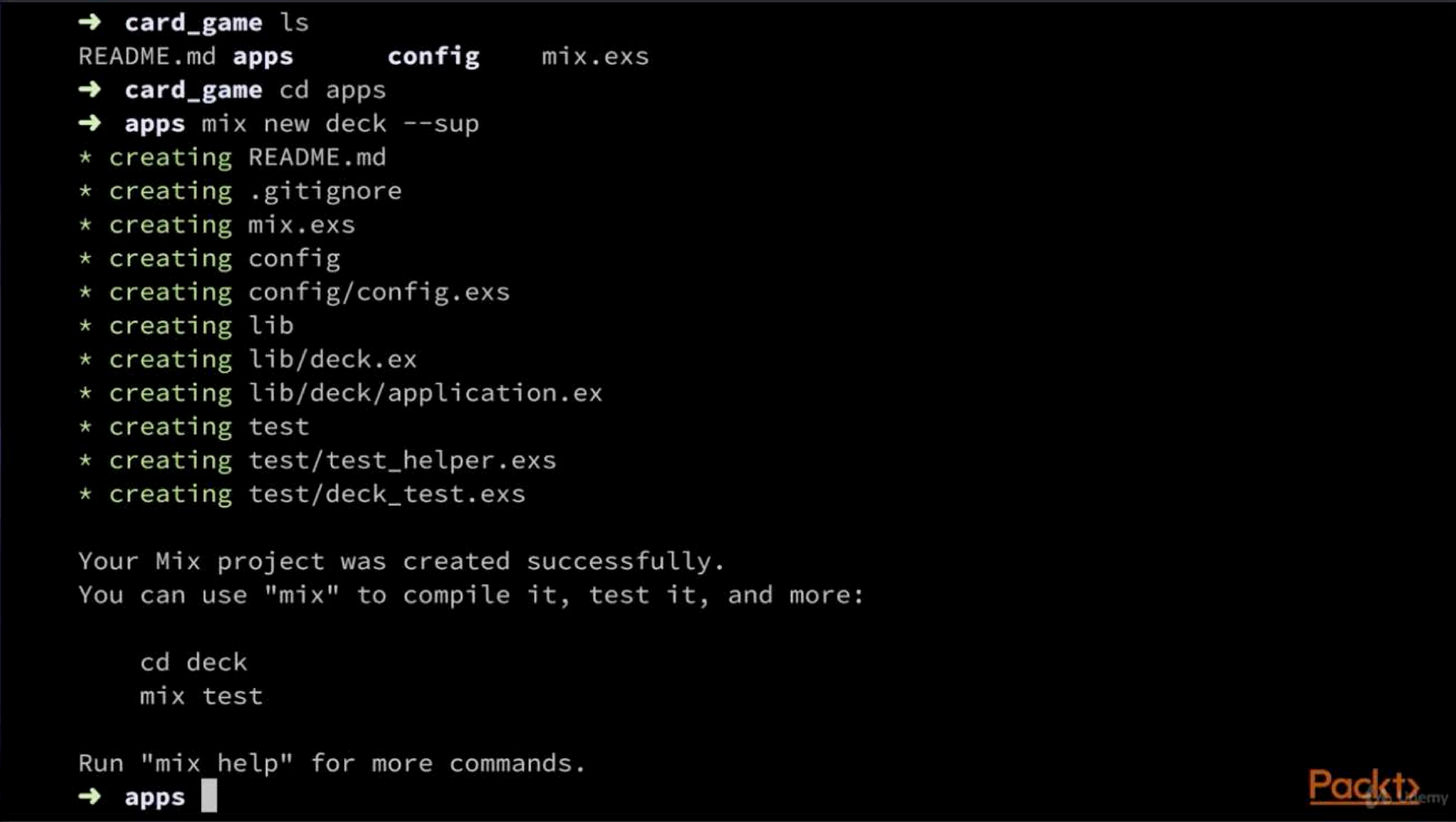

Our umbrella apps go here
apps
config
card_game
mix.exs



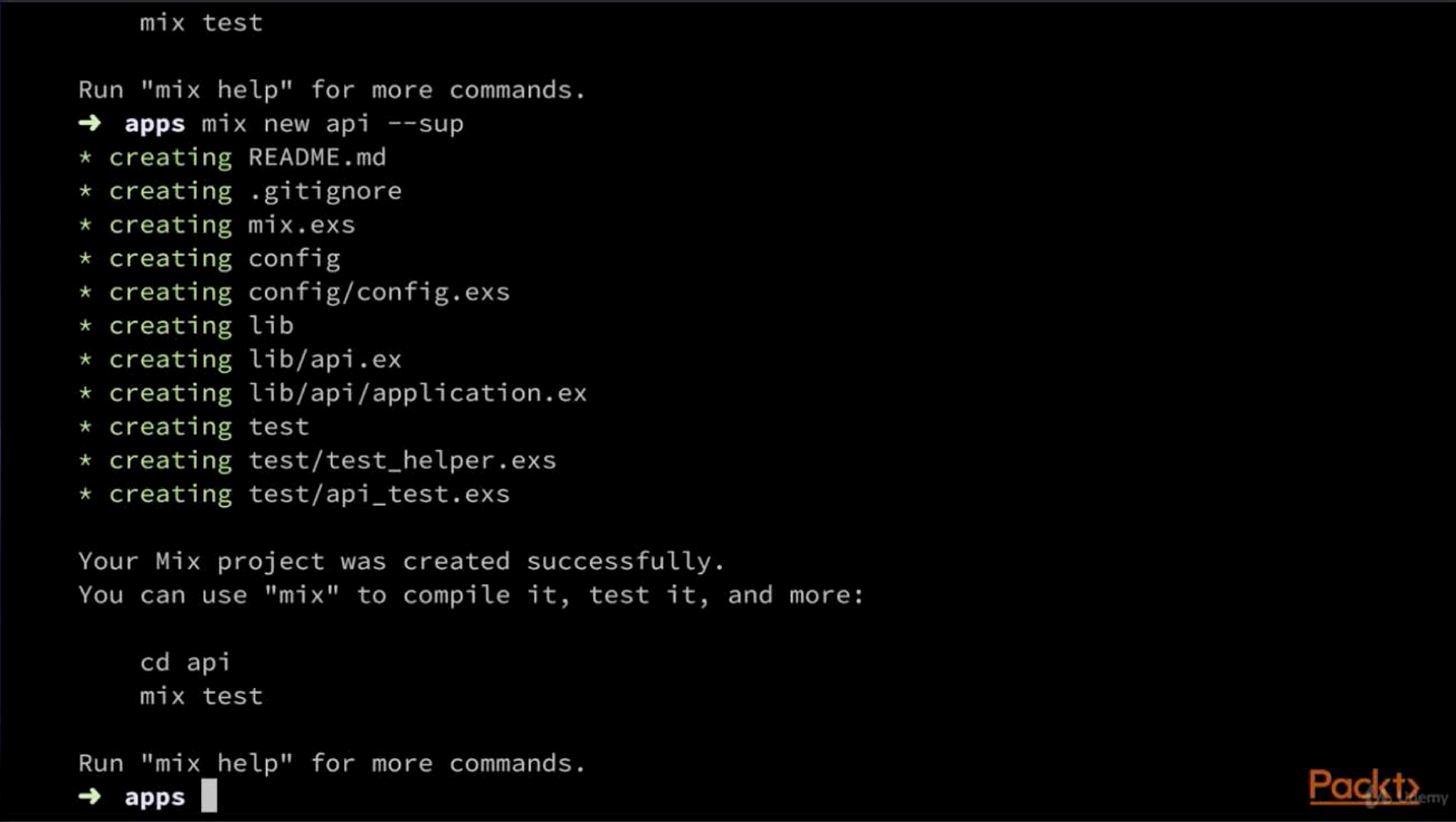
```
➜  Projects mix new card_game --umbrella
* creating .gitignore
* creating README.md
* creating mix.exs
* creating apps
* creating config
* creating config/config.exs

Your umbrella project was created successfully.
Inside your project, you will find an apps/ directory
where you can create and host many apps:

    cd card_game
    cd apps
    mix new my_app

Commands like "mix compile" and "mix test" when executed
in the umbrella project root will automatically run
for each application in the apps/ directory.
➜  Projects █
```



