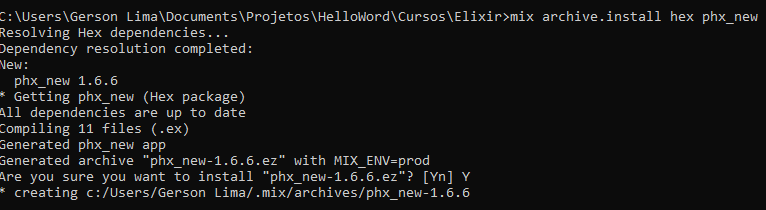
# Startando o APP Web com Phoenix

Para rodar uma aplicação web no Elixir precisamos da extensão Phoenix na nossa máquina. Vamos então rodar o seguinte comando para fazer a instalação do Phoenix:

* mix archive.install hex phx\_new

Saída:



Uma vez com o Phoenix instalado para criar uma aplicação podemos rodar o seguinte comando( não execute sem ler as tags):

* mix phx.new <nome do projeto>

Porém, se rodarmos o código desta maneira, ele criara uma aplicação pronta para rodar com JSON, HTML e com banco de dados através do ecto. Para isso temos algumas tags opcionais para criação, sendo elas:

**--no-ecto:** Não será gerada a funcionalidade do banco de dados com ecto.

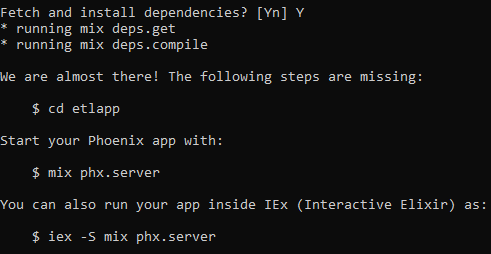
**--no-html:** Não prepara um front-end(bom para API)

**--no-webpack:** Não prepara o pacote web padrão(recomendado para API)

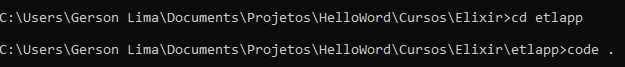
Como é um conteúdo introdutório, não vamos gerar nenhum das requisições, então vamos rodar o comando desta maneira:

* mix phx.new <nome do projeto> –no-ecto –no-html –no-webpack

Resumo da saída:



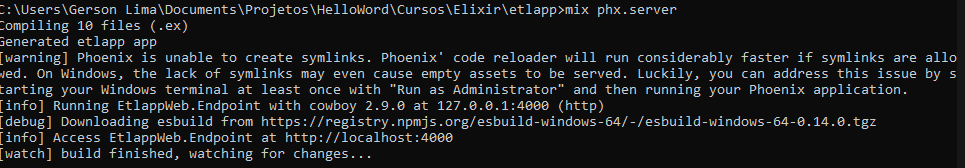
Finalizando os processos de instalação, vamos agora entrar na past do app criado e abrir com o vscode (opcional, pode ser usado um editor de preferência).



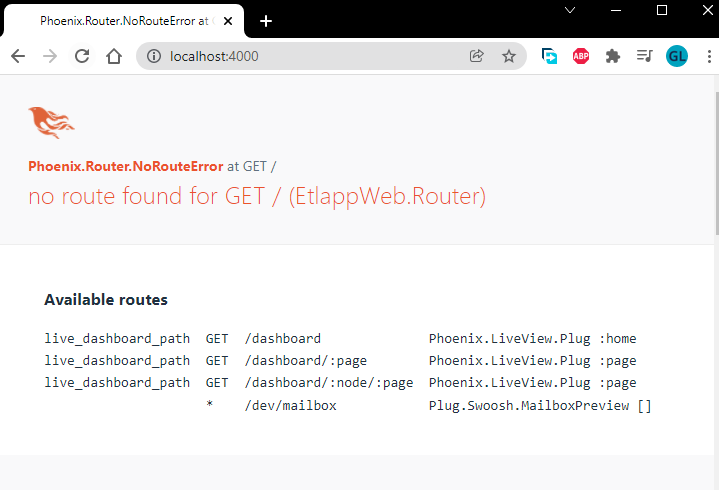
Ainda no terminal, podemos rodar o seguinte comando para executar o server:

* mix phx.server

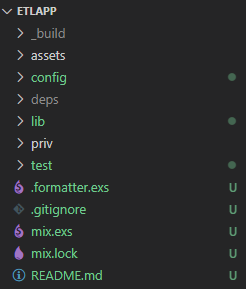
Saída esperada:



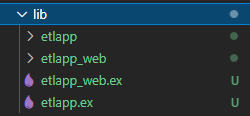
Agora se entrarmos no navegador e na URL digitar o IP e porta informados podemos entrar na verificar a execução funcionando (no caso o IP é localhost:4000):



# Entendendo um pouco da Estrutura dos diretórios na aplicação Phoenix:

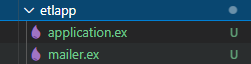


**<lib>**

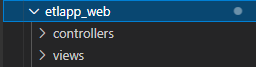


Sempre em uma aplicação Phoenix o diretório lib terá dois contextos:

1. O primeiro é que a pasta padrão dentro do lib sem ser o web, nesse caso a <etlapp> ela compõe tudo que envolve regra de negócios, relação com banco de dados, modelagem de dados estará nela.

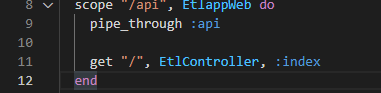


1. E tudo dentro da pastinha com o nome da aplicação mais o “\_web” (nesse caso o <etlapp\_web>) irá conter tudo relacionado a web, ou seja, os controllers, os views e as rotas.



