**CREATE NODEJS PROJECT**

1. npm init -y
2. Install all necessary modules
3. npm i ***module name***

|  |  |
| --- | --- |
| **NAME** | **DESCRIPTION** |
| express | Framework |
| morgan | Logging request details |
| cors | In other words, ***CORS*** is a browser security feature that restricts cross-origin HTTP requests with other servers and specifies which domains access your resources. |
| reflect-metadata | ***reflect-metadata*** Allows you to do runtime reflection on types.  The native (non reflect-metadata) version of type inference is much poorer than reflect-metadata and consists only of ***typeof*** and ***instanceof.*** |
| bcryptjs | This module enables storing of passwords as hashed passwords instead of plaintext. |
| cookie-parser | ***cookie-parser*** will parse the Cookie header and handle cookie separation and encoding, maybe even decrypt it! |
| dot-env | The ***dotenv*** is a zero-dependency module that loads environment variables from a *.env* file into *process.env.* Storing configuration in the environment separate from code is based on the Twelve-Factor App methodology |
| ejs | ***EJS (Embedded JavaScript Templating)*** is one of the most popular template engines for JavaScript. As the name suggests, it lets us embed JavaScript code in a template language that is then used to generate HTML. |
| JWT - jsonwebtoken | ***JSON Web Token*** is an open standard for securely transferring data within parties using a JSON object. JWT is used for stateless authentication mechanisms for users and providers, this means maintaining session is on the client-side instead of storing sessions on the server. |
| Express-session | The ***express-session*** module provides a method and properties that can set and get the values from the session. Express Sessions are used in a Node js web application to maintain the state of a user. |

1. If you are using typescript
   1. npm i typescript -D Install typescript as development dependency
   2. npx tsc –init tsc: Typescript container- Create file to tell compiler how to work
   3. Go to tsconfig.json file and search for:
      1. rootDir – uncomment and set as: “./src” or name folder source files
      2. outDir – uncomment and set name to where all compiled files are going to be
   4. npx tsc
2. Install module to compile automatically (nodemon, ts node)
   1. npm i ts-node-dev -D install as dependency of development
3. Go to package.json file and set in scripts section:

"dev":"ts-node-dev --respawn src/index.ts"

And use npm run dev

1. Install more modules
   1. npm i express morgan cors
   2. npm i @types/express @types/morgan @types/cors -D Add data types to typescript
   3. npm install typeorm --save Install typeORM
      1. npm install reflect-metadata –save
      2. npm install @types/node --save-dev
   4. npm install pg --save Install database in this case PostgreSQL
   5. In tsconfig.json uncomment – This is complementary of **reflect-metadata** "emitDecoratorMetadata": true, "experimentalDecorators": true
2. Setting to typeorm
   1. In tsconfig.json uncomment and set as false

"strictPropertyInitialization": false

**ROUTES**

|  |  |
| --- | --- |
| **NodeJs pure** | **Typescript** |
| *const* express = require('express')  *const* router = express.Router()  router.get('/',(*req*,*res*)=>{      res.render('index')  })  *module*.*exports*= router | import {Router} from 'express'  *const* router = Router()  router.get('/',(*req*,*res*)=>{      res.send('index')  })  export default router |
| app.use('/',require('./routes/all.routes')) | import testRoutes from './routes/all.routes'  app.use(testRoutes) |