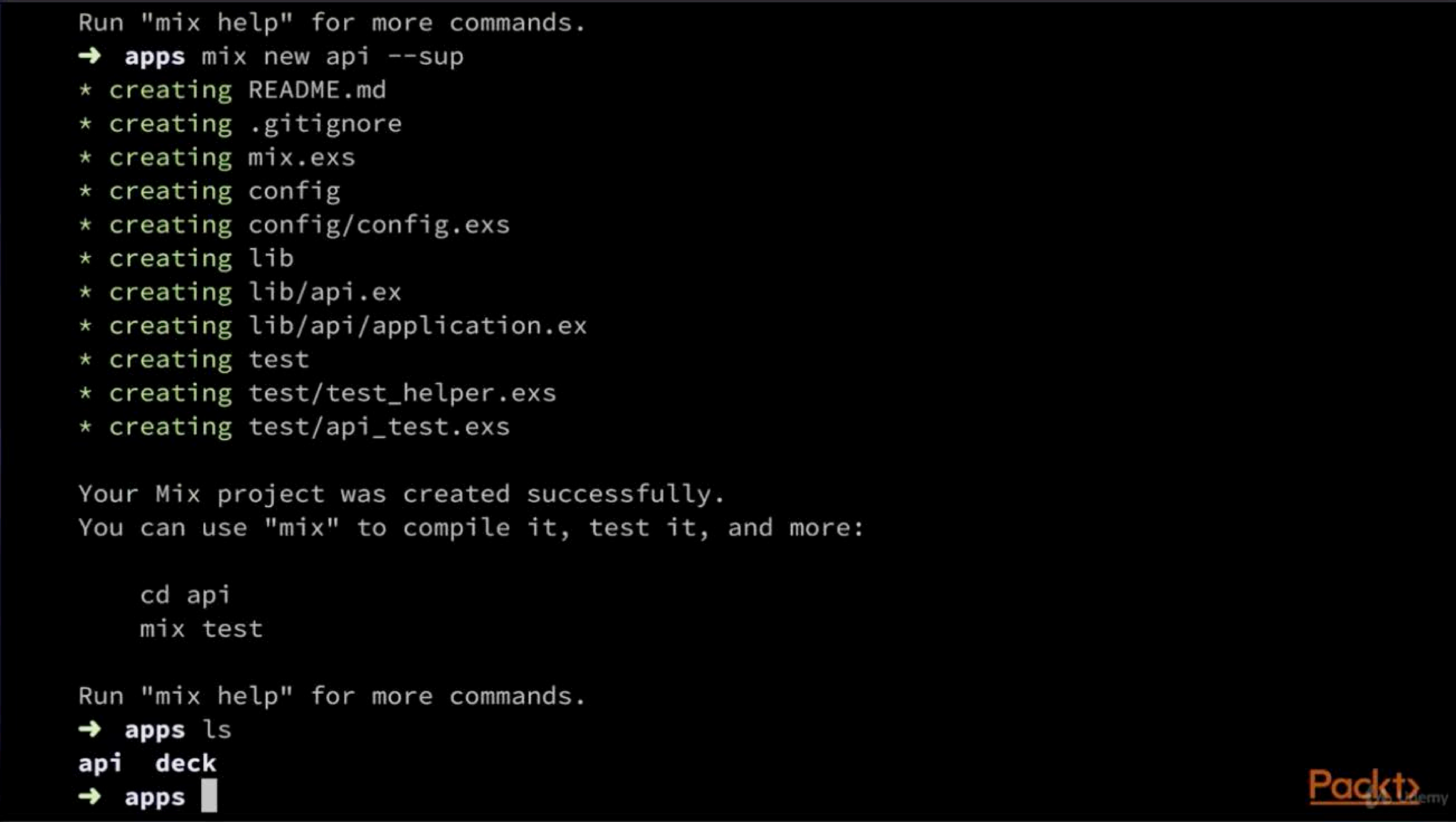
```
➜  Projects mix new card_game --umbrella
* creating .gitignore
* creating README.md
* creating mix.exs
* creating apps
* creating config
* creating config/config.exs

Your umbrella project was created successfully.
Inside your project, you will find an apps/ directory
where you can create and host many apps:

    cd card_game
    cd apps
    mix new my_app

Commands like "mix compile" and "mix test" when executed
in the umbrella project root will automatically run
for each application in the apps/ directory.
➜  Projects cd card_game
➜  card_game ls
README.md apps       config    mix.exs
➜  card_game cd apps
➜  apps
```



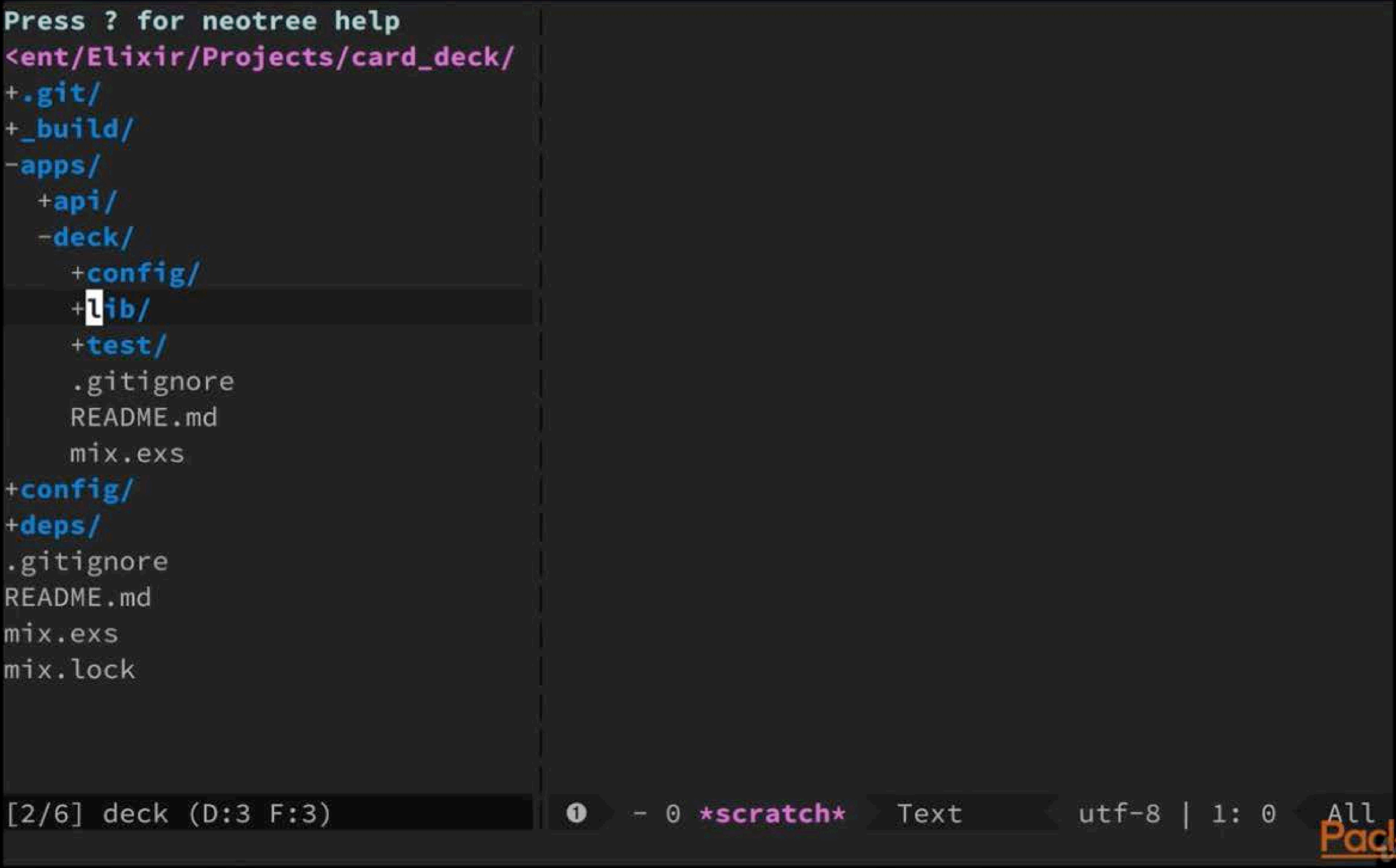
```
➜  card_game ls
README.md apps      config    mix.exs
➜  card_game cd apps
➜  apps mix new deck --sup
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/deck.ex
* creating lib/deck/application.ex
* creating test
* creating test/test_helper.exs
* creating test/deck_test.exs

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

    cd deck
    mix test

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



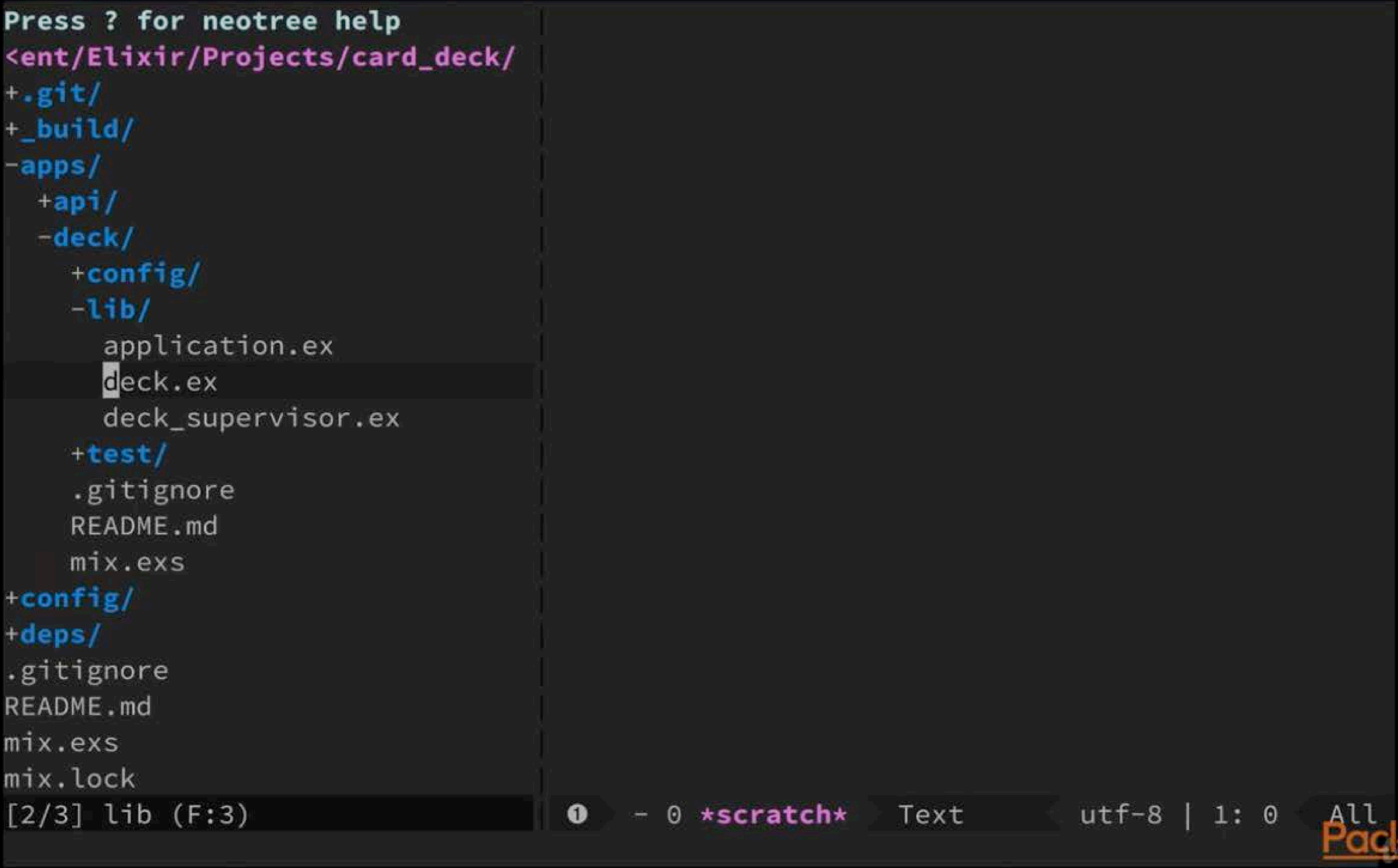
```
    mix test

