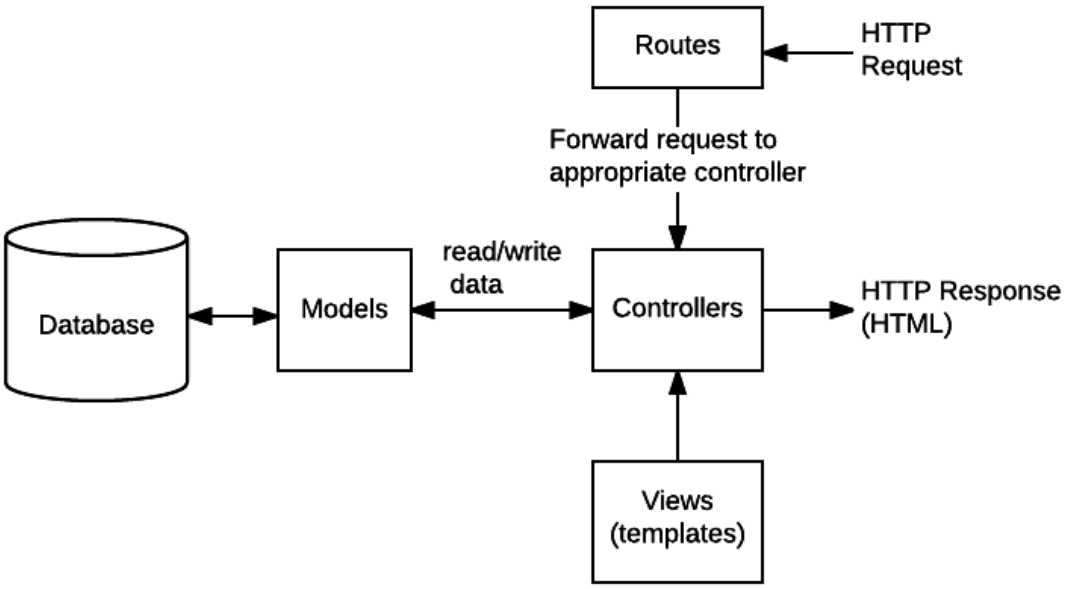


Schemat aplikacji (źródło: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes)

Database – mongoDB, do zarządzania MongoDBCompass, do obsługi mongoose

Models – modele, schematy, czyli to wszystko jak wygląda dany element kolekcji

Routes – do przekazywania żądań i informacji pochodzących z adresu url

Controllers – łączy modele się z modelami, generuje widoki i w tym miejscu wykonywane są wszystkie obróbki danych.

Views – widoki, czyli szablony do wyświetlania danych, warstwa interfejsu użytkownika, może już zawierać wszystkie graficzne technologie np. html i css

