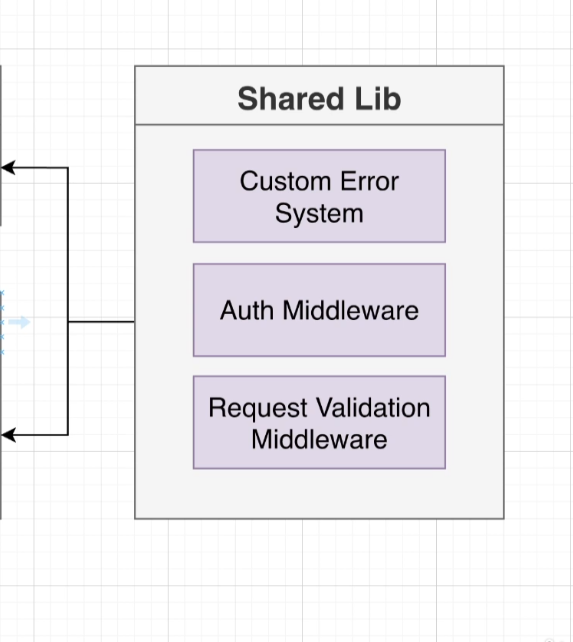
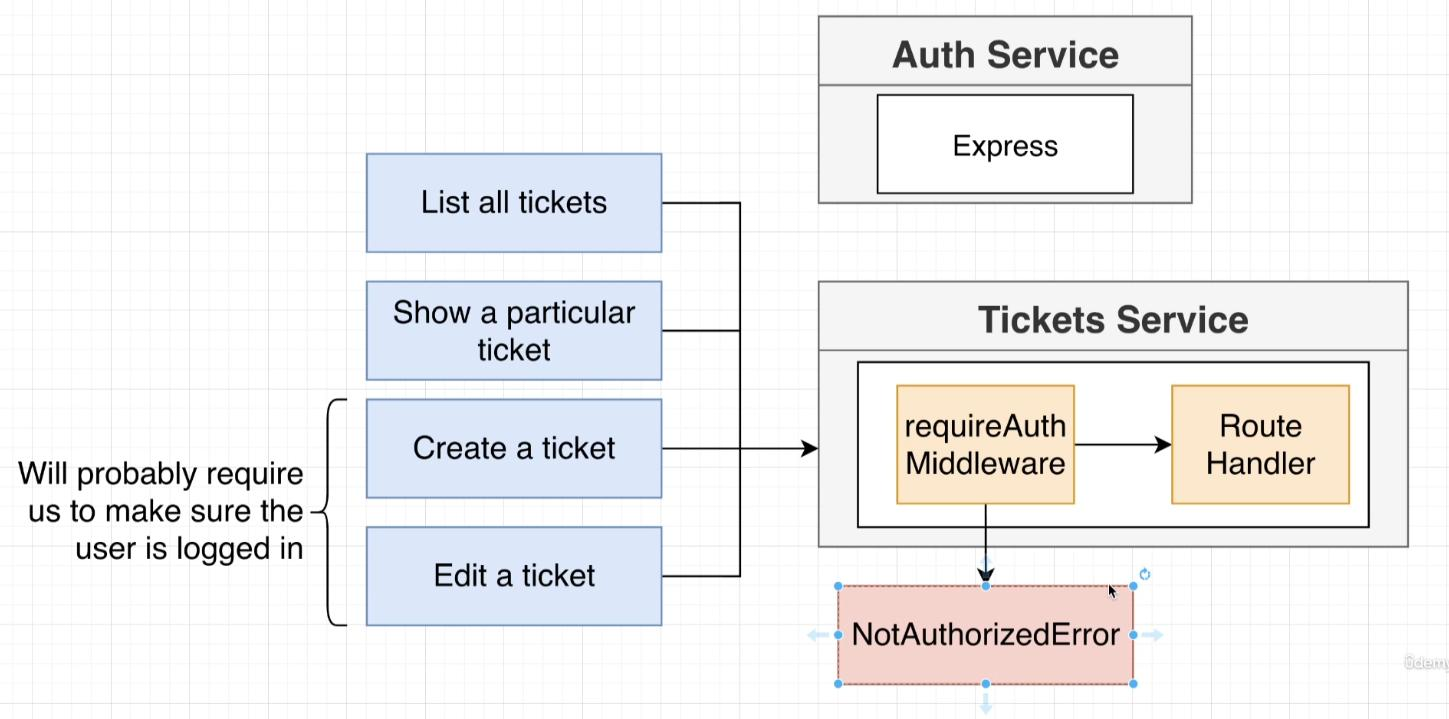
Sharing Resouces

The concept of sharing resouces comes from the compacting code stratergy , that’s how we can use the same code to other organizations or for other projects without writing it again.

