Пути можно не менять. По дефолту будет D:\CatsProject

1. Заходишь в container
2. Удаляешь node\_modules и package\_lock.json
3. npm install --force
4. npm run build:prod
5. Заходишь в build
6. Зайти в C:/Program Files/Git/bin/bash.sh, командами cd добраться до папки build проекта, ввести ./jenkins\_build\_modules.sh – запуск скрипта билда модулей фронтенда.
7. Заходишь в server
8. Изменяешь содержимое src/app.ts на

import express from 'express';

import path from 'path';

import { createProxyMiddleware, Filter, Options, RequestHandler } from 'http-proxy-middleware';

import \* as modules from './modules.json';

const app = express();

const port = 3000;

const targetDomain = "https://localhost/LMPlatform.UI";

//const targetDomain = "http://localhost:2021";

//const targetDomain = "http://host27072020.of.by";

app.use(express.static(path.resolve('/.temp/apps')))

const allowedExt = [

'.js',

'gif',

'.ico',

'.css',

'.png',

'.jpg',

'.woff2',

'.woff',

'.ttf',

'.svg',

'.mp4',

]

function getModule(url: string) {

const module = url.split('/')[1]

return modules[module]

}

const proxyServiceOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/subject/Services': 'Services', // rewrite path

'^/course/Services': 'Services', // rewrite path

'^/diplom/Services': 'Services', // rewrite path

'^/libBook/Services': 'Services', // rewrite path

'^/Services': 'Services', // rewrite path

},

secure: false,

}

const proxyAccountOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/Account': 'Account', // rewrite path

},

secure: false,

}

const proxyTestPassingOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/TestPassing': 'TestPassing', // rewrite path

},

secure: false,

}

const proxyTestOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/Tests': 'Tests', // rewrite path

},

secure: false,

}

const proxyApigOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/course/api': 'api', // rewrite path

'^/diplom/api': 'api', // rewrite path

'^/subject/api': 'api', // rewrite path

'^/libBook/api': 'api', // rewrite path

},

secure: false,

}

const proxyAdmingOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/Administration': 'Administration', // rewrite path

},

secure: false,

}

const proxyProfileOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/Profile': 'Profile', // rewrite path

},

secure: false,

}

const proxySubjectOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/subject/Subject': 'Subject', // rewrite path

},

}

const proxyStatisticOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/subject/Statistic': 'Statistic', // rewrite path

},

secure: false,

}

const proxyElasticSearchOptions = {

target: targetDomain,

changeOrigin: true,

pathRewrite: {

'^/ElasticSearch': 'ElasticSearch', // rewrite path

},

secure: false,

}

const proxyChatOptions = {

target: 'https://localhost:4200/',

changeOrigin: true,

pathRewrite: {

'^/catService': '/ChatApi', // rewrite path

},

secure: false,

}

const proxySignalROptions = {

target: 'https://localhost:4200/',

pathRewrite: {

'^/chatSignalR': '/chat',

},

secure: false,

}

app.use('\*/chatSignalR/\*', createProxyMiddleware(proxySignalROptions))

app.use('\*/catService/\*', createProxyMiddleware(proxyChatOptions))

app.use('\*/Services/\*', createProxyMiddleware(proxyServiceOptions))

app.use('\*/Account/\*', createProxyMiddleware(proxyAccountOptions))

app.use('\*/Profile/\*', createProxyMiddleware(proxyProfileOptions))

app.use('\*/TestPassing/\*', createProxyMiddleware(proxyTestPassingOptions))

app.use('\*/Tests/\*', createProxyMiddleware(proxyTestOptions))

app.use('\*/api/\*', createProxyMiddleware(proxyApigOptions))

app.use('\*/Administration/\*', createProxyMiddleware(proxyAdmingOptions))

app.use('\*/subject/Subject/\*', createProxyMiddleware(proxySubjectOptions))

app.use('\*/subject/Statistic/\*', createProxyMiddleware(proxyStatisticOptions))

app.use('\*/ElasticSearch/\*', createProxyMiddleware(proxyElasticSearchOptions))

app.get('\*', (req, res) => {

const url = req.url

const module = getModule(url)

console.log('url', url)

const setModule = module === undefined ? modules['web'] : module

const modulePath = setModule.path

const entryPoint = setModule.entryPoint

console.log(setModule, modulePath, entryPoint, req.url)

if (allowedExt.filter((ext) => url.indexOf(ext) > 0).length > 0) {

console.log(1)

// start убирает дубли с урла (временное решение)

let updatedPath

const splitModulePath = modulePath.split('/').filter(Boolean)

const splitReq = req.url.split('/').filter(Boolean)

const common = [...splitModulePath]

splitReq.map((value) => {

if (!common.includes(value)) {

common.push(value)

}

})

updatedPath = common.join('/')

// end

res.sendFile(path.resolve(updatedPath))

res.setHeader('Cache-Control', 'max-age=3153600')

} else {

console.log(2)

res.sendFile(path.resolve(`${modulePath}/${entryPoint}`))

}

})

app.listen(port, () => {

return console.log(`server is listening on ${port}`)

})

1. npm install --force
2. npm run watch
3. Запускаешь бэк
4. В браузере localhost:3000