WHO ARE YOU
PRAKTIKA II

Azureko URL: https://whoareyafull.aimarmanci14.eus/

Egileak:
Aimar Mancisidor
Ainhitze Ituarte

AURKIBIDEA

SARRERA

Who Are Ya futbolari baten izena asmatzeko jokoa da. Orokorrean “Quién es quién” delako jokoaren antzekoa da, baina galderak egin ordez, adibideetan oinarritzen da. Honetarako, futbolarien datuak kudeatzeko datubase bat behar da.

Proiektu honetarako, WhoAreYou aplikazioaren backend-eko zatia implementatu dugu, non bertan administratzaileak diren erabiltzaileak sartu daitezkeen, bai eta eginkizun desberdinak egin ere.

Proiektu hau kudeatzeko, administratzailea izango den erabiltzaile bat izango dugu datu basean. Honez gain, jokorako beharrezko jokalari, talde eta ligen informazio guztia izango dugu gordeta.

Informazio guzti hau zuzen lortu ahal izateko hasieran, APIetatik datuak deskargatuko ditugu, hauek gure zerbitzarian sartzea posible izan dezagun eta honela APIen dependenteak izan ez gaitezen. APIetan aldaketaren bat egon ezkero, ez genuke inongo arazorik izango. Aipatu beharra dago genuen players.json fitxategiko jokalariak id enumerazio bat jarraitzen dutela, ez datorrena bat api-football-ean dagoenarekin. Hori kontuan hartu dugu gutxi gora-behera eta jokalarien leagueId atributua aldatu dugu api-football-ekin bat etortzeko (datubasean ligek id hauek dituzte). Ez ditugu aldatu, ordea, jokalarien teamId-ak.

Behin beharrezko datu guztiak zerbitzarian ongi sartuta izan ondoren, sortu dugun backend orrira sartu gaitezke edozein datu aldatu edo sortzera.

Orri honetan, jokalariak sortu, editatu, ezabatu… ditzazkegu.

Proiektu honen lehenengo ariketa Githubera igotakoaren backend karpeta barruko app.js-n dago egina. Bigarren ariketatik aurrera (berezko praktika izango litzatekena), backend karpetako app karpetan dago kokatuta. Bigarren zati hau da gure makinara igo duguna atala.

1-BEHARREZKO FITXATEGIAK LORTZEN

1.1.1 Ariketa

1. fs.mkdirSync metodoa modu sinkrono batean sortzeko direktorio bat da. Errekurtsiboki sortzen du direktorioa. Gainera, metodoak recursive false izango balitz, undefined itzuliko luke, baina true jarrita direktorioaren path-a itzultzen du.

2. fs.createWriteStream metodoak adierazitako fitxategian idazteko edo sortzeko ahalmena ematen du (stream esaten zaio ingelesez) eta adierazitako fitxategia ez bada jada existitzen, sortu egiten du. Kasu honetan json/leagues/ direktorioan liga bakoitzeko izena duen .png bat sortuko du.

3. Balio bakoitzak honakoa adierazten du:

> 302: Found→ Eskatutako baliabideak URL horretara izan dira mugituta tenporalki.
>
> 401: Unauthorized→ Eskaera ez da exekutatu baimen nahikoak ez daudelako eskatutako baliabidearentzat.
>
> 429: Too Many Requests→ Erabiltzaileak eskaera gehiegi egin ditu denbora zehatz batean.
>
> 500: Internal Server Error→ Zerbitzariak espero ez zen errore bat izan du eta ezin du erabiltzaileak egindako eskaera bete.

4. res.body.pipe metodoak idazpen fluxu bat irakurtze fluxu bati esleitzen dio. Beste hitz batzuetan, metodo honen bidez body-n duguna zehaztutako fitxategian idatzi ahal izango dugu.

1.2.1 Ariketa

1. Herrialde guztien ikurrak lortzeko, 1.1.1 ariketan erabilitako kode bera hartu dugu, baina oraingoan sortutako url-an herrialde bakoitzaren ikurra gorde dezan zehaztu diogu, readFileSync egitean “nationalities.txt” aldatuz. Writepath-a ere aldatu diogu, nationalities path-ean idatzi dadin.