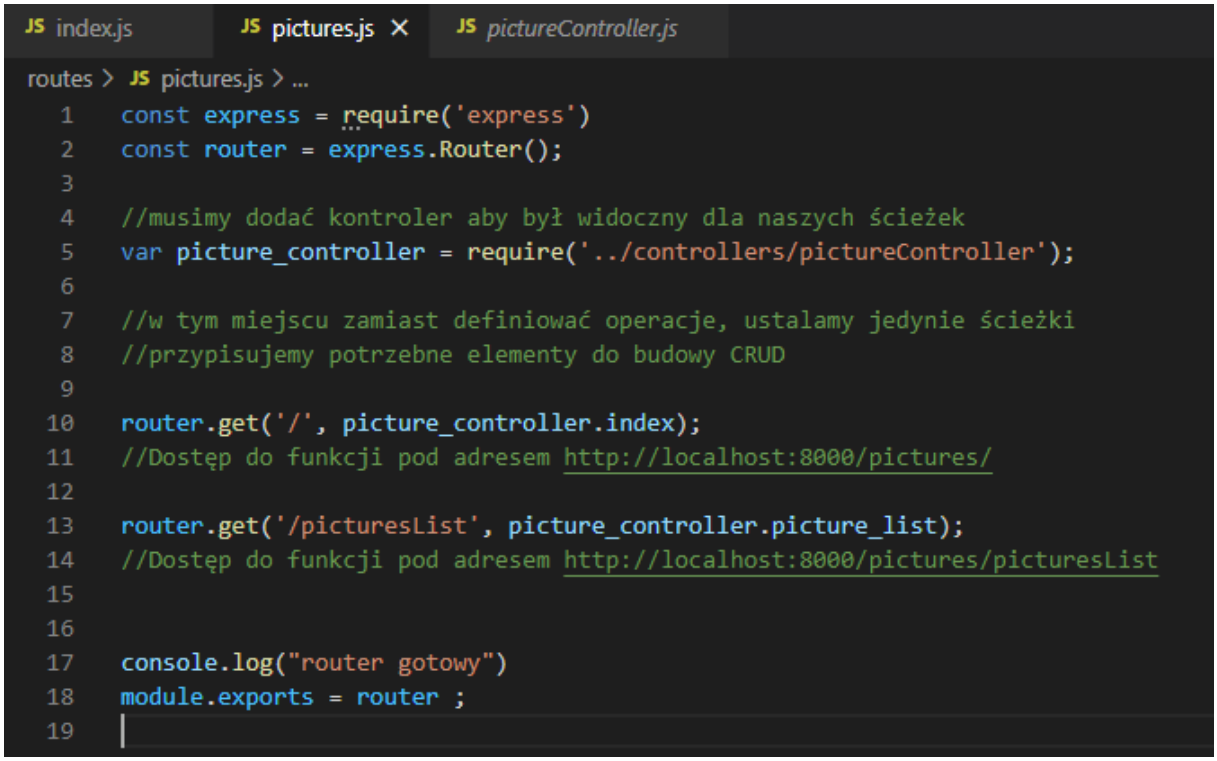
Porządki zgodnie ze schematem

Wszystkie operacje na bazie danych będzie trzeba przenieść z pliku index.js. Zatem na początku zgodnie z tutorialem (https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes) zakomentujmy dotychczasowe operacje i ustalmy routers dla Picture. Od razu możemy stworzyć:

1. folder controllers, a w środku pictureController.js,
2. folder views
3. plik w folderze models picture.js
4. plik w folderze routes pictures.js

Należy zatem dodać linijkę 6 i 13. Po lewej stronie widać od razu układ plików. Dzięki dodaniu ścieżki mamy teraz możliwość przerzucić poszczególne operacje do innych plików robiąc aplikację bardziej czytelną.

Zajmijmy się plikiem routes/pictures.js – w tym pliku będą ścieżki po których będziemy odnosić się do funkcji kontrolera.



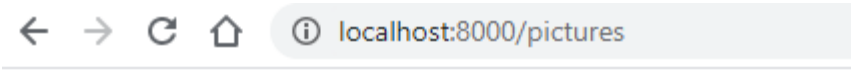
Kolejnym krokiem będzie zdefiniować kontroler dla tych funkcji, dla przykładu niech to będzie funkcja obsługująca router.get('/') dla domyślnej strony dla ścieżki pictures, oraz router.get('picturesList') do obsługi wylistowania wszystkich elementów kolekcji zdjęć. W komentarzu podane dokładne ścieżki po których możemy je uruchomić.

```
JS index.js   JS pictures.js   JS pictureController.js ×
controllers > JS pictureController.js > ...
1   var Picture = require('../models/picture');
2
3   // zdefiniowanie pustych funkcji
4   exports.index = function(req, res) {
5       console.log("wywołanie get index")
6       res.send('NOT IMPLEMENTED: Site Home Page');
7   };
8
9   exports.picture_list = function(req, res) {
10      console.log("wywołanie get pictures")
11      res.send('NOT IMPLEMENTED: Picture list');
12  };
13
14  console.log("kontroler gotowy")
```

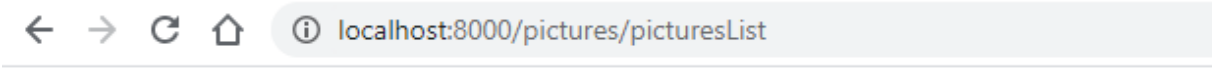
Wygląda że wszystko jest poprawnie przeniesione, dodatkowo dołączyłem w każdym pliku odpowiednie komunikaty, aby zobaczyć czy wszystkie połączenia są prawidłowe i aplikacja je widzi.

```
PS C:\Web Dev\szkieletAplikacji + controllers> node index.js
model gotowy
kontroler gotowy
router gotowy
baza danych uruchomiona
```

