



# Express.js

## Structure our code/project

# Εισαγωγή

- Καθώς το project μας μεγαλώνει, είναι ιδιαίτερα σημαντικό να το οργανώσουμε με τέτοιο τρόπο ώστε να κάνει ευκολότερη την ανάγνωσή του και την συντήρησή του.
- Στη συνέχεια θα δούμε ορισμένα απλά βήματα για το πως μπορούμε να το επιτύχουμε αυτό

# Εισαγωγή

- Στο project που έχουμε δει μέχρι τώρα, αρχικά είδαμε ότι δημιουργήσαμε
- Routers
- Route handlers ή Controllers
- Και τους διαχωρίσαμε έτσι ώστε να είναι πιο ευανάγνωστος ο κώδικας.

# Εισαγωγή

- Έστω τώρα ότι θέλουμε να δημιουργήσουμε Routes και Controllers για τους χρήστες (users) που θα έχουμε στο σύστημά μας.

```
//user controllers
> const getAllUsers = (req,res)=>{ ...
}

> const getUserById = (req,res)=>{ ...
}

> const addUser= (req,res)=>{ ...
}

> const updateUserById = (req,res)=>{ ...
}

> const deleteUserById = (req,res)=>{ ...
}
```

```
//User Routes
app.route('/api/v1/users')
  .get(getAllUsers)
  .post(addUser)

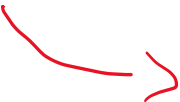
app.route('/api/v1/users/:id')
  .get(getUserById)
  .patch(updateUserById)
  .delete(deleteUserById)
```

# Διαχωρισμός των routers

- Μια καλή πρακτική είναι ο λογικός διαχωρισμός/ οργάνωση των routers σε διαφορετικά αρχεία.
- Θα ξεκινήσουμε χρησιμοποιώντας την παρακάτω εντολή η οποία δημιουργεί ένα νέο **αντικείμενο router** :
- **`express.Router([options])`**

# express.Router([options])

- Ένα **αντικείμενο δρομολογητή (router object)** αποτελεί ένα `middleware`
- Κάθε εφαρμογή Express διαθέτει **ενσωματωμένο app router**, όπως είδαμε στον κώδικα



```
//User Routes
app.route('/api/v1/users')
  .get(getAllUsers)
  .post(addUser)
```

- Με τη μέθοδο `Router()` μπορούμε να δημιουργήσουμε ένα νέο αντικείμενο δρομολογητή (router object)

# express.Router([options])

```
9 //lets create a new router-> it is a middleware (1)  
0 const landmarkRouter = express.Router();  
1 //here we specify the route(URL) for which we wish to use our middleware(landmarkRouter)  
2 app.use('/api/v1/landmarks', landmarkRouter); (2)  
3  
4  
5
```

↓  
path

↓  
middleware

# express.Router([options])

- 1) Τώρα που με τη μέθοδο Router () δημιουργήσαμε ένα νέο αντικείμενο δρομολογητή (router object) : router object handles request!
- 2) Κάναμε mount τον νέο router σε ένα route(ορίζουμε για πιο route/url/path θα τρέχει αυτό το router object)
- 3) μπορούμε να αντικαταστήσουμε τον app router
- 4) και να αλλάξουμε το url στη μέθοδο route



# Διαχωρισμός των routers

```
//ROUTES
```

```
//lets create a new router-> it is a middleware
const landmarkRouter = express.Router();
//here we specify the route(URL) for which we wish to use our middleware(landmarkRouter)
app.use('/api/v1/landmarks',landmarkRouter);
```

```
//landmark Routes
3 landmarkRouter.route('/') ← 4
  .get(getAllLandmarks)
  .post(addLandmark)
```

```
3 landmarkRouter.route('/:id') 4
  .get(getLandmarkById)
  .patch(updateLandmarkById)
  .delete(deleteLandmarkById)
```

Πως να αν...

```
//landmark Routes
app.route('/api/v1/landmarks')
  .get(getAllLandmarks)
  .post(addLandmark)

app.route('/api/v1/landmarks/:id')
  .get(getLandmarkById)
  .patch(updateLandmarkById)
  .delete(deleteLandmarkById)
```