Run "mix help" for more commands.
➜  apps mix new api --sup
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/api.ex
* creating lib/api/application.ex
* creating test
* creating test/test_helper.exs
* creating test/api_test.exs

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

    cd api
    mix test

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



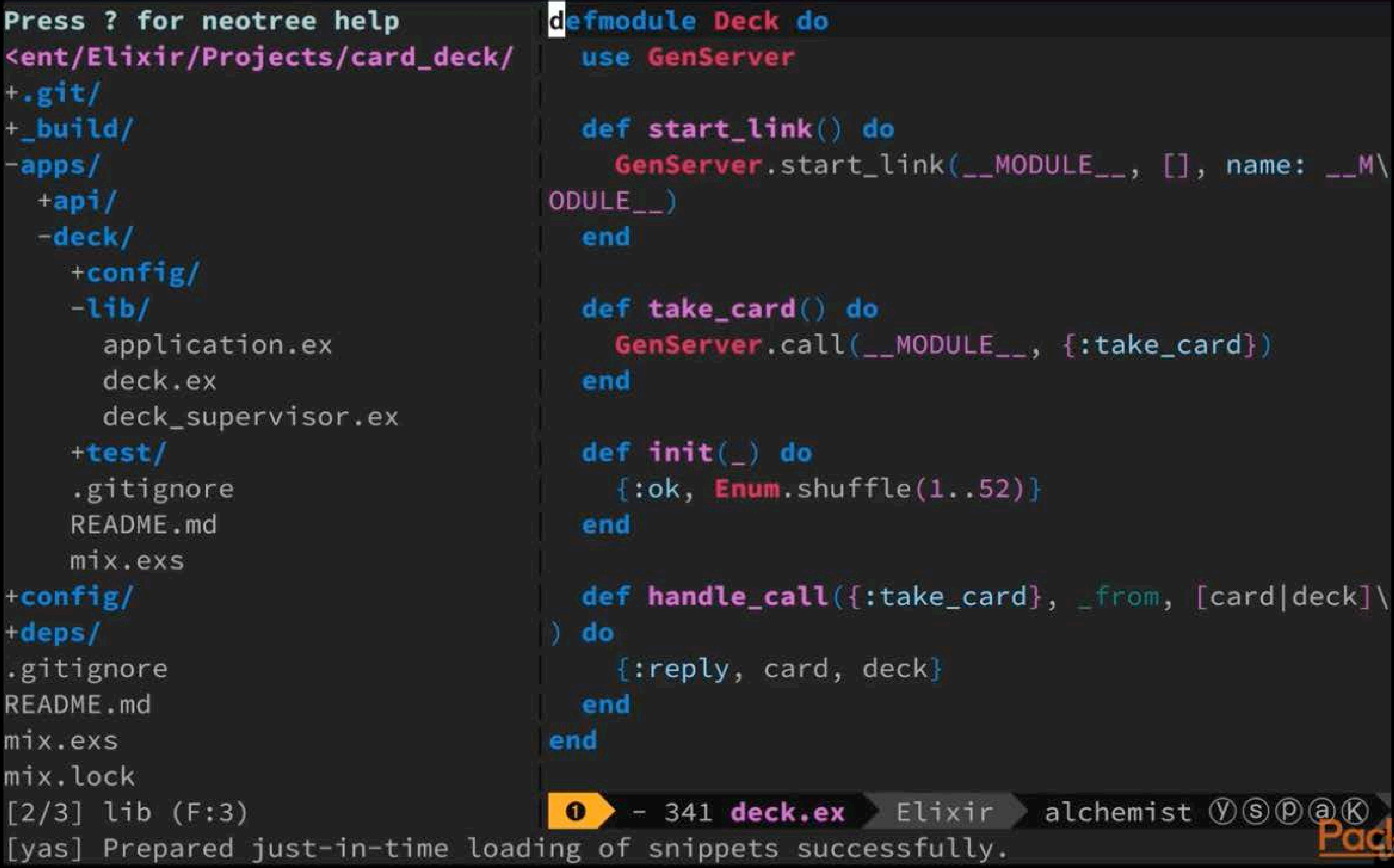
```
Run "mix help" for more commands.
➜  apps mix new api --sup
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/api.ex
* creating lib/api/application.ex
* creating test
* creating test/test_helper.exs
* creating test/api_test.exs

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

    cd api
    mix test

Run "mix help" for more commands.
➜  apps ls
api  deck
➜  apps 
```



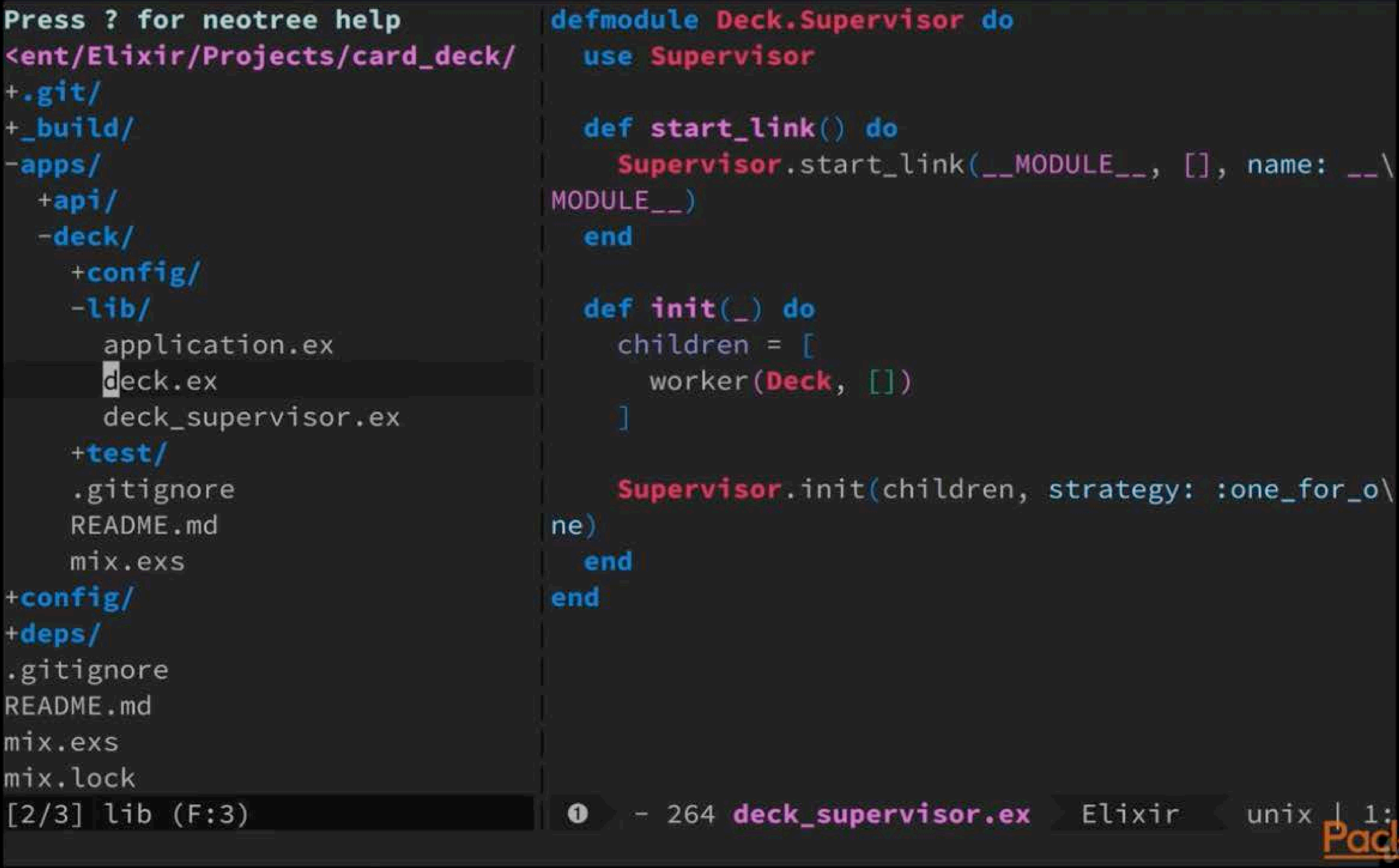
```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  +api/
  -deck/
    +config/
    +lib/
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock

[2/6] deck (D:3 F:3)     ❶  - 0 *scratch*   Text   utf-8 | 1: 0   All
```



```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  +api/
  -deck/
    +config/
    -lib/
      application.ex
      deck.ex
      deck_supervisor.ex
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock
[2/3] lib (F:3)
```



