



Praktika: Who Are You?

Juanan Pereira




Web Sistemak. 2022-2023

Who Are Ya? ²⁷



Awesome




VIRGIL VAN DIJK



DF

31




DANILO



DF

31

MIKEL OYARZABAL



FW

25 ↑

Gaien aurkibidea

1	Sarrera eta jokoaren arauak	1
1.1	Nola jokatu?	1
1.2	Xehetasunak	1
1.3	Noiz aldatzen da jokalaria misteriotsua?	1
2	Jokalarien profilak lortu	2
3	Txapelketak lortu	3
3.1	Ariketak	4
4	API deiak modu errazean probatzen	5
5	Txapelketa baten taldeak lortu	6
5.1	Atributu berriak sartu	6
5.2	Atributu baten izena aldatu	7
5.3	Atributu baten balioak aldatu	7
5.4	Ariketa	7
6	Jokalaria misteriotsua sortzen	7
6.1	Ariketak	7
7	Jokalariak asmatzen	8
7.1	Ariketak	8
8	ComboBox - Aukera zerrenda	10
8.1	Ariketak	12
9	Saiakerak eta Game Over	13
9.1	Ariketak	13
10	Estatistikak	14
10.1	Ariketak	15
11	Jokalaria misteriotsuaren bilaketa hobetzen	16
11.1	Ariketak	16
12	Berriro jokatu nahi?	16
12.1	Ariketak	17
13	Hautazko ariketak	17

1 Sarrera eta jokoaren arauak

Who Are Ya¹ futbolari baten izena asmatzeko jokia da. Orokorrean "Quién es quién" delako jokoaren antzekoa da, baina galderak egin ordez, adibideetan oinarritzen da.

1.1 Nola jokatu?

Probatu futbolari buruzko zure ezagutzak eta asmatu futbolaria 8 saialditan. Asmatu beharreko jokalarik misterioitsuak The Big Five Europako ligaren batean jokatzen du.

Jokoa hastean makinak pentsatu duen joklariaren irudi difuminatu bat bistaratuko da. Helburua, joklari hori zein den asmatzea da. Ez fidatu irudiaren koloreez manipulaturik baitaude.

Jokoak feedback-a emango dizu saiakera bakoitzaren ondoren, makinak pentsatu duen joklaria eta zurea alderatuz.

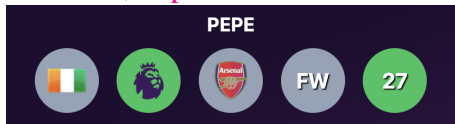
1.2 Xehetasunak

Jolasten hasteko, hasi idazten futbolari baten izena. Joklariaren izenaren bi letra bakarrik sartzean, letra horiek dituzten joklariaren zerrenda irekiko da. Bertan, futbolari baten izenean klik egin dezakezu zure ustea baieztatzeko.

Asmatzen duzunean, joklariaren xehetasunak beren irudi lausotuaren azpian kargatuko dira (5 ezaugarriko ilara bat osatuz).

Ezaugarri horietakoren bat ondo asmatu baduzu, berdez koloreztatuko da. Ezezkoan, grisez. Horrez gain, joklariaren adina ez baduzu asmatu, aplikazioak joklari misterioitsua zaharragoa edo gazteagoa den ere esango dizu.

Adibidez, **Pepe** izena sartu eta honako emaitza jasotzen baduzu:

