In expansion pan

|  |
| --- |
|  |

els found this code, check why it is for?

|  |
| --- |
| <mat-list-item \*ngFor="let folder of folders; last as last">  <mat-icon mat-list-icon>folder</mat-icon>  <h4 mat-line>{{folder.name}}</h4>  <p mat-line class="demo-2"> {{folder.updated}} </p>  <mat-divider [inset]="true" \*ngIf="!last"></mat-divider>  </mat-list-item> |

**How create a Node API:**

**Always Use small case letters to create anything. Like app name, file names, folders, etc**

1. Create the folder using “mkdir <<Service Name>>”
2. Now go to the newly created folder “cd <<Service Name>>”
3. Now create package.json file using “npm init -y”
4. In this project we want to use Ecma Script 6 features and NODE.JS is not yet fully supported to ES 6 and for that we need to use a compiler which will convert ES6 code to ES 5 and that is “**babel**”
5. Let’s install these dependencies using “**npm install - -save-dev babel-cli babel-preset-env**”, ‘**-dev**’ means we need these dependencies while development only.

**Note: Don’t copy paste the above command, use double dash (-) before save**

1. To get babel running we need to add a “babel.rc” file in the services root folder that is our current folder. So to do this use “**new-item .babelrc**” and add the below code.

|  |
| --- |
|  |

1. To keep babel running in the background we need to install a tool called babel watch which will run in the background and will check for changes and on any change it’ll update automatically. Use “**npm install --save-dev babel-watch**”
2. Now set babel watch at config level so that whenever we run command npm dev we our babel watch should also run automatically without manually initiating it and listens for code changes in server.js. For this add the below code in package.json 🡪 inside 1st curly braces 🡪 “scripts” 🡪

|  |
| --- |
| “dev”: “babel-watch server.js” |

Server.js is the file where we kill keep the code for server implementation.

1. Now add the next dependency library and the important one “**express JS**” as we will use express as middleware in our application, using “**npm install express**”.
2. Now install **mongoose** library which will help our services to talk to our Mongo DB, using “**npm install mongoose**”
3. Now install **cors** library which will help in cors situations, using “**npm install cors”**, right now for this app we have mongo DB in another server.
4. Now create server.js file using “**new-item server.js**”
5. Add the below code and type “<http://localhost:4000/>” in the browser to check the service.

|  |
| --- |
| import express from 'express';  const app = express();  app.get('/', (req, res) => res.send('Hello World!!!'));  app.listen(4000, () => console.log('Express server running on port 4000')); |

1. Now we need to write some more code, please refer server.js file.
2. As we use Mongo DB in an object-oriented way, lets add a folder for models using “**mkdir models**”, now go to that folder using “**cd models**”. Now add our model file user.js using “**new-item user.js**”.
3. Now go back to our root folder using “**cd..**”.
4. Sample code to insert a record into the mongo DN collection is

|  |
| --- |
| {  Name: “John”,  Age: 45  } |

1. Then use postman to check the remaining HTTP requests.
2. **While post you need to put all the properties of the object**.

**How to connect Angular 6 Project with Node APIs:**

1. Now add the service in our Angular application and to do so, go to the Angular project in terminal and then add the service using command “**ng g s User**” where **g** stands for generate and **s** stands service.
2. Now add the below code in app.module.ts,

|  |
| --- |
| * In the very first import add   **import { HttpClientModule } from '@angular/common/http';**   * Make this as the last import statement, where UserService is your service class name which is there in service.ts file   **import { UserService } from './user.service';**   * In the NgModule 🡪 imports add the **HttpClientModule** andin the providers add **UserService** |

1. Jasdkjsd

* Add user.model.ts in UI project 🡪 src 🡪 app.