2. Aurreko kodearekin ez dugu behar izan funtzio berezirik hori egiteko. Izan ere, replace egitean hutsuneak kendu ditugu bai eta karaktere berezia zuen izna bat aldatu ere(costa de marfil esaterako).

1.3.1 Ariketa

1.Talde guztien identifikatzailea lortzeko komandoa honakoa da:

```
jq -er 'map(.teamId) | .[]' players.json | Sort-Object {[int]$_} -Unique > teamIDs.txt
```

2. Talde guztien ikurrak lortzeko, 1.1.1 ariketan erabilitako kode bera hartu dugu, baina oraingoan sortutako url-an talde bakoitzaren ikurra gorde dezan zehaztu diogu, readFileSync egitean “teams.txt” aldatuz. Writepath-a ere aldatu diogu, teams path-ean idatzi dadin.

BONUS POINT:

Hiru metodoak elkartu ahal izateko, what izeneko aldagai bat sortu dugu, non ‘leagues’, ‘teams’ edo ‘nationalities’ balioa pasako zaion, hiru metodoetatik nahi duguna egin dezan:

```
let download = function(what){
    let data = []
    let writepath =''
    if(what=='leagues'){
        writepath = 'json/leagues'
        data = fs.readFileSync('leagues.txt', 'utf8').split("\n")
    }
    else if(what=='teams'){
        writepath = 'json/teams'
        data = fs.readFileSync('teamIDs.txt', 'utf8').split("\n")

    } else if (what=='nationalities'){
        writepath = 'json/nationalities'
        data = fs.readFileSync('nationalities.txt', 'utf8').split("\n")
    } else {
        console.log('Error: unknown download type')
    }
    fs.mkdirSync(writepath, {recursive:true})
    data.forEach( (elem, idx) => {
        elem=elem.replace(/[\u0000-\u001F\u007F-\u009F]/g, "")
        if (what === 'leagues') {
            //const url = `https://playfootball.games/media/competitions/${elem}.png`
            const url = `https://aimarmanci14.eus/football/leagues/${elem}.png`
            fetch(url)
                .then(res => {
                    // check status
                    if (res.status === 200) {
                        res.body.pipe(fs.createWriteStream(`${writepath}/${elem}.png`))
                    } else {
                        console.log(`status: ${res.status} line: ${idx} elem:${elem} not found`)
                    }
                })
                .catch(err => console.log(err))
        } else if (what === 'teams') {
            //const url = `https://cdn.sportmonks.com/images/soccer/teams/${elem % 32}/${elem}.png`
            const url = `https://aimarmanci14.eus/football/teams/${elem}.png`
            fetch(url)
                .then(res => {
                    // check status
                    if (res.status === 200) {
                        res.body.pipe(fs.createWriteStream(`${writepath}/${elem}.png`))
                    } else {
                        console.log(`status: ${res.status} line: ${idx} elem:${elem} not found`)
                    }
                })
                .catch(err => console.log(err))
        } else if (what === 'nationalities') {
            let url = ''
            console.log(elem)
            if (elem == "C ┼ te d'Ivoire") {
                url = `https://aimarmanci14.eus/football/nationalities/Cte dIvoire.svg`
            }
            else{
                url = `https://aimarmanci14.eus/football/nationalities/${elem}.svg`
            }
            fetch(url)
                .then(res => {
                    // check status
                    if (res.status === 200) {
                        res.body.pipe(fs.createWriteStream(`${writepath}/${elem}.svg`))
                    } else {
                        console.log(`status: ${res.status} line: ${idx} elem:${elem} not found`)
                    }
                })
                .catch(err => console.log(err))
        }
    })
}
```

1.4.1 Ariketa

Jokalarien ID-ak lortzeko honako jq hau egin dugu, id guztiak txt batean gordez:

```
jq -er 'sort_by(.id) | map(.id) | .[]' players.json | Sort -Unique > playerIDs.txt
```

