1. **Install Fresh Project:**

For installing a new project:

1. Install NodeJS on your OS.
2. Create a random folder in a random place on your HDD.
3. Create a new package.json.

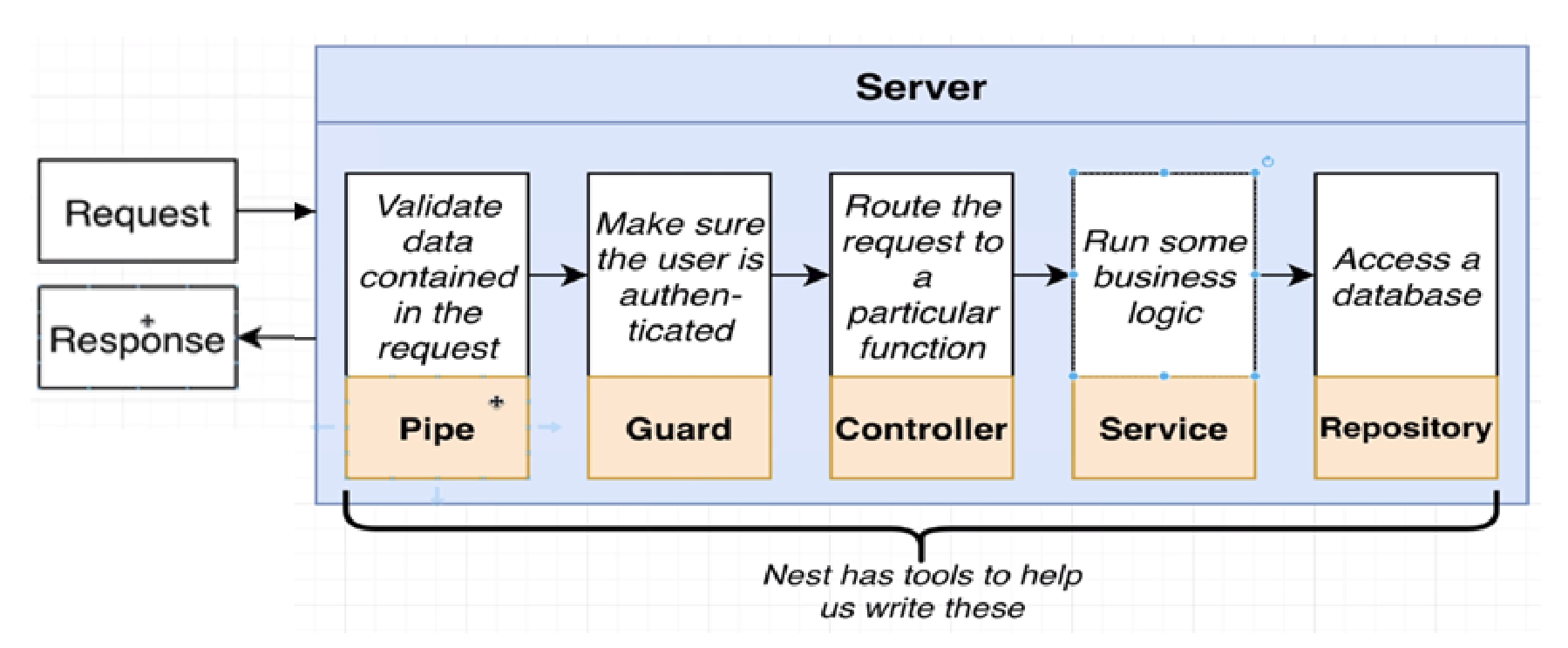
npm init -y

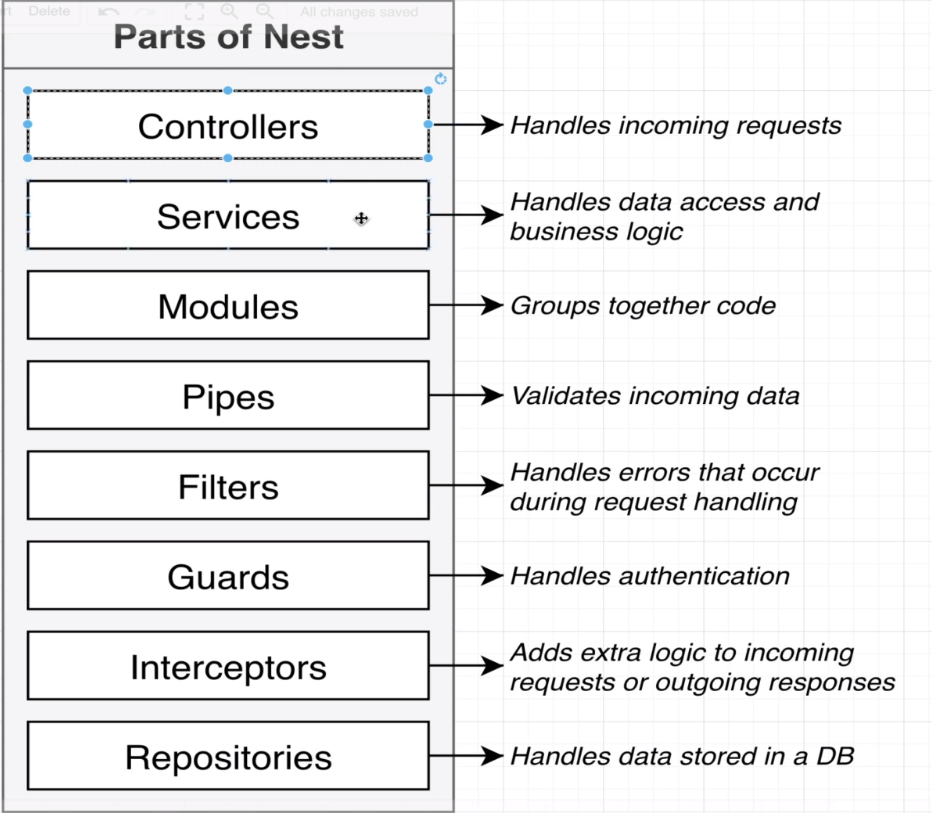
1. Add all required dependencies for creating a new NestJS project.

npm install @nestjs/common @nestjs/core @nestjs/platform-express reflect-metadata typescript

1. Create a new “tsconfig.json” file in the project’s root.

|  |
| --- |
| 1{   1. "compilerOptions": { 2. "module": "CommonJS", 3. "target": "ES2017", 4. "experimentalDecorators": true, 5. "emitDecoratorMetadata": true, 6. } 7. } |





To add new module and controller:

1. Create a new folder named “src” in the project's root.
2. Add “main.ts” file in “src” folder.
3. /\*\*
4. \* This is the first file executed in any NestJS project.
5. \* In this example, we will create modules and controllers here.
6. \* In the following example, we will split the modules and controllers into separate files.
7. \*/
8. // To create a module and controllers, import them from nestjs.
9. // Import Module from nestjs/common.
10. import { Controller, Module, Get } from "@nestjs/common";
11. // Import NestFactory from nestjs/core.
12. import { NestFactory } from "@nestjs/core";
13. // Create Controller.
14. // Call the Controller() decorator. That means we are creating a new controller, and the AppController class is going to serve as a controller inside of the application.
15. @Controller()
16. // Create a new class called AppController.
17. class AppController {
18. // Call the Get() decorator. That means we are creating a new route handler, and the getRootRoute() method will serve as a route handler inside the application.
19. @Get()
20. // Handle a GET request to the root route of the application.
21. getRootRoute() {
22. // Return a string. This message will be shown when we visit the root route of the application.
23. return "Hello World!";
24. }
25. }
26. // Every App needs a module.
27. //We must pass a configuration option or object when using the Module decorator.
28. // Inside the configuration object, we must pass an array of controllers.