# Οργάνωση της δομής των αρχείων

- Στο app.js αρχείο έχουμε
  - modules
  - Middleware
  - Route handlers ή Controllers
  - Routes

# Οργάνωση της δομής των αρχείων

- Αυτό που θέλουμε να κάνουμε είναι να «σπάσουμε» τον κώδικα σε μικρότερα κομμάτια, σε μικρότερες εφαρμογές ας το πούμε οι οποίες θα κάνουν η καθεμία μια συγκεκριμένη λειτουργία.
- Με `export-import` μετά θα «ενώσουμε» όλα αυτά τα κομμάτια
- Στο `app.js` θα παραμείνει ουσιαστικά ο **προσδιορισμός του `middleware` για όλα τα `routes` και τα `modules`**

# Οργάνωση της δομής των αρχείων

- Στο app.js αρχείο έχουμε προς το παρόν

- modules

- Middleware

- Route handlers ή Controllers

- Routes

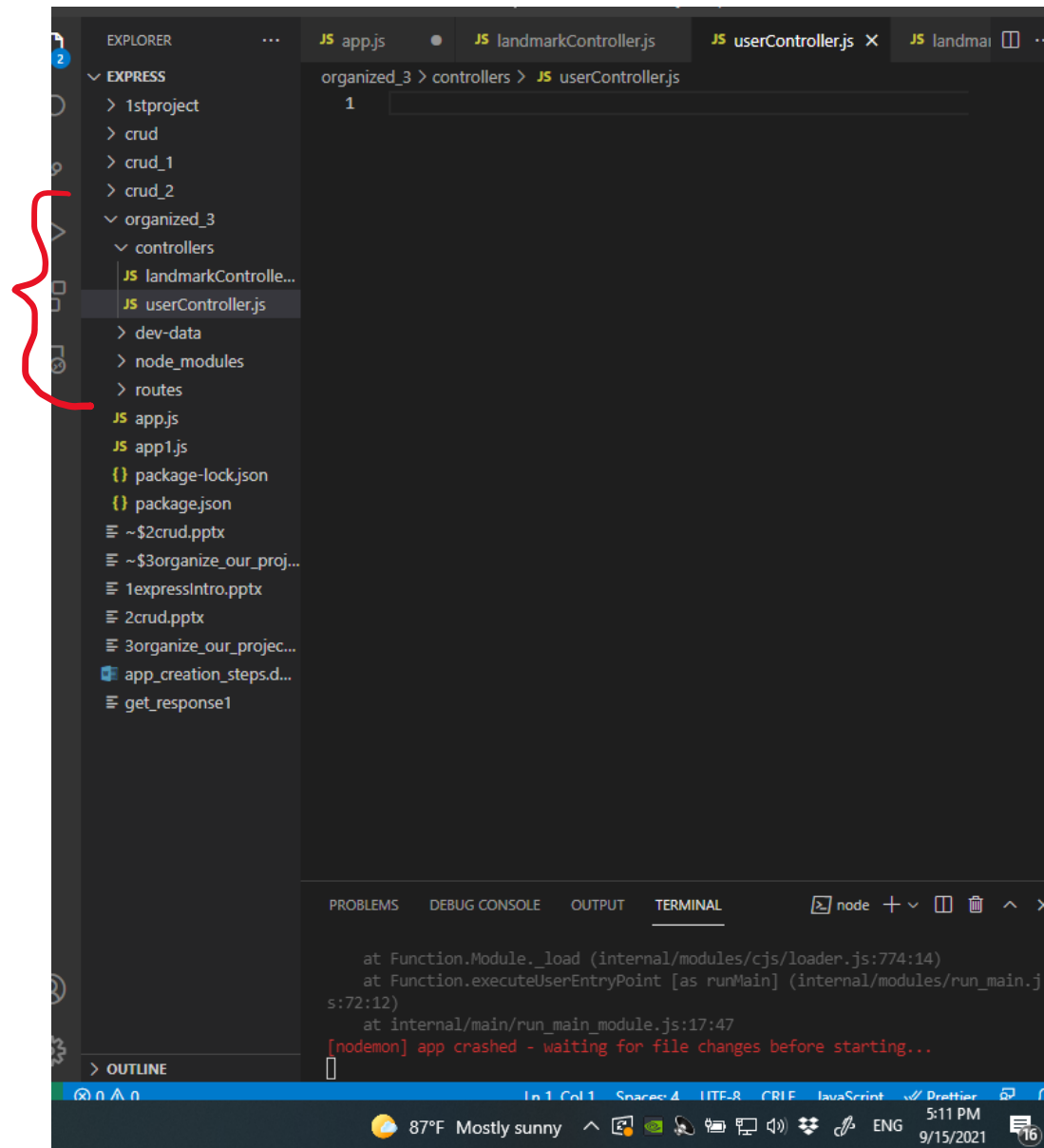
→ Θα αποθηκεύσουν αλληλ

# Διαχωρισμός των routes

- Αφού έχουμε κατανοήσει αυτό το κομμάτι, μπορούμε να προχωρήσουμε στην δημιουργία της παρακάτω δομής (app.js)