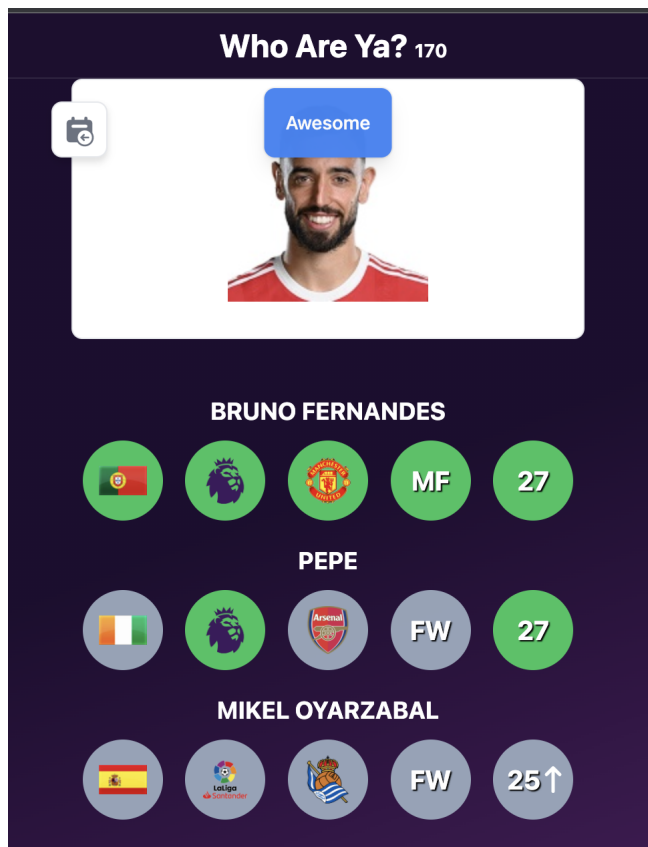


Joklari misterioitsua ez dela Boli Kostakoa, Premier League-n bai jokatzen duela, baina ez Arsenal taldean. Joklaria ez da aurrelaria baina 27 urte bai dituela esan nahi du.

1.3 Noiz aldatzen da joklari misterioitsua?

Puzzle berri bat lor dezakezu erlojuak 12am jotzen duenean zure eskualdean. Partida amaitzen duzunean estatistikak ikusiko dituzu. Estatistika leihok berean ikusi ahalko duzu zenbat denbora geratzen den hurrengo puzzlea egiteko.

¹<https://missing11.com/who-are-ya/>; berez WhoAreYou izan beharko litzateke, baina slang moduan dago idatzia



Jokoa programatzeko jarraitu beharreko argibideak jasoko dituzu praktika honetan. Atal bakoitzean eskatzen diren funtzioak programatu eta probatu beharko dituzu. Zure kodearen bertsio bat GitHubeko proiektu (repo) pribatu batera igo beharko duzu. Nahi adina *commit* egin ditzakezu baina atal bakoitzeko *Git Tag* bat sortu beharko duzu ^a.

^aEz baduzu inoiz *Git Tag* bat sortu edo ez badakizu zer den ezta ere, lasai, horren inguruko argibideak ere izango dituzu. Momentuz, has zaitez dokumentu hau irakurtzen: <https://git-scm.com/book/en/v2/Git-Basics-Tagging>

2 Jokalarien profilak lortu

WhoAreYa jokoak berez ordainpeko API bat erabiltzen badu ere, atal honen helburua doako API bat ([football-data.org](https://www.football-data.org)) erabiliz behar ditugun datuak nola lor genezake ikastea da. Hurrengo ataletan (3-6) dauden ariketak bukatu ostean WhoAreYa jokoak erabiltzen dituen datu berberak lortuko ditugu (alde batetik demostratuz APIak erabiltzen badakigula eta bestetik jokoak behar dituen datu guztiak doan atera daitezkeela).

Beraz, [football-data.org](https://www.football-data.org) webgunea erabiliko dugu jokalarien datuak lortzeko. Doako API honek dokumentazio on bat eskaintzen du horretarako <https://www.football-data.org/documentation/quickstart> webgunean. Berriz, APIa bera, api.football-data.org URLan aurkituko dugu. Bertan izena eman eta API Token bat lortuko duzu. Token horrekin HTTP eskaerak egin ahalko ditugu.

3 Txapelketak lortu

Erabiliko ditugun Futbol txapelketak “Big 5 leagues” deritzonak dira, zehazki:

Premier League (Ingalaterrako futbol liga garrantzitsuenak), La Liga (Espainakoa), Bundesliga (Alemaniakoa), Serie A (Italiakoa) eta Ligue 1 (Frantziakoa).

Txapelketa guztiak hemen aurkituko ditugu, JSON formatuan: <http://api.football-data.org/v4/competitions> JSON erantzun hori parseatu egin beharko da. Nola? *fetch()* agindua² Chrome DevTools-etik erabili dezakegu:

```
fetch('http://api.football-data.org/v4/competitions')
  .then(
    r => r.json()
  )
  .then(
    data => {
      console.log(data)
    }
  )
```

CORS arazoak?

Kontsolan CORS arazoak jasotzen badituzu, DevTools hemengo orritik exekutatu (alegia, sar zaitez webgune honetan eta bertan zaudelarik ireki DevTools eta fetch komandoa landu): <http://api.football-data.org/v4/competitions>

Bertan, 163 txapelketen datuak zehazten direla aurkituko dugu. **Gorde ezazu** JSON erantzuna `competitions.json` izeneko fitxategi batean. JSON fitxategia parseatzeko `jq` (JSON Query) tresna erabiliko dugu.

JQ nola instalatu

`jq` tresna instalatu egin beharko duzu.