29. @Module({
30. // Pass an object.
31. // That is the controllers’ property. It will list all the controllers we want to register with this module.
32. // By that approach, we can register multiple controllers with a single module. Whenever NestJS finds a controller, it will automatically register it with the module by creating a new instance of that controller.
33. controllers: [AppController],
34. })
35. // Create a new class called AppModule.
36. // This class will serve as a module inside of the application.
37. class AppModule {}
38. // Create a new function called bootstrap. That function will be responsible for creating a new NestJS application. The function’s name is not essential, but NestJS uses the 'bootstrap' name by default.
39. async function bootstrap() {
40. // Create a new NestJS application. This creates an instance of the NestApplication class.
41. const app = await NestFactory.create(AppModule);
42. // Start the application on port 3000.
43. await app.listen(3000);
44. }
45. // Call the bootstrap function.
46. bootstrap();
48. // Summary:
49. // We created a new NestJS application.
50. // We created a new module.
51. // We created a new controller and told that controller that we had done the module.
52. // We define the bootstrap function.
53. //      The bootstrap function creates a new instance of the NestApplication class (this is how to create a new NestJS application).
54. //      The bootstrap function starts the application on port 3000.
55. // We call the bootstrap function.
57. // At this stage, we are creating the project from scratch. We have made this file from scratch. If we want to execute this file, we need to run the following command:
58. // > npx ts-node-dev src/main.ts
59. // This command will execute the main.ts file. The npx command will execute the ts-node-dev package. The ts-node-dev package will execute the main.ts file.
60. // The console will show the following message:
61. // > Nest application successfully started
62. // if there is errors, we need to double check if the port 3000 is not used by another application.

In “main.ts” we have…