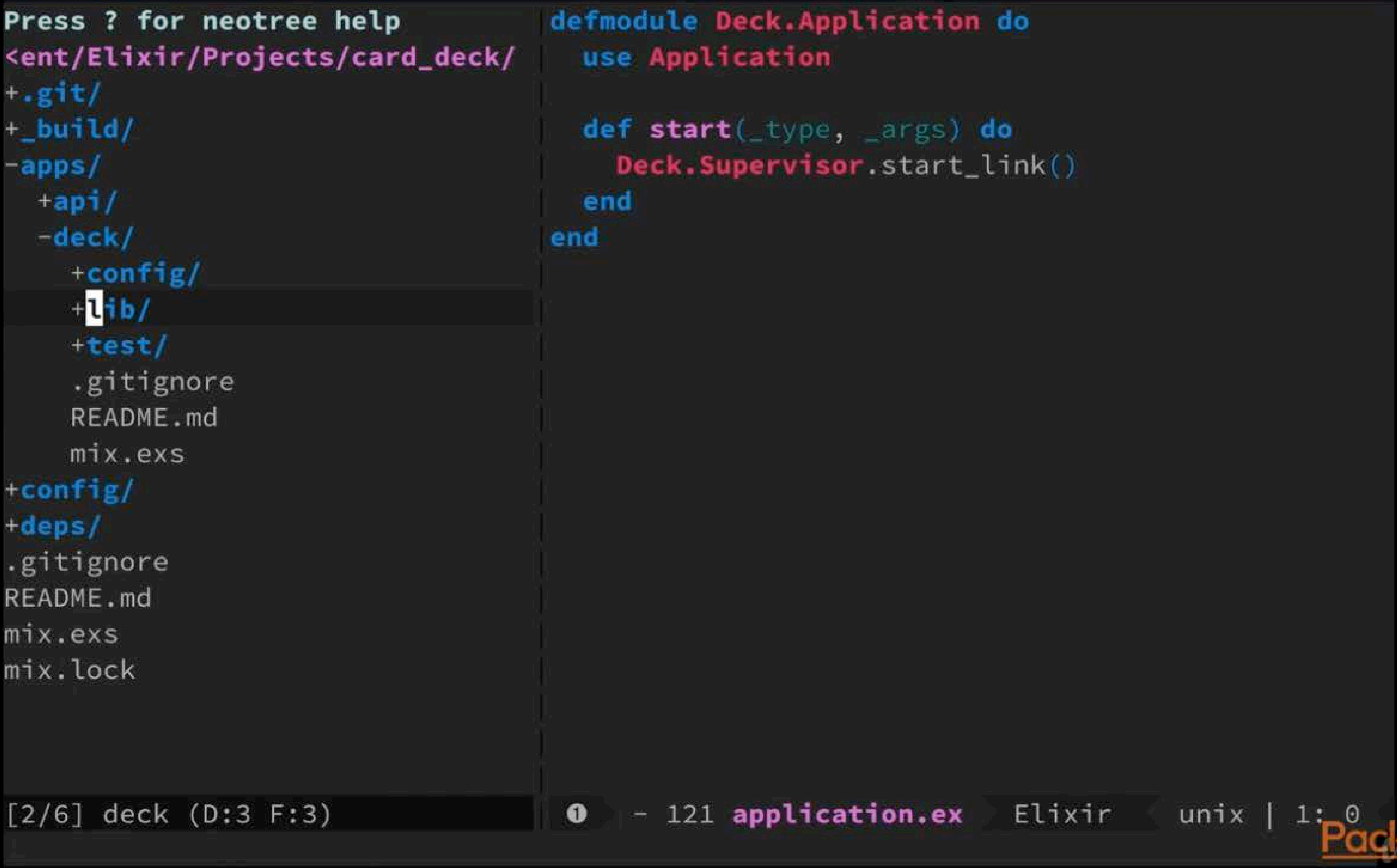
```
Press ? for neotree help                 defmodule Deck do
<ent/Elixir/Projects/card_deck/            use GenServer
+.git/
+_build/                                   def start_link() do
-apps/                                       GenServer.start_link(__MODULE__, [], name: __M\
  +api/                                  ODULE__)
  -deck/                                   end
    +config/
    -lib/                                  def take_card() do
      application.ex                         GenServer.call(__MODULE__, {:take_card})
      deck.ex                              end
      deck_supervisor.ex
    +test/                                 def init(_) do
    .gitignore                               {:ok, Enum.shuffle(1..52)}
    README.md                              end
    mix.exs
+config/                                   def handle_call({:take_card}, _from, [card|deck]\
+deps/                                   ) do
.gitignore                                   {:reply, card, deck}
README.md                                  end
mix.exs                                  end
mix.lock
[2/3] lib (F:3)                          ❶ - 341 deck.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ
[yas] Prepared just-in-time loading of snippets successfully.
```



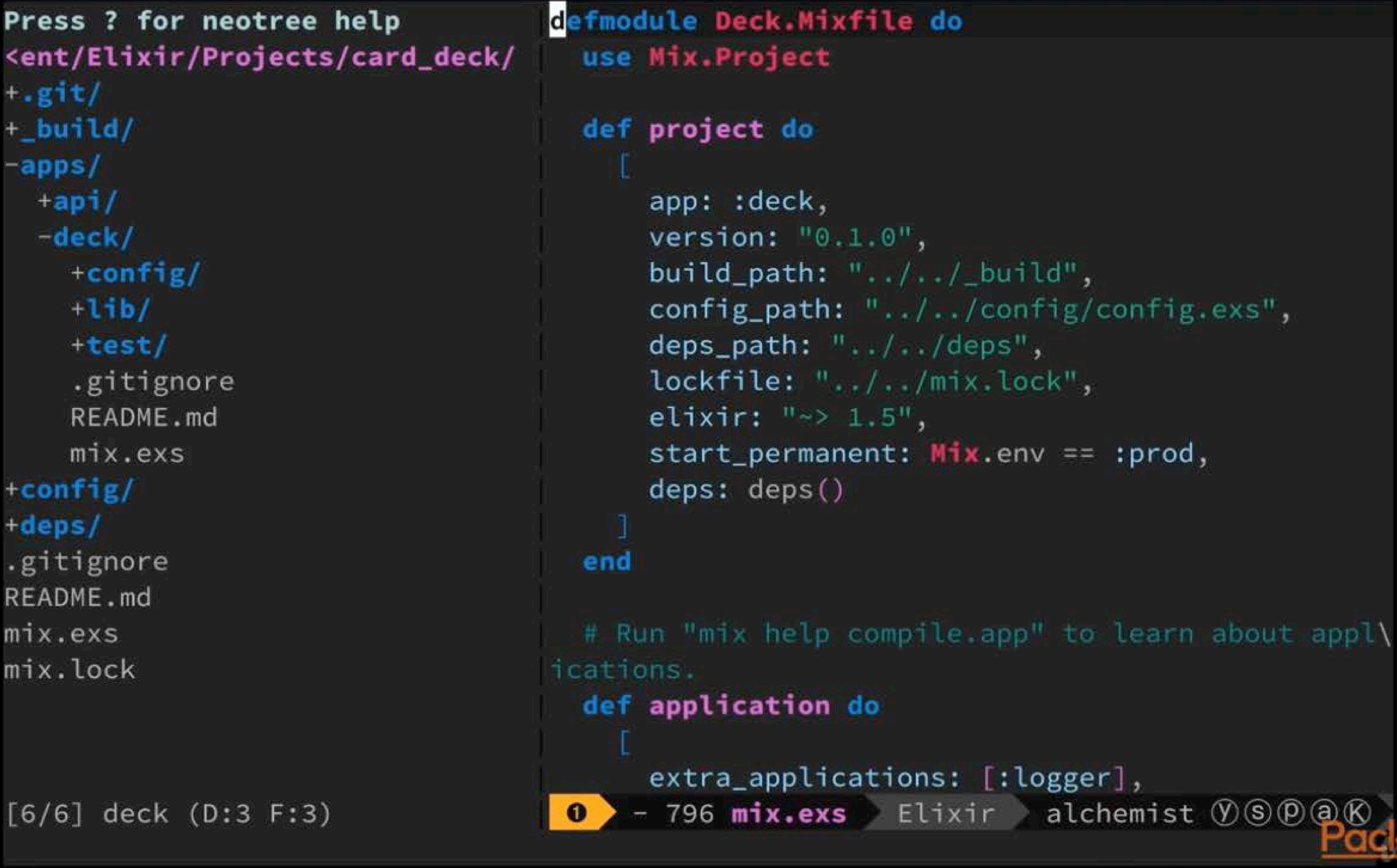
```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  +api/
  -deck/
    +config/
    -lib/
      application.ex
      deck.ex
      deck_supervisor.ex
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock
[2/3] lib (F:3)
```

```
defmodule Deck.Supervisor do
  use Supervisor

  def start_link() do
    Supervisor.start_link(__MODULE__, [], name: __\
MODULE__)
  end

  def init(_) do
    children = [
      worker(Deck, [])
    ]

    Supervisor.init(children, strategy: :one_for_o\
ne)
  end
end
```

```
1 - 264 deck_supervisor.ex   Elixir   unix | 1:
```