Jokalari guztien ikurrak lortzeko, 1.1.1 ariketan erabilitako kode bera hartu dugu, baina oraingoan sortutako url-an jokalari bakoitzaren ikurra gorde dezan zehaztu diogu, readFileSync egitean “playerIDs.txt” aldatuz. Writepath-a ere aldatu diogu, players path-ean idatzi dadin. Honez gain, url-a gure zerbitzariko domeinuaz ordezkatu dugu. Honetarako Aimarren zerbitzaria hartu dugu eta url-a sortzean helbide hori ezarri dugu.

Ondoren, setIniterval bat ezarri dugu denbora adierazteko, hau da, amaieran 500ms idatzi dizkiogu, segunduro 10 eskaera egin ahal izateko.

2-APIA SORTZEN

2.4 Ariketa

Jokalari berriak sortu ahal izateko formulario bat sortu dugu. Get ruta bat sortu dugu formulario hau inprimatu eta jokalari berriaren datuak idatzi ahal izateko. Jarraian, post ruta bat sortu dugu, non formulariotik jasotako datuak hartu eta aztertu ditugun. Express-validator erabiliz emandako datuak hutsik ez daudela frogatu ditugu eta emandako id-a duen jokalaririk ez egotea konprobatu dugu. Guztia ongi badago, datubasera sartuko ditugu datu guztiak (hasiera batean kontsolan erakustea genuen, ariketan esan bezala, baina ondorengo ariketak egin ondoren orain datubasera igotzen du jokalaria).

4-TALDEEN INFORMAZIOA LORTZEN

4.1 Ariketa

LaLigako taldeen identifikatzailea lortzeko get ruta bat sortu dugu. Bertan 1.3.1 ariketan lortutako id bakoitzerako request bat sortu dugu. Honen bidez, uneko id-a duen talde bat dagoeneko datu basean dagoen ikusi dugu, dagoeneko baldin badago, sortuta dagoelaren mezua azalduko zaigu. Bestela, taldea txertatuko du datubasean.

Big5 bost liga horien informazioa lortzeko, berriz, aurreko berdina egin dugu. Kasu honetan, requesta sortzean, id-a begiratu beharrean liga hori ea datubasean dagoeneko dagoen begiratzen dugu, bestela, txertatu egiten dugu.

5-CRUD

5.1 Ariketa

Lau endpointak kudeatzeko ruta desberdinak sortu ditugu:

-Jokalarien datuak bistaratzeko, get ruta bat sortu dugu. Jokalaria bilatu eta informazio guztia pantailaratzea adierazi diogu.

-Jokalarien datuak ezabatzeko (remove), get ruta bat sortu dugu. Jokalaria id-aren bidez bilatu eta ezabatzeko adierazi diogu.

-Jokalariak gehitzeko (add), bi ruta ezarri ditugu. Lehenengoan get egin dugu, eta honen bidez addPlayer izeneko formulario batera bidaliko digu, jokalarien datuak ezartzeko. Jarraian post bat ezarri dugu. Formulariotik lortutako datu guztiak aztertu eta hutsik ez daudela ikusi dugu, bai eta datuak zuzen daudela ere. Ondoren, emandako jokalari berria dagoeneko datubasean ez dagoela konprobatu dugu (eta beste mota batzuetako egiaztapenak ere egin ditugu, sartutako leagueId “leagues” collectionean egotea adibidez). Egon ezean, jokalaria datubasean txertatuko da, bestela, jadanik badagoelaren mezua azalduko zaigu.