- Linux-en: `apt install jq`
- macOS-en: `brew install jq` ^a
- Windows-en: `choco install jq` ^b

`jq` erabiltzeko, terminal bat ireki behar duzu IntelliJ-n. Windows-en bazaude eta `cat` komandoa ez badoa -PowerShell terminal batean badabil, baina cmd terminal batean agian ez-, `type` komandoa erabili.

^aEz baduzu `brew` pakete kudeatzailea macOS-en instalatu, orain da momentu egokia, argibide hauek jarraituz: <https://brew.sh/>

^bEz baduzu `choco` pakete kudeatzailea Windows-en instalatu, orain da momentu egokia, argibide hauek jarraituz: <https://chocolatey.org/install>

Gorde duzun JSON fitxategi horretan `data` objektua aurkituko duzu. Bertan, `competitions` arraya. Jarraian, array horren elementu baten adibidea azter dezakegu. Ohar zaitez `name` atributuan kompetizio edo ligaren izena agertzen dela:

²Ikus: <https://developer.mozilla.org/en-US/docs/Web/API/Response/json>

```
$ jq -e '.competitions[] | select(.id == 2014)' competitions.json
{
  "id": 2014,
  "area": {
    "id": 2224,
    "name": "Spain",
    "code": "ESP",
    "flag": "https://crests.football-data.org/760.svg"
  },
  "name": "Primera Division",
  "code": "PD",
  "type": "LEAGUE",
  "emblem": "https://crests.football-data.org/PD.png",
  "plan": "TIER_ONE",
  "currentSeason": {
    "id": 1504,
    "startDate": "2022-08-14",
    "endDate": "2023-06-04",
    "currentMatchday": 4,
    "winner": null
  },
  "numberOfAvailableSeasons": 92,
  "lastUpdated": "2022-03-20T09:20:08Z"
}
```

Adi

Windows eta PowerShell erabiltzen baduzu, baliteke kakotx bikoitzak ez onartzea jq agindu batean, eta horrelako errore mezuren bat jasotzea:

jq: error: syntax error, unexpected INVALID_CHARACTER (Windows cmd shell quoting issues?)

Horrela bada, kakotx bikoitzak eskapatu (*escape the double quotes*). Alegia, `"` idatzi ordez, `\` idatzi.

3.1 Ariketak

1. JQ erabili dugu JSON fitxategitik 2014 ID-a duen elementua lortzeko. Jarraian aurkituko dituzun ariketetan gauza bera egitea eskatzen zaizu, baina oraingo honetan JavaScript erabiliz. Has gaitezen lehenengo ariketarekin: *fetch* funtzioaz JSON fitxategia lortu eta jarraian filtratu id=2014 duen objektua. Argibidea: *filter()* metodoa erabili dezakezu.
2. Gure hurrengo helburua, Big5 ligak iragaztea izango da. Lehenengo eta behin, soilik TIER_ONE (maila goreneko ligak) aterako ditugu.

```
$ jq -e '.competitions[] | select(.plan == "TIER_ONE")' competitions.json
...
(13 emaitza lortu behar dira).
...

$ jq -e ' [.competitions[] | select(.plan == "TIER_ONE")] | length '
competitions.json
13
```

Gauza bera egin, baina oraingo honetan JavaScript erabiliz (*fetch* komandoaz JSON fitxategia lortu, jarraian filtratu TIER_ONE ligak)

3. Zein izango litzateke Espainako ligak lortzeko kodea? (JS erabiliz)
4. Zein izango litzateke ESP, DEU, ENG, ITA, FRA herrialdetako TIER_ONE ligak lortzeko kodea? (JS erabiliz)
5. Aurreko ariketan arazo bat gertatuko da: emaitzan nahi ez dugun (Big5-ekoa ez den) liga bat sartu da (Championship izenekoa).

```
0: {id: 2016, area: {}, name: 'Championship', code: 'ELC', type: 'LEAGUE', ...}
1: {id: 2021, area: {}, name: 'Premier League', code: 'PL', type: 'LEAGUE', ...}
2: {id: 2015, area: {}, name: 'Ligue 1', code: 'FL1', type: 'LEAGUE', ...}
3: {id: 2002, area: {}, name: 'Bundesliga', code: 'BL1', type: 'LEAGUE', ...}
4: {id: 2019, area: {}, name: 'Serie A', code: 'SA', type: 'LEAGUE', ...}
5: {id: 2014, area: {}, name: 'Primera Division', code: 'PD', type: 'LEAGUE', ...}
length: 6
```

Nola moldatu aurreko ariketaren *fetch* komandoa Championship liga emaitzetatik kentzeko?