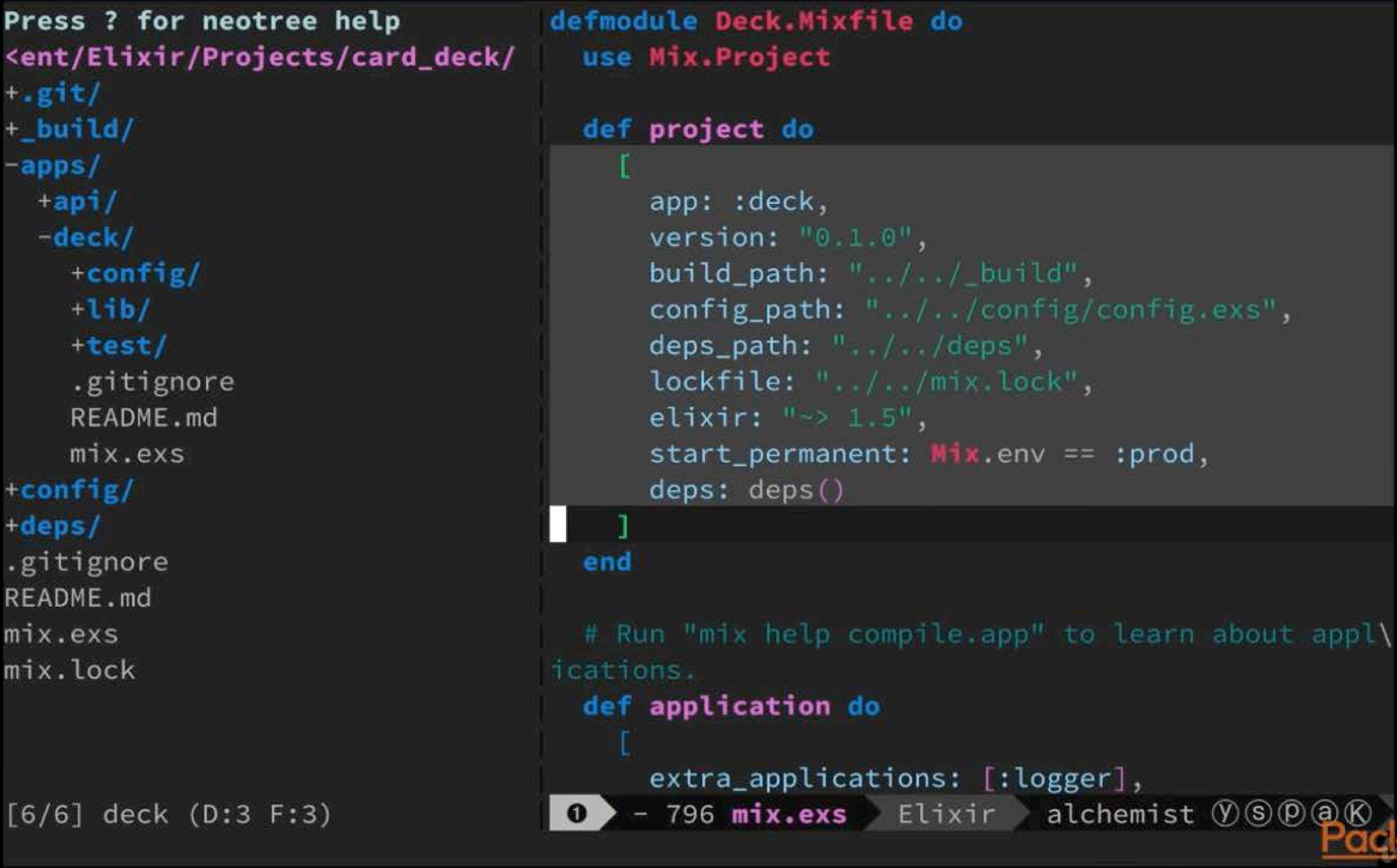
```
Press ? for neotree help              defmodule Deck.Application do
<ent/Elixir/Projects/card_deck/         use Application
+.git/
+_build/                                def start(_type, _args) do
-apps/                                    Deck.Supervisor.start_link()
  +api/                                 end
  -deck/                              end
    +config/
    +lib/
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock
[2/6] deck (D:3 F:3)                  ❶  - 121 application.ex  Elixir  unix | 1: 0
```



```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  +api/
  -deck/
    +config/
    +lib/
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock

[6/6] deck (D:3 F:3)
```

```elixir
defmodule Deck.Mixfile do
  use Mix.Project

  def project do
    [
      app: :deck,
      version: "0.1.0",
      build_path: "../../_build",
      config_path: "../../config/config.exs",
      deps_path: "../../deps",
      lockfile: "../../mix.lock",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about appl\
ications.
  def application do
    [
      extra_applications: [:logger],
```

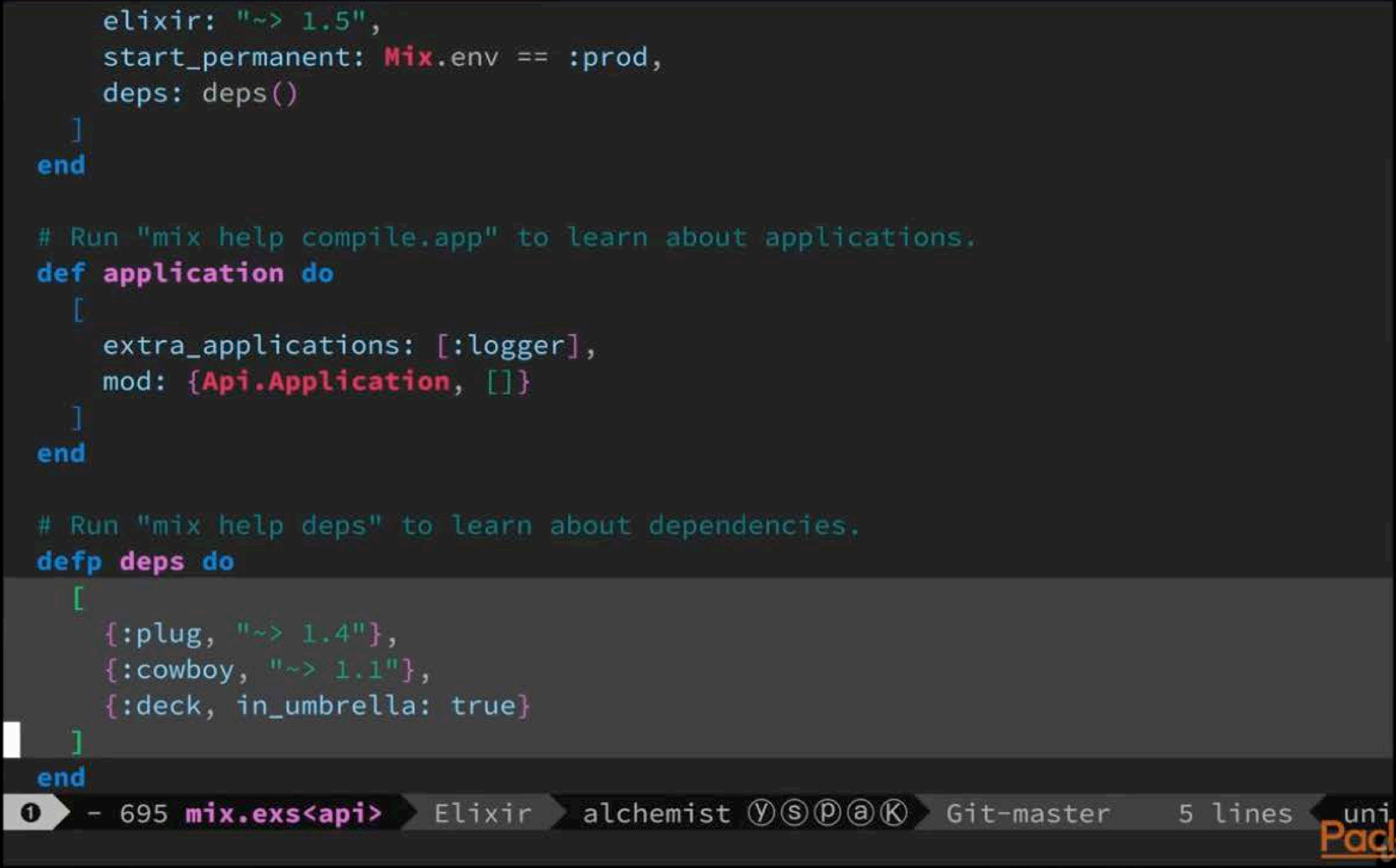
```
❶ - 796 mix.exs  Elixir  alchemist ⓨⓢⓟⓐⓚ
```



```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  +api/
  -deck/
    +config/
    +lib/
    +test/
    .gitignore
    README.md
    mix.exs
+config/
+deps/
.gitignore
README.md
mix.exs
mix.lock

[6/6] deck (D:3 F:3)
```

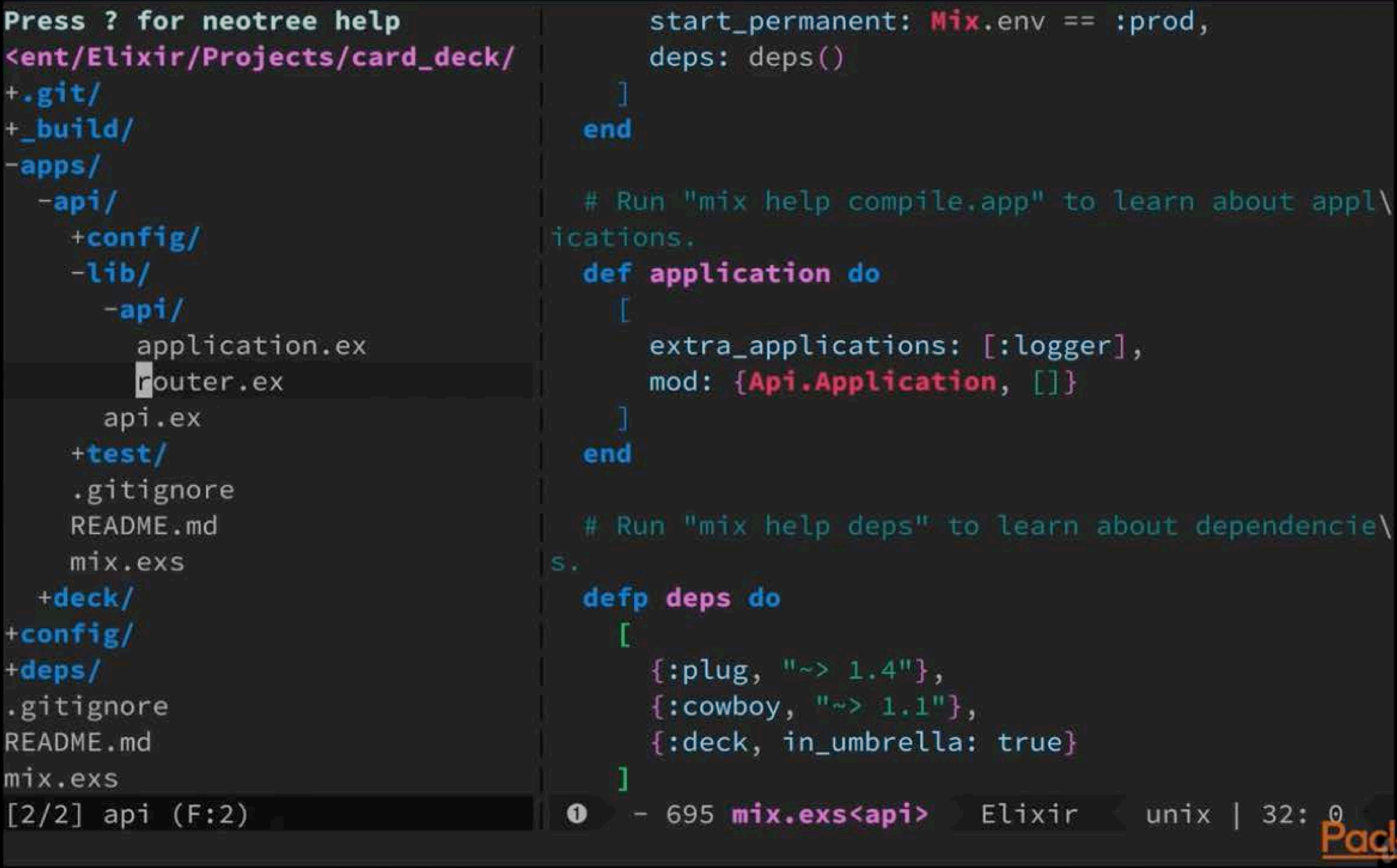
```
defmodule Deck.Mixfile do
  use Mix.Project

  def project do
    [
      app: :deck,
      version: "0.1.0",
      build_path: "../../_build",
      config_path: "../../config/config.exs",
      deps_path: "../../deps",
      lockfile: "../../mix.lock",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about appl\
ications.
  def application do
    [
      extra_applications: [:logger],
```

```
❶ - 796 mix.exs  Elixir  alchemist ⓨⓢⓟⓐⓚ
```



```
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about applications.
  def application do
    [
      extra_applications: [:logger],
      mod: {Api.Application, []}
    ]
  end

  # Run "mix help deps" to learn about dependencies.
  defp deps do
    [
      {:plug, "~> 1.4"},
      {:cowboy, "~> 1.1"},
      {:deck, in_umbrella: true}
    ]
  end
```