For this project



We require Custom Error System, Auth Middleware , Request Validation Middleware to be used from Auth Service to Tickets Service again

So there are certian methods to share codes

1. (NOT RECOMENDED) Direct Copy Paste
2. (CAN BE RECOMENDED) Use Common Git Repository(Git submodule) for uploading Code and sharing
3. (RECOMENDED) Use NPM PACKAGE publish for NPM Registry eg-> gittix-common @ 0.0.1

NPM Publish Security Options

NPM Public Registry

1. Globaly to all npm users ( No Extra money )
2. Public only in public organization ( No Extra money )
3. Public only in private organization ( Extra money )

NPM Private Registry ( Extra money )

1. Only available privately for indivitual and to other how you give the direct access.

For this Project we will use Public Organization with Public Publish

Steps to follow for publish

1. Signup for npmjs.com
2. Goto profile -> Add organization
3. Enter Name -> sgtickets-ticketing
4. Skip if you are not using private organization or else invite via email id
5. Now create an common folder and put all the code inside you want to share
6. Now run -> npm init -y ( for initializing package.json )
7. Now change the package.json configrations
8. Change Name with format -> @OrganizationName/PackageName eg-> @sgtickets-ticketing/common
9. Change Version accordingly 1.0.1
10. Npm publishing require git to organize code.
11. -> git init
12. -> git add .
13. -> git commit -m “initial Commit”
14. Now Login to Npm
15. -> npm login
16. Now Publish Package to npm
17. -> npm publish --access public (forget to --access public then will treat as private)
18. Now for our project we want to use typescript flexibly in case of versions or in case where no typescript environment will be used in other projects. So we will Write Common Code in Typescript but will convert it to JavaScript before publishing.
19. tsc will do the job of conversion and also keep type defination file
20. For tsc to work install typescript globally --> sudo npm install -g typescript
21. initialize tsc --> tsc --init
22. Configure tsconfig
23. uncomment declaration and outDir
24. change outDir to “./build” and declaration will help to conserve type defination of code
25. now install typescript and del-cli in development env (del-cli is used to delete files specified in cli)
26. -> npm install typescript del-cli --save-dev
27. Now add some file in src -> index.ts
28. Now clean and run --> del-cli ./build/\* && tsc
29. Now configure the package.json file again
30. **“main”** : ” **./build/index.js** ” (it will be main file we will be looking for while importing lib)
31. **“types”** : ” **./build/index.s.ts** “ (it will be main type defination file )
32. **“files”** : [ “ **build/\*\*/\*** “ ] (it will be the all files we want to make sure to be published)
33. Now for version change use -> npm patch version
34. To wrap up all commands in one script add “pub” in Script
35. **“clean”** : ” **del-cli ./build/\*** ”
36. **“build”** : ” **npm run clean && tsc**”
37. **“pub”** : **”git add . && git commit -m \“Updates\” && npm version patch && npm run build && npm publish --access public”**
38. **-->** npm run pub
39. Now move error and middleware folders from **auth -> src** directory to **common -> src**
40. Now inside common directory install
41. --> npm install express express-validator jsonwebtoken cookie-session @types/cookie-session @types/jsonwebtoken @types/jsonwebtoken
42. These libraries must be installed in common directory so that errors and middleware can work
43. Now there are two ways we can import these errors and middlewares

a. import {BadRequestError} from ‘@sgtickets-ticketing/common/errors/bad-request-error

b. import {BadRequestError} from ‘@sgtickets-ticketing/common’

bth way is more convinent thus user not require to memorize directory tree for importing.

bth way can be achived by exporting all Methods from common **index.ts** file

export \* from "./errors/bad-request";

export \* from "./errors/custom-error";

export \* from "./errors/database-connection-error";

export \* from "./errors/not-found-error";

export \* from "./errors/request-validation-error";

export \* from "./errors/unauthentic-request-error";

export \* from "./middlewares/current-user";

export \* from "./middlewares/error-handler";

export \* from "./middlewares/requireAuth";

export \* from "./middlewares/validate-request-handler";

1. Now install this library ( @sgtickets-ticketing/common ) to auth directory
2. -> npm install @sgtickets-ticketing/common
3. -> npm install @sgtickets-ticketing/common (incase we update our common package)
4. Now import all the Methods by going in files of auth from ‘@sgtickets-ticketing/common’
5. Eg-> import {BadRequestError} from ‘@sgtickets-ticketing/common’