```
└─ organized_3
  └─ dev-data
  └─ node_modules
  └─ routes
    └─ JS landmarkRoutes.js
    └─ JS userRoutes.js
  └─ JS app.js
  └─ JS app1.js
  └─ {} package-lock.json
  └─ {} package.json
```

# Διαχωρισμός των controllers



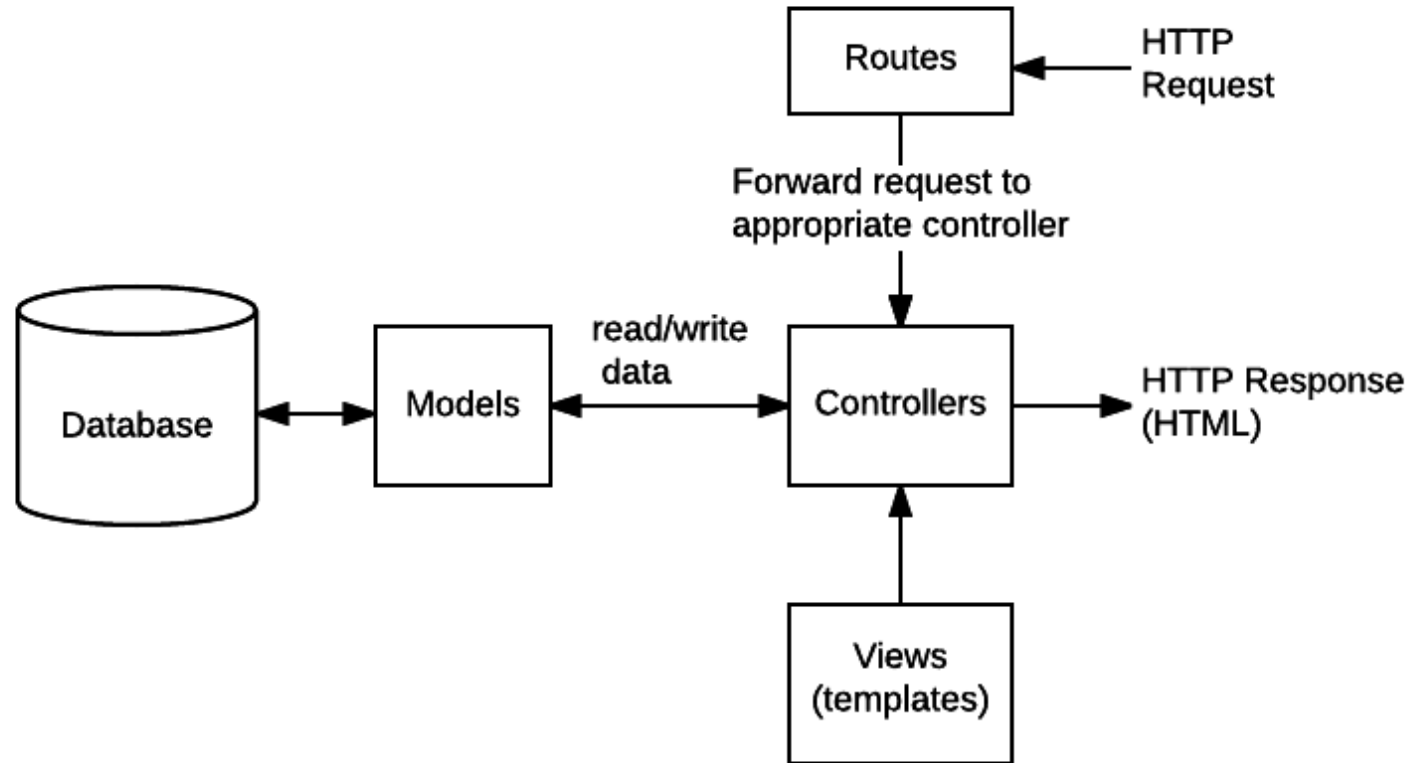
The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure under the name 'EXPRESS'. A red bracket highlights the 'controllers' directory and its contents: 'landmarkControlle...', 'userController.js', 'dev-data', 'node\_modules', and 'routes'. The main editor window shows the 'userController.js' file with a single line of code: '1'. The terminal at the bottom displays a stack trace and a message: '[nodemon] app crashed - waiting for file changes before starting...'. The system tray at the bottom shows the date and time as 5:11 PM on 9/15/2021.

```
EXPLORER  
EXPRESS  
  > 1stproject  
  > crud  
  > crud_1  
  > crud_2  
  > organized_3  
    > controllers  
      JS landmarkControlle...  
      JS userController.js  
    > dev-data  
    > node_modules  
    > routes  
  JS app.js  
  JS app1.js  
  {} package-lock.json  
  {} package.json  
  ~$crud.pptx  
  ~$3organize_our_proj...  
  1expressIntro.pptx  
  2crud.pptx  
  3organize_our_projec...  
  app_creation_steps.d...  
  get_response1
```

```
organized_3 > controllers > JS userController.js  
1
```

```
at Function.Module._load (internal/modules/cjs/loader.js:774:14)  
at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:72:12)  
at internal/main/run_main_module.js:17:47  
[nodemon] app crashed - waiting for file changes before starting...
```