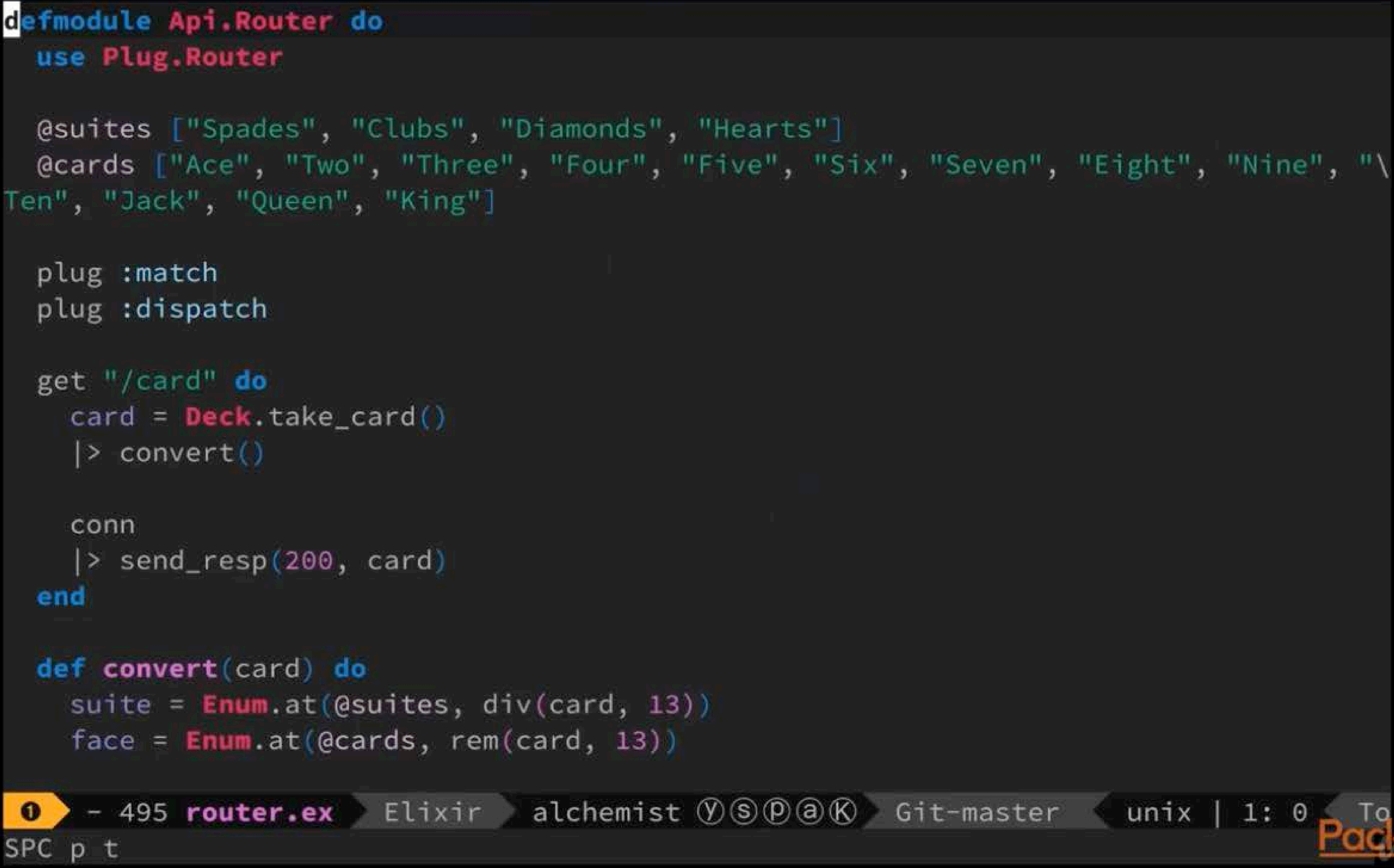
```
❶ - 695 mix.exs<api>  Elixir  alchemist ⓨⓢⓟⓐⓚ  Git-master   5 lines  uni
```



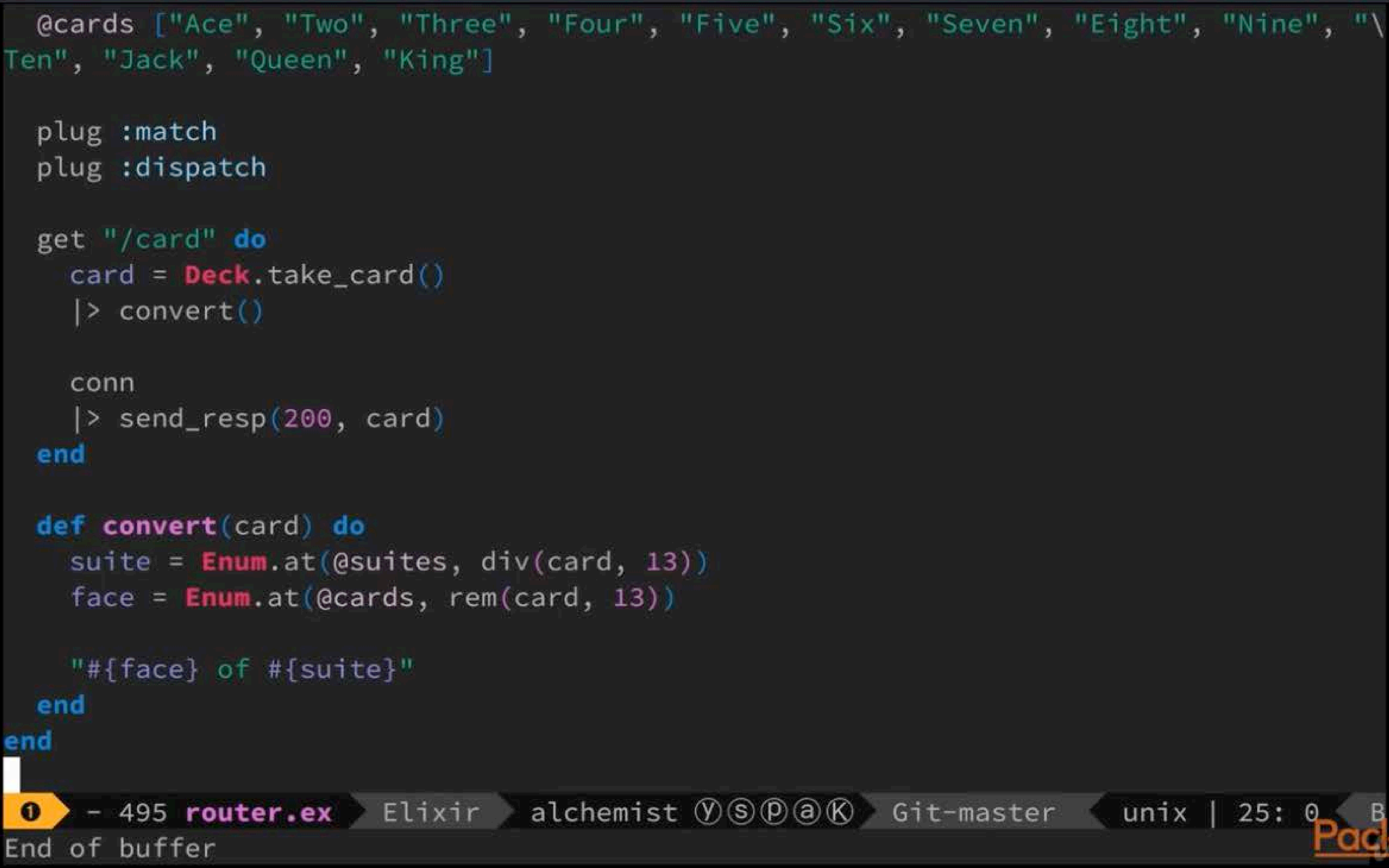
```
Press ? for neotree help
<ent/Elixir/Projects/card_deck/
+.git/
+_build/
-apps/
  -api/
    +config/
    -lib/
      -api/
        application.ex
        router.ex
      api.ex
    +test/
    .gitignore
    README.md
    mix.exs
  +deck/
+config/
+deps/
.gitignore
README.md
mix.exs
[2/2] api (F:2)
```

```
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about appl\
ications.
  def application do
    [
      extra_applications: [:logger],
      mod: {Api.Application, []}
    ]
  end

  # Run "mix help deps" to learn about dependencie\
s.
  defp deps do
    [
      {:plug, "~> 1.4"},
      {:cowboy, "~> 1.1"},
      {:deck, in_umbrella: true}
    ]
```