# Traçando Rota

Primeiro, vamos fazer o seguinte caminho: > lib > etlapp\_web ao chega no diretório da web vamos entrar em router.ex.  
Dentro do router.ex no método principal nos scope “/api” vamos traçar o nome da controle com o diretório de chamada



get "/", EtlController, :index

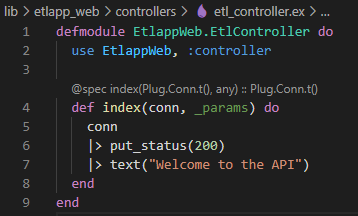
Nesse caso EtlController é a controller que aindar iremos criar e o :index é o caminho de requisição da controller.

# Criando a Controller

Na pasta controllers vamos criar um novo arquivo:



Dentro do arquivo criado, vamos escrever nossa controller da seguinte maneira:



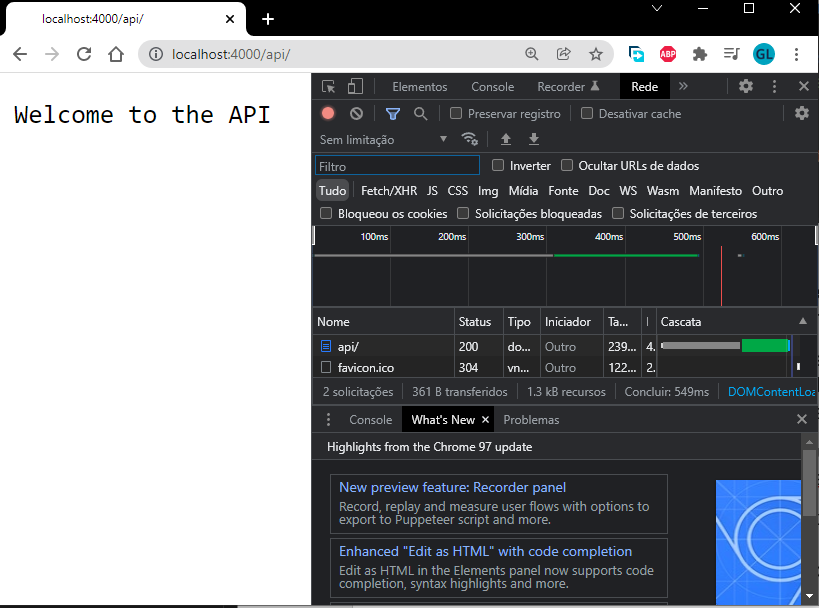
Dentro criamos um def index(), e é importante entender que toda action de uma controller recebe 2 parametros sempre:

* Os parâmetros que vieram através da rota, ou seja, posso receber parâmetros via protocolo HTTP ou via URL.
* E também os parâmetros para manipularmos nossa conexão.

Sendo assim, o index ficou da seguinte maneira def index(conn, \_params), mas por enquanto não usaremos o “\_params” mas ele é obrigatório.

Em seguida dentro da função index, pegamos a conexão “conn”, chamamos uma função da controller colocando o status 200 “|> put\_status(200)” e chamamos outra função para devolver um texto “|> text (‘Welcome to the API’)”

Acessando o caminho do scope “/api” mais o diretório “/” configurado no get “/” do router.ex com o caminho <http://localhost:4000/api/>, temos o seguinte retorno:



Com esse retorno vemos o nosso text configurado “Welcome to the API” e vemos também no console de Rede o Status 200.

# Consumir API

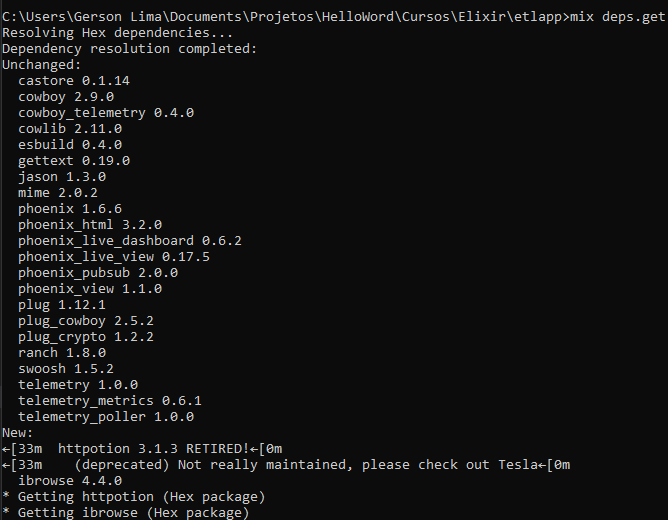
Para consumir uma outra API, primeiramente iremos precisar adicionar a biblioteca HTTPotion que pode ser encontrado no link: <https://hexdocs.pm/httpotion/readme.html>

Pegaremos do endereço o seguinte trecho e adicionar na função deps do arquivo mix.exs: {:httpotion, "~> 3.1.0"}. Desta forma:

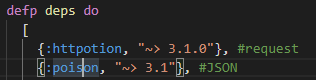


Com este trecho adicionado, iremos agora rodar no nosso terminal o comando:

* mix deps.get

Saída: 

Faremos o mesmo para a bilblioteca {:poison, "~> 3.1"}:



Com este trecho adicionado, iremos agora rodar no nosso terminal o comando:

* mix deps.get

Saída: 