

T7 - MSc Pool

T-POO-700

API

Bootstrap



 $\{$ EPITECH. $\} oldsymbol{\mathsf{L}}$



The objective of this bootstrap is the creation of an API to manage a todo list. The goal here being to manipulate an API, we do not want/need a Front-end so far.

To do this, we will use Elixir and the Phoenix framework, as well as PostGreSQL for the database.



Carefully check you installed the necessary tools to use these tools.

PROJECT CREATION

Let's create the project. Here is the command to do so.

```
Terminal

- + x

~/T-P00-700> mix phx.new XXX -app YYY -module Todolist -no-html -no-webpack

creating todo-list/config/config.exs

creating todo-list/config/dev.exs

creating todo-list/config/prod.exs

...

creating todo-list/assets/static/images/phoenix.png

creating todo-list/assets/static/favicon.ico

Fetch and install dependencies? [Yn]
```



You have to understand this very basic command, and what is the purppose of --no-html and --no-webpack arguments.

Install the necessary dependencies.

Take some time to browse the created directory, and check some of the files in it.

We now need to initialize the PostgreSQL database to store the application data.

 ∇ Terminal - + χ \sim /T-P00-700> mix ecto.create





DATA SCHEMAS

Let's discuss the notion of relationship between two tables.

- 1. Being a first data schema corresponding to a user:
- a first name
- a last name
- 2. A second schema corresponding to a task:
- a title
- a description
- a status
- a user

First, generate these two schemas by applying the proper command.

Then 'migrate' your database.

Finally, create associations between tasks and users.



we want to access a user from the task he is working on. For example, task.user gives us the information of the assigned user.

ROUTES

Once your database is properly *configured* and your associations made, we need to set up the routes. We want to set up a CRUD for user management.

Access the router of your application in the following file: lib/todo_list_web/router.ex. It must be somehow similar to this:

```
defmodule TodolistWeb.Router do
    use TodolistWeb, :router

pipeline :api do
    plug :accepts, ["json"]
  end

scope "/api", TodolistWeb do
    pipe_through :api
  end
end
```





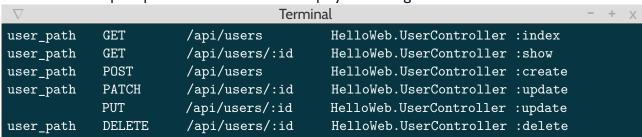
You can therefore add the following command within the scope /api:

resources "/users", UserController, except: [:new, :edit]



You must understand this command!

The command to prompt the routes should now display something like:



Now, in order to properly manage the associations between the tasks and the users, or even the tasks themselves, we will set up dedicated routes.

We will first add a scope, inside the scope / api, which will be named / tasks. Inside this same scope, set up various routes.

- GET http://localhost:4000/api/tasks => code 200 if there is no error.
- GET http://localhost:4000/api/tasks/:id => code 200 if the task is found.
 - => code 404 if the task is unknown.
- POST http://localhost:4000/api/tasks
 code 201 if creation succeeds
 code 400 if there is an error
- DELETE http://localhost:4000/api/tasks/:id
- PUT http://localhost:4000/api/tasks/:id
- GET http://localhost:4000/api/tasks/users/:idUser => code 200 if the task is found.
 - => code 404 if the task is unknown.