# Οργάνωση της δομής των αρχείων



Source: [https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\\_Nodejs/routes](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes)

# Create our own modules

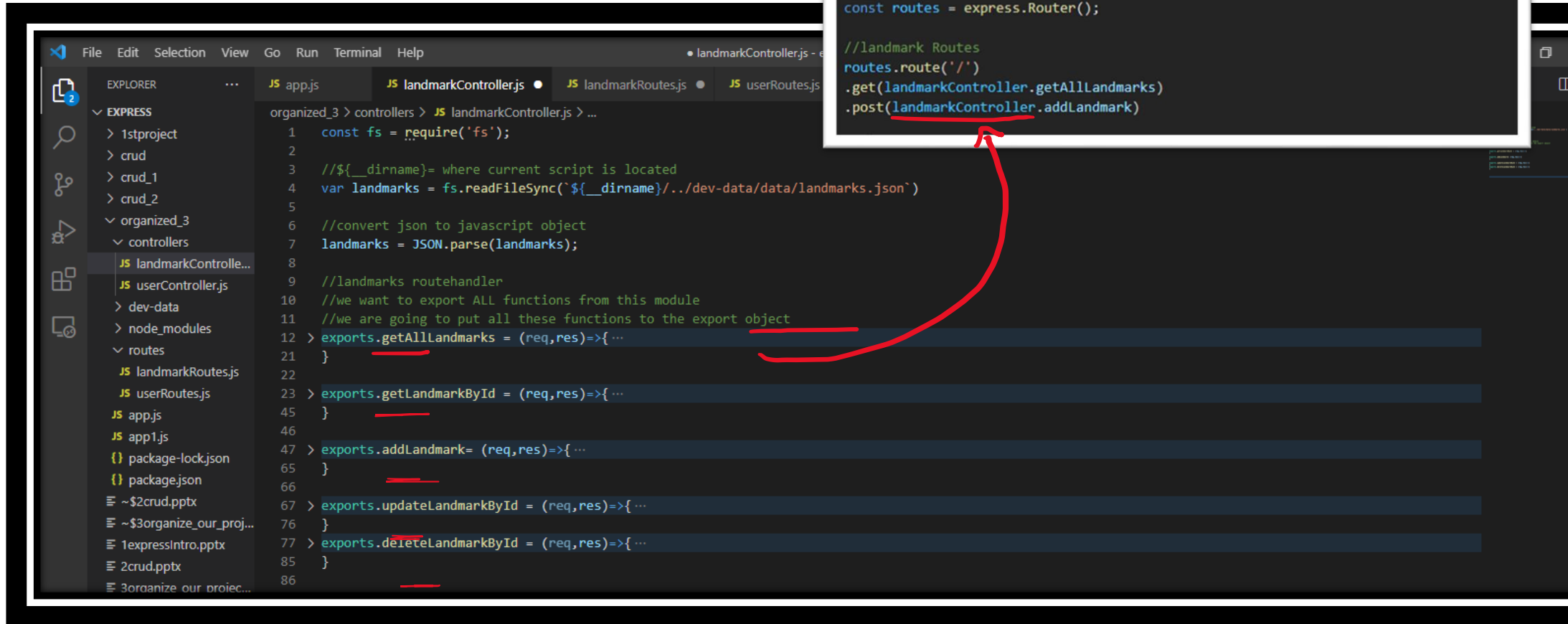
- Μπορούμε εύκολα να δημιουργήσουμε και να ενσωματώσουμε τις δικές μας λειτουργικές μονάδες (modules) στις εφαρμογές μας
- Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το `module.exports`
- Το `module.exports` -> χρησιμοποιείται για την εξαγωγή οποιουδήποτε `var`, `function` ή `object` ως `module` (λειτουργική μονάδα)



# Routehandlers II Controllers

*landmarkRoutes.js*

Export multiple functions...



```

organized_3 > controllers > JS landmarkController.js > ...
1  const fs = require('fs');
2
3  //${_dirname}= where current script is located
4  var landmarks = fs.readFileSync(`${_dirname}/../dev-data/data/landmarks.json`);
5
6  //convert json to javascript object
7  landmarks = JSON.parse(landmarks);
8
9  //landmarks routehandler
10 //we want to export ALL functions from this module
11 //we are going to put all these functions to the export object
12 > exports.getAllLandmarks = (req,res)=>{ ...
21 }
22
23 > exports.getLandmarkById = (req,res)=>{ ...
45 }
46
47 > exports.addLandmark= (req,res)=>{ ...
65 }
66
67 > exports.updateLandmarkById = (req,res)=>{ ...
76 }
77 > exports.deleteLandmarkById = (req,res)=>{ ...
85 }
86

```

```

//import landmarkController module
const landmarkController=require(`${_dirname}/../controllers/landmarkController`);
//lets create a new router-> it is a middleware
const routes = express.Router();

//landmark Routes
routes.route('/')
.get(landmarkController.getAllLandmarks)
.post(landmarkController.addLandmark)

```

require the module  
using the `require` keyword

# Routes

```
JS app.js JS landmarkController.js JS landmarkRoutes.js JS userRoutes.js JS userController.js
organized_3 > routes > JS landmarkRoutes.js > ...
 1 //import express
 2 const express= require('express');
 3 //import landmarkController module
 4 const landmarkController=require(`${__dirname}../controllers/landmarkController`);
 5 //lets create a new router-> it is a middleware
 6 const routes = express.Router();
 7
 8 //landmark Routes
 9 routes.route('/')
10 .get(landmarkController.getAllLandmarks)
11 .post(landmarkController.addLandmark)
12
13 routes.route('/:id')
14 .get(landmarkController.getLandmarkById)
15 .patch(landmarkController.updateLandmarkById)
16 .delete(landmarkController.deleteLandmarkById)
17
18 //lets export our module!
19 module.exports= routes;
```

import  
controller

router object->  
handles request!