Możemy również sprawdzić działanie przy pomocy ścieżki:



NOT IMPLEMENTED: Site Home Page



NOT IMPLEMENTED: Picture list

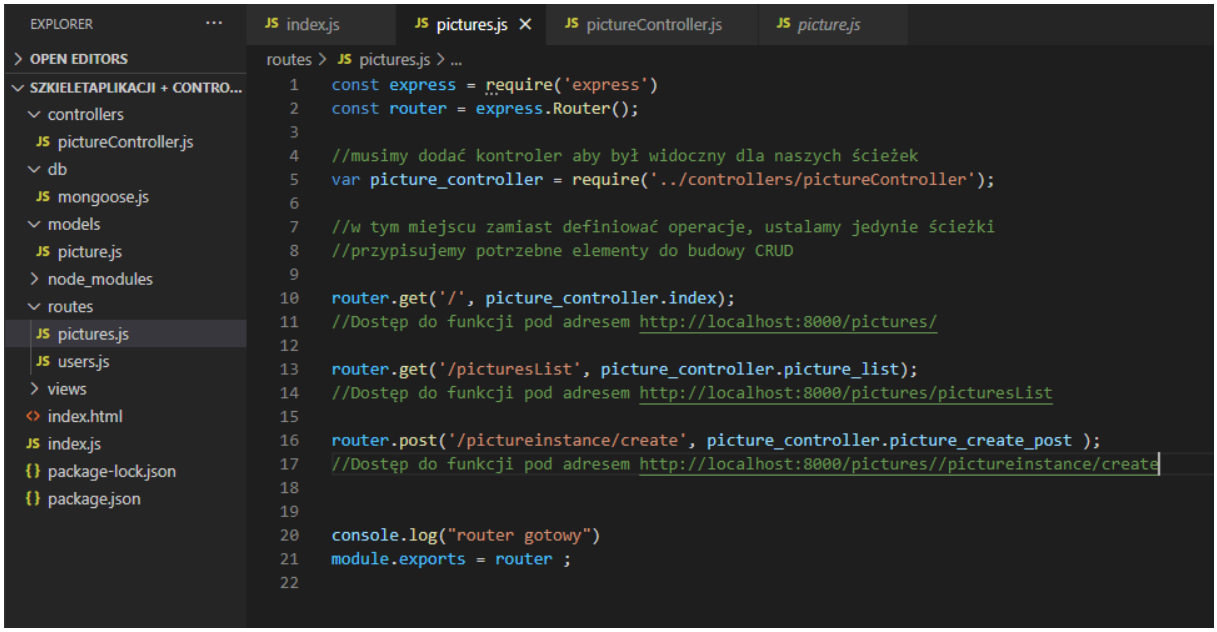
Teraz możemy uzupełnić kontroler aby pobierał dane z bazy danych. Możemy zatem przenieść utworzone wcześniej operacje na bazie danych do konkretnych funkcji w kontrolerze. W przykładzie poniżej będzie wypisanie elementów kolekcji zdjęć.

```
var Picture = require('../models/picture');

// zdefiniowanie pustych funkcji
exports.index = function(req, res) {
    console.log("wywołanie get index")
    res.send('NOT IMPLEMENTED: Site Home Page');
};

exports.picture_list = function(req, res) {

    let picturesList

    const getPictures = async () => {
        try{
            picturesList = await Picture.find()
            console.log(picturesList)
            res.send(picturesList);
        }catch(err){
            console.log(err)
        }
    }

    getPictures()


};

console.log("kontroler gotowy")
```

localhost:8000/pictures/picturesList


```
[
  {
    "sciezka": "./images",
    "_id": "5fb12f8ea686470558f27286",
    "nazwa": "foto_02",
    "rozmiar": 5345,
    "__v": 0
  },
  {
    "sciezka": "./images",
    "_id": "5fb131a7a8ddd3410446623f",
    "nazwa": "foto_02",
    "rozmiar": 3345,
    "__v": 0
  },
  {
    "sciezka": "./images",
    "_id": "5fb14f86411db043f0ad93f7",
    "nazwa": "foto_03",
    "rozmiar": 4567,
    "__v": 0
  },
  {
    "sciezka": "./images",
    "_id": "5fb8f4ec9e14de0340b90047",
    "nazwa": "foto_03",
    "rozmiar": 4567,
    "__v": 0
  }
]
```