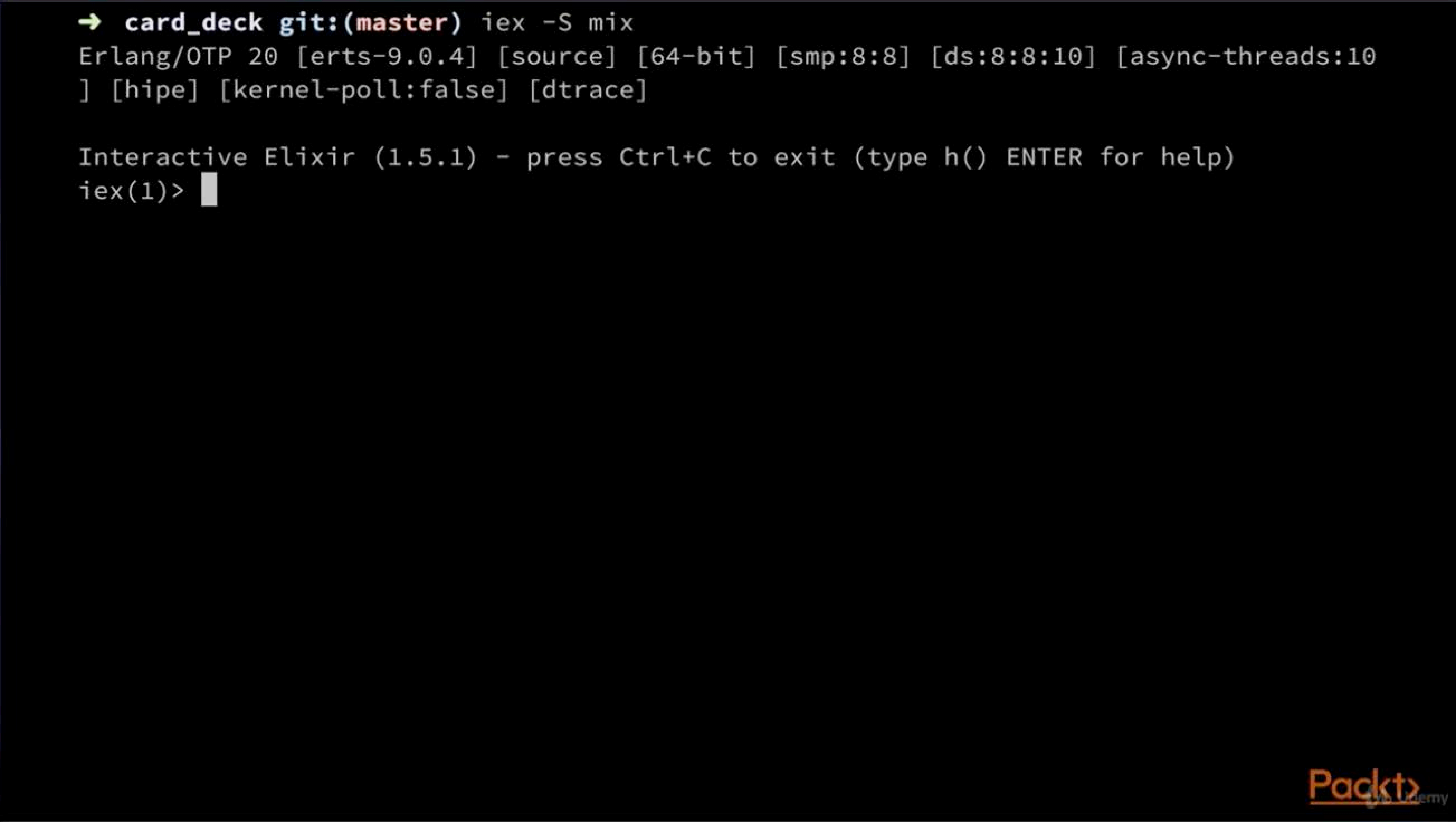
```
1  - 695 mix.exs<api>  Elixir  unix | 32: 0
```



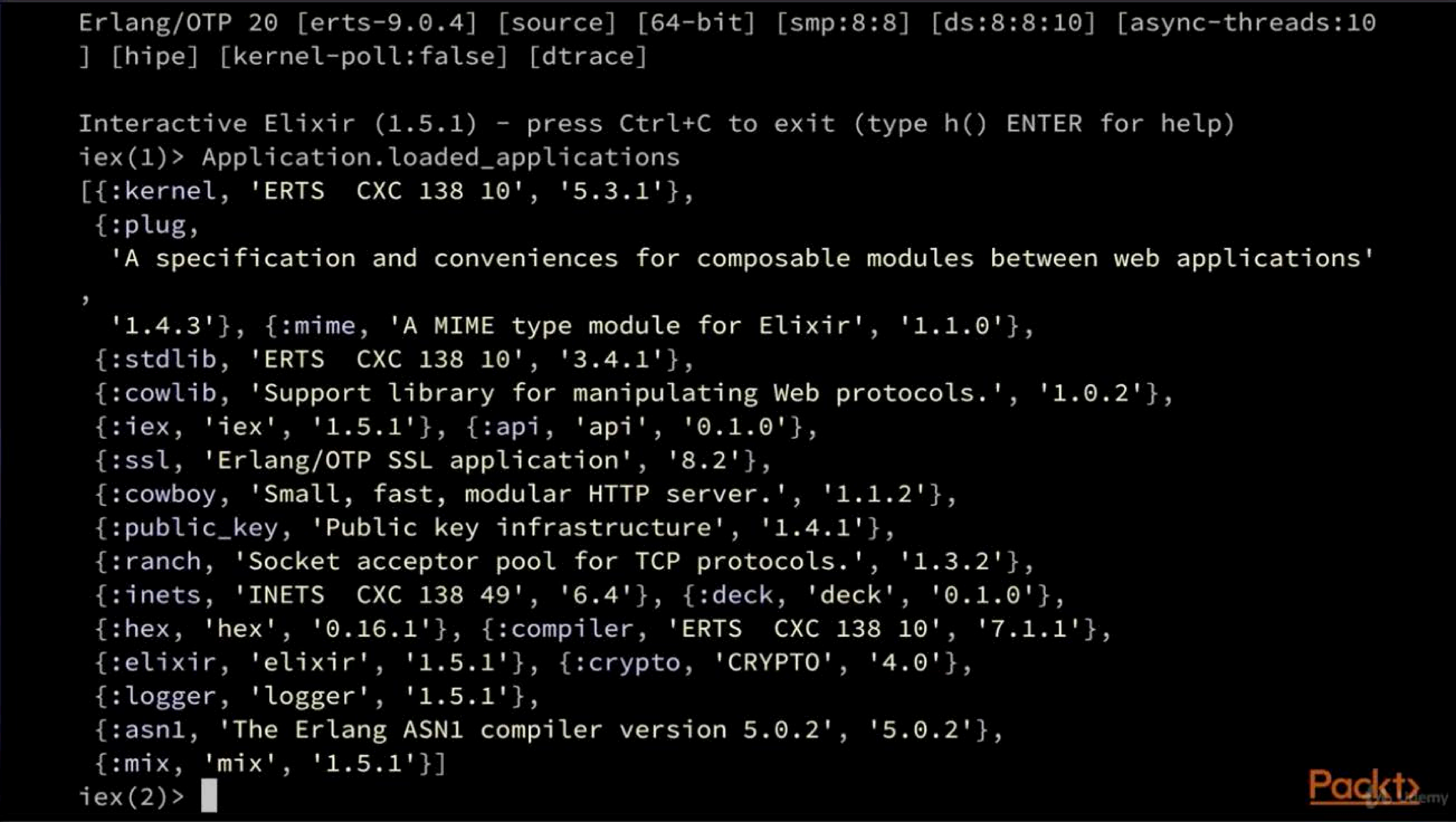
```
defmodule Api.Router do
  use Plug.Router

  @suites ["Spades", "Clubs", "Diamonds", "Hearts"]
  @cards ["Ace", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "\
Ten", "Jack", "Queen", "King"]

  plug :match
  plug :dispatch

  get "/card" do
    card = Deck.take_card()
    |> convert()

    conn
    |> send_resp(200, card)
  end

  def convert(card) do
    suite = Enum.at(@suites, div(card, 13))
    face = Enum.at(@cards, rem(card, 13))

 ❶  - 495 router.ex   Elixir   alchemist ⓨⓢⓟⓐⓚ   Git-master   unix | 1: 0   To
SPC p t
```