-Jokalariak editatzeko (edit), hiru ruta ezarri ditugu. Lehenengoan get egin dugu, eta honen bidez editPlayer izeneko formulario batera bidaliko digu. Formulario honetan uneko jokalariaren datu guztiak aurkituko ditugu, bai eta editatzeko aukera ere. Behin hau editatuta, post ruta bat erabiliko dugu ezarritako datu berriak konprobatzeko, hau da, hutsik ez daudela ikusteko, bai eta guztiak ongi ezarrita daudela ziurtatzeko. Arazorik egon ezkero, horren mezu bat azalduko zaigu, bestela, jokalariaren datuak eguneratuko dira datubasean. Post ruta bat ere egin dugu, multer bidez jasotako irudia tratatzeko (oharretan komentatzen dugu gehiago post ruta bat egin behar izanaren arrazoia).

6-ERABILTZAILEEN KUDEAKETA

6.1 Ariketa

Erabiltzaileak erregistratzeko bi ruta ezarri ditugu. Lehenengoa get izango da, zeinek register izeneko formulario batera eramango digun. Behin erregistratzeko datu guztiak ezarri eta erregistratzeko eman ondoren, post ruta izango dugu. Bigarrengo honen bidez, emandako datu guztiak konprobatuko ditugu, hutsa ez izatetik datuak egokiak direla ikusi arte. Guztia zuzen dagoela ikusi ondoren, erabiltzaile berri bat sortuko dugu datu horiekin, user role-a duena.

Login egiteko berriz, bi ruta ere erabili ditugu. Lehenengoan get ruta izango dugu, zeinek erabiltzailearen email-a aztertuko duen. Datubasean “admin” motako erabiltzaile bat izateko defektuz, sortu dugun erabiltzaile bat (`"baba@baba.com", pasahitza 1234 da`), bere “role”-a admin izatera pasako dugu. Honetarako, honako metodoa erabili dugu:

```
users.users.update({email:"baba@baba.com"},{$set:{type:"admin"}}
```

Jarraian, post ruta izango dugu. Honek datu basean bilatuko du email eta pasahitz hori duen erabiltzailea. Datuak ongi egon ezkero, orri printzipalera eramango gaitu, beti ere admin bada.

AIPATZEKO OHARRAK

Atal honetan proiektua egiten ari ginela izandako hainbat arazo edo komentatu beharreko hainbat gauza azalduko ditugu:

Lehenik, aipatu beharra dago sortutako edozein user admin izatera pasa behar dela backend orrian, logeatu ahal izateko. Honetarako, admin den erabiltzailea web orrian sartzean, aukera bat izango dugu sortutako erabiltzaile berriak admin izatera pasatzeko, edo user rolera aldatzeko. Hau egiteko, beste formulario bat sortu dugu, ezarritako botoi baten bidez honera joan eta bertan erabiltzaile guztiak zerrendatuta izango ditugu. Zerrendako baten bat aldatu nahi izan ezkero, bi botoi izango ditugu, batek erabiltzailearen role-a user-era aldatuko du eta besteak admin-era.

Jokalariak bilatzeko aukera ere ezarri dugu. Bilatzaile bat azalduko da orri printzipalean eta bertatik edozein jokalari bilatu dezakegu ondoren edozein eginkizun egiteko.

Beste alde batetik, proiektua egitean, enuntziatuak LaLigako teams batzuk lortzeko eskatzen zuen, baina gure proiektuak ongi egin zezan, teams guztiak eskuratu ditugu, bai eta beste ligakoak ere, adibidez Bundesligakoak.

Hauez gain, datubasea hasieratu daiteke, eta honela hasieran genituen datu berberak izango genituzke, administratzaileek egindako aldaketak ezabatuz. Honek funtzionatu dezan botoi bat ezarri dugu, zeinek reset egiten duen eta datubasea hasieratzen duen.

Orokorrean enuntziatuak eskatzen zituen atal guztiak ongi egitea lortu dugu. Baina arazo bat izan dugula ere aipatu beharra dugu. Multer erabiltzean jokalariaren argazkia aukeratzeko, add egitean guztiak ongi egiten du, baina edit egitean arazoak izan ditugu eta azkenean put ruta erabiliz jokalariaren datuak aldatzea egin dugu. Argazkia aldatzeko berriz, post ruta baten bidez egin behar izan dugu.