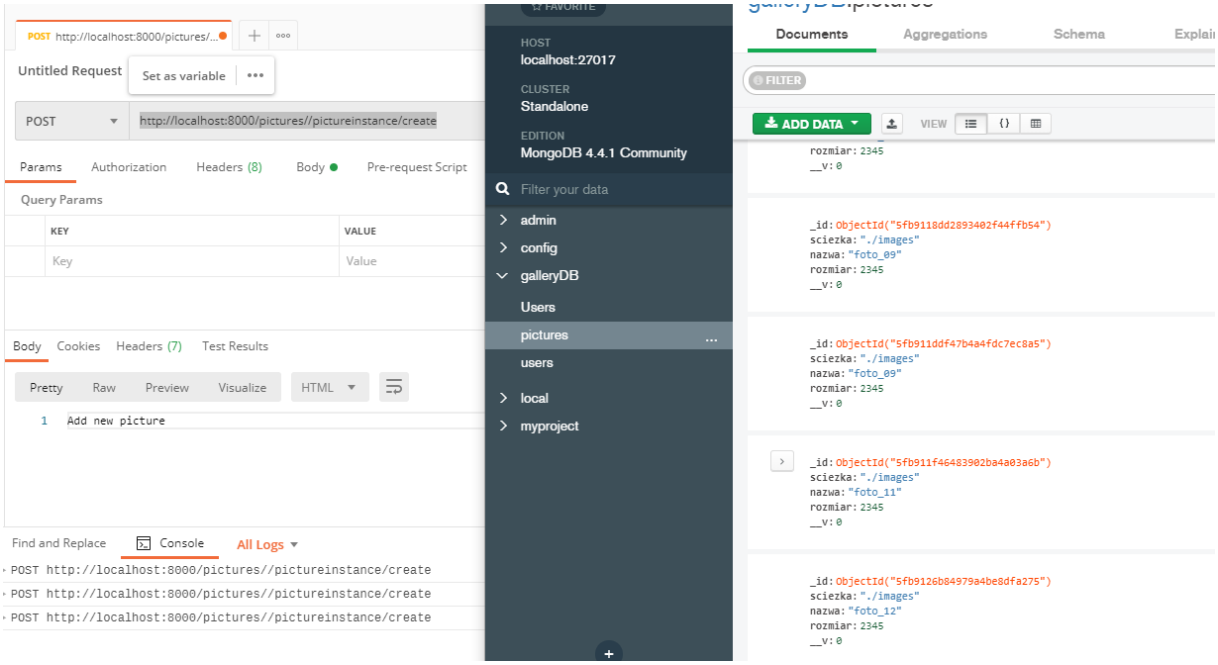

Jeśli chodzi o metodę Post, możemy stworzyć z tymczasowymi danymi na sztywno i dodać nowe zdjęcie. Zacznijmy od ścieżki.



A następnie w kontrolerze musimy mieć możliwość dodać użytkownika, nie mamy jeszcze żadnych formularzy w widoku dlatego zróbmy na sztywno.

```
27
28  exports.picture_create_post = function(req, res) {
29
30      const picture = new Picture({nazwa: 'foto_12', sciezka: './images', rozmiar: '2345'})
31      picture.save().then(() => {
32          console.log(picture)
33          res.send('Add new picture');
34      }).catch(err => {
35          console.log(err)
36      })
37
38  };
39
40  console.log("kontroler gotowy")
```

Teraz aby przetestować musimy użyć Postmana i dla podanego url wybrać metodę post i wysłać.



Po odświeżeniu w Postmanie możemy zobaczyć że pozycja została dodana.

Zadanie.

Na podstawie tego tutorialu utwórz wszystkie potrzebne kontrolery i przetestuj działanie. Wszystkie potrzebne funkcje możemy zaprojektować na podstawie API.