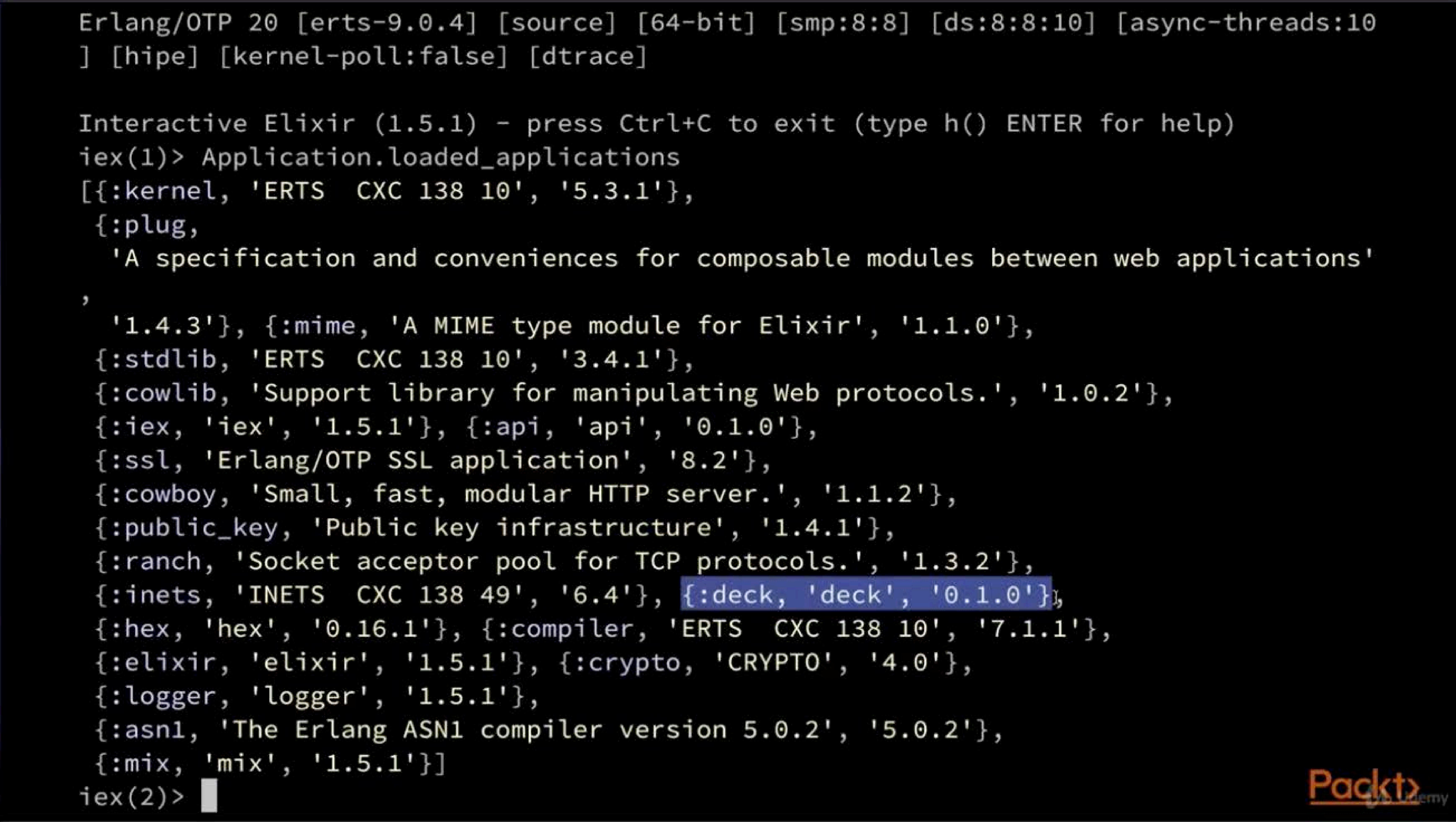
```
  @cards ["Ace", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "\
Ten", "Jack", "Queen", "King"]

  plug :match
  plug :dispatch

  get "/card" do
    card = Deck.take_card()
    |> convert()

    conn
    |> send_resp(200, card)
  end

  def convert(card) do
    suite = Enum.at(@suites, div(card, 13))
    face = Enum.at(@cards, rem(card, 13))

    "#{face} of #{suite}"
  end
end

 ❶  - 495 router.ex   Elixir   alchemist ⓨⓈⓅⓐⓀ   Git-master   unix | 25: 0   B
End of buffer
```



```
➜  card_deck git:(master) iex -S mix
Erlang/OTP 20 [erts-9.0.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:10
] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> 
```

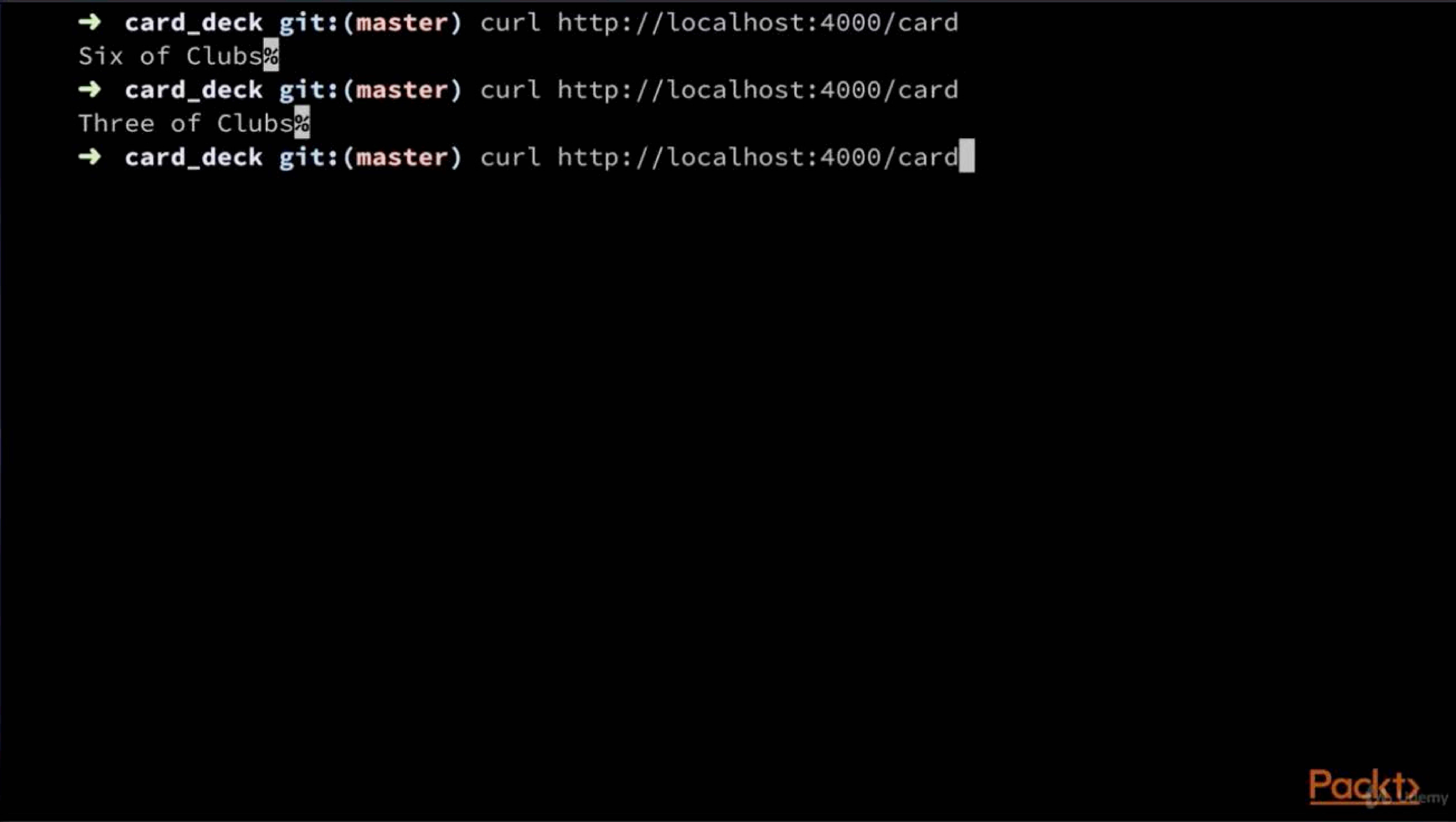


```
Erlang/OTP 20 [erts-9.0.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:10
] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Application.loaded_applications
[{:kernel, 'ERTS  CXC 138 10', '5.3.1'},
 {:plug,
  'A specification and conveniences for composable modules between web applications'
,
  '1.4.3'}, {:mime, 'A MIME type module for Elixir', '1.1.0'},
 {:stdlib, 'ERTS  CXC 138 10', '3.4.1'},
 {:cowlib, 'Support library for manipulating Web protocols.', '1.0.2'},
 {:iex, 'iex', '1.5.1'}, {:api, 'api', '0.1.0'},
 {:ssl, 'Erlang/OTP SSL application', '8.2'},
 {:cowboy, 'Small, fast, modular HTTP server.', '1.1.2'},
 {:public_key, 'Public key infrastructure', '1.4.1'},
 {:ranch, 'Socket acceptor pool for TCP protocols.', '1.3.2'},
 {:inets, 'INETS  CXC 138 49', '6.4'}, {:deck, 'deck', '0.1.0'},
 {:hex, 'hex', '0.16.1'}, {:compiler, 'ERTS  CXC 138 10', '7.1.1'},
 {:elixir, 'elixir', '1.5.1'}, {:crypto, 'CRYPTO', '4.0'},
 {:logger, 'logger', '1.5.1'},
 {:asn1, 'The Erlang ASN1 compiler version 5.0.2', '5.0.2'},
 {:mix, 'mix', '1.5.1'}]
iex(2)> █
```



```
Erlang/OTP 20 [erts-9.0.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:10
] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Application.loaded_applications
[{:kernel, 'ERTS  CXC 138 10', '5.3.1'},
 {:plug,
  'A specification and conveniences for composable modules between web applications'
,
  '1.4.3'}, {:mime, 'A MIME type module for Elixir', '1.1.0'},
 {:stdlib, 'ERTS  CXC 138 10', '3.4.1'},
 {:cowlib, 'Support library for manipulating Web protocols.', '1.0.2'},
 {:iex, 'iex', '1.5.1'}, {:api, 'api', '0.1.0'},
 {:ssl, 'Erlang/OTP SSL application', '8.2'},
 {:cowboy, 'Small, fast, modular HTTP server.', '1.1.2'},
 {:public_key, 'Public key infrastructure', '1.4.1'},
 {:ranch, 'Socket acceptor pool for TCP protocols.', '1.3.2'},
 {:inets, 'INETS  CXC 138 49', '6.4'}, {:deck, 'deck', '0.1.0'},
 {:hex, 'hex', '0.16.1'}, {:compiler, 'ERTS  CXC 138 10', '7.1.1'},
 {:elixir, 'elixir', '1.5.1'}, {:crypto, 'CRYPTO', '4.0'},
 {:logger, 'logger', '1.5.1'},
 {:asn1, 'The Erlang ASN1 compiler version 5.0.2', '5.0.2'},
 {:mix, 'mix', '1.5.1'}]
iex(2)>
```

```
➜  card_deck git:(master) curl http://localhost:4000/card
Six of Clubs%
➜  card_deck git:(master) 
```



```
➜  card_deck git:(master) curl http://localhost:4000/card
Six of Clubs%
➜  card_deck git:(master) curl http://localhost:4000/card
Three of Clubs%
➜  card_deck git:(master) curl http://localhost:4000/card
```

Pitfalls

Pitfalls

Dependency Conflicts

Pitfalls

Dependency Conflicts

Start-up and configuration of “Common” dependent Applications

Pitfalls

Dependency Conflicts

Start-up and configuration of “Common” dependent Applications

Complexity