← from imported module

# Controllers

```
JS app.js JS landmarkController.js JS landmarkRoutes.js JS userRoutes.js JS userController.js X
organized_3 > controllers > JS userController.js > ...
1
2 //user controllers
3 > exports.getAllUsers = (req,res)=>{ ...
9 }
10
11 > exports.getUserById = (req,res)=>{ ...
18 }
19
20 > exports.addUser= (req,res)=>{ ...
25 }
26
27 > exports.updateUserById = (req,res)=>{ ...
33 }
34 > exports.deleteUserById = (req,res)=>{ ...
40 }
41 |
```

# Routes

```
JS app.js JS landmarkController.js JS landmarkRoutes.js JS userRoutes.js X JS userController.js
organized_3 > routes > JS userRoutes.js > routes
1
2 //import express
3 const express= require('express');
4 //import landmarkController module
5 const userController=require(`_${dirname}../controllers/userController`);
6
7 const routes = express.Router();
8
9 //User Routes
10 routes.route('/')
11 .get(userController.getAllUsers)
12 .post(userController.addUser)
13
14
15 routes.route('/:id')
16 .get(userController.getUserById)
17 .patch(userController.updateUserById)
18 .delete(userController.deleteUserById)
19
20 //lets export our module!
21 module.exports= routes;
```

# App.js

import Routers →

```
appjs > ...
1 //import express
2 const express= require('express');
3 //import morgan
4 const morgan = require('morgan');
5
6 const landmarkRouter= require('./routes/landmarkRoutes')
7 const userRouter= require('./routes/userRoutes')
8
```

mount routers  
with url

```
app.use('/api/v1/landmarks',landmarkRouter);
app.use('/api/v1/users',userRouter);
```

# Server.js

- Είναι καλή πρακτική, επίσης, να διαχωρίζουμε ότι έχει να κάνει με τον server σε ένα ξεχωριστό αρχείο

```
app.js x JS server.js package.json JS landmarkControlle
rganized_3 > JS app.js > app.use() callback
9 // create app variable -> assigned result of calling expre
10 const app= express();
11
12 //MIDDLEWARES
13 app.use(express.json());
14
15 //3rd party middleware morgan
16 app.use(morgan('dev'));
17
18 //my middleware
19 app.use((req,res,next)=>{
20   console.log("hi my friend, I am your middleware!");
21   next();
22 });
23
24 //here we specify the route(URL) for which we wish to use
25 app.use('/api/v1/landmarks',landmarkRouter);
26 app.use('/api/v1/users',userRouter);
27
28 //export app to use it in the server.js file
29 module.exports = app;
```

```
JS app.js JS server.js package.json
rganized_3 > JS server.js > ...
1 const app = require('./app');
2
3 //START SERVER
4 app.listen(8080, () => {
5   console.log('Yeah I run');
6 });
```

- Με τη νέα αυτή δομή πρέπει να τρέχουμε το αρχείο `server.js` όχι το `app.js`...
- Για να μην μπερδευόμαστε, θα χρησιμοποιήσουμε την ιδιότητα "scripts" του αρχείου `package.json` που μας επιτρέπει να ορίσουμε το αρχείο που πρέπει να εκτελεστεί για να «τρέξει» η εφαρμογή μας.
- Για παράδειγμα μέχρι τώρα χρησιμοποιούσαμε την εντολή
- `Nodemon app.js` στο terminal
- Τώρα θα πρέπει να θυμόμαστε να εκτελούμε την εντολή:
- `Nodemon sever.js` στο terminal
- Ας το κάνουμε πιο εύκολο....

Ορίζουμε στα scripts από ποιο αρχείο θα εκτελεστεί η εφαρμογή

Στη συνέχεια εκτελούμε το script που ορίσαμε με την παρακάτω εντολή στο terminal

```
npm start
```

```
server.js package.json ...
ized_3 > {} package.json > ...
{
  "name": "crud",
  "version": "1.0.0",
  "description": "My first crud Ops",
  "main": "index.js",
  ▶ Debug
  "scripts": {
    start: "nodemon server.js"
  },
  "author": "Aristea Kontogianni",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "morgan": "^1.10.0"
  }
}
```



To be continued...