6. *map()* komandoa erabili aurreko *fetch* emaitzatik ligen ID-ak lortzeko. Emaitza hau izan beharko litzateke: [2021, 2015, 2002, 2019, 2014]

4 API deiak modu errazean probatzen

API batek eskaintzen dituen deiak modu errazean ulertzen hasteko IntelliJ IDEak tresna berezi bat eskaintzen du: *HTTP Client*.

IntelliJ aplikazioan, **Tools / HTTP Client / Create Request in HTTP Client** aukeratuko dugu. Jarraian, datu hauek sartuko ditugu, LaLigan jokatzeko duen Real Sociedad taldearen jokalariei lortzeko.

```
GET https://api.football-data.org/v4/teams/90
Accept: application/json
X-Auth-Token: ZURE_TOKENA
```



Adi

api.football-data.org webguneko APIa erabiltzeko TOKEN bat beharko duzu, eta azken hau lortzeko football-data.org webgunean erregistratu beharko zara (doako erregistroa da).

Orain **Run all requests in file** aukeratu. Emaitza JSON fitxategi batean gordeko da, automatikoki *.idea/httpRequests* karpeta barruan. HTTP eskaera guztiak **Scratches and Consoles / Scratches** karpetan gordeko dira.

5 Txapelketa baten taldeak lortu

Behin txapelketen IDak jaso ondoren, txapelketa zehatz batean parte hartzen duten taldeen izenak jaso ditzakegu.

Adibidez, Premier League ligaren IDa 2021 dela jakinda, bertan parte hartzen duten taldeen izenak horrela lortuko genituzke (emaitza `premier.json` fitxategi batean gorde):

```
GET https://api.football-data.org/v4/competitions/2021/teams
```

Bertan, team bakoitzeko, bere jokalaririk (squad) lor daitezke baita ere. Adibidez, `jq` erabiliz, Arsenal taldearen jokalaririk lortzeko:

```
jq -e '.teams[] | select( .name == "Arsenal FC")' premier.json
```

Eta soilik jokalaririk izenak nahiko bagenu:

```
jq -r '.teams[] | select( .name == "Arsenal FC") | .squad[].name' premier.json
```

Arsenal taldearen lehenengo jokalaririk datuak lortzeko:

```
jq -r '.teams[] | select( .name == "Arsenal FC") | .squad[1]' premier.json
```

Bertan ikusten dugun formatua hau da:

```
{
  "id": 5530,
  "name": "Aaron Ramsdale",
  "position": "Goalkeeper",
  "dateOfBirth": "1998-05-14",
  "nationality": "England"
}
```

Baina gure jokorako behar dugun formatua beste hau da:

```
{
  "id": 186932,
  "name": "Mikel Oyarzabal",
  "birthdate": "1997-04-21",
  "nationality": "Spain",
  "leagueId": 140,
  "teamId": 548,
  "position": "FW"
},
```

Ikusten dugun bezala, atributu batzuen izenak aldatu beharko ditugu (`dateOfBirth:birthDate`), beste berri batzuk sartu (`leagueId`, `teamId`) eta atributu baten balioa moldatu (`Goalkeeper:GK`)

5.1 Atributu berriak sartu

Talde bakoitzaren ID-a lortu (`.id`), ligaren ID-a lortu (2021, Premier League kasurako) eta taldekide bakoitzari esleituko diogu

```
jq -r '.teams[] | .squad[] += { "teamId" : .id, "leagueId": 2021}' premier.json
```

Eta soilik jokalaririk datuak nahi baditugu:

```
jq -r '[_teams[] | .squad[] += { "teamId" : .id, "leagueId": 2021} | .squad | flatten[] ]' premier.json
```


5.2 Atributu baten izena aldatu

dateOfBirth atributua birthDate izatea nahi dugu:

```
jq -r '.teams[] | .squad[] += { "teamId" : .id, "leagueId": 2021} | .squad |
  flatten[] | with_entries(if .key == "dateOfBirth" then .key = "birthDate"
    else . end)' premier.json
```

5.3 Atributu baten balioak aldatu

Aurreko guztiaz gain, atributu hauen balioak aldatu nahi ditugu:

Offence -> FW

GoalKeeper -> GK

Midfield -> MF

Defence -> DF

Hurrengo kodean soilik Offence eta Midfield aldaketak bistaratzen dira (beste biak begibizta-koak baitira, patroia jarraituz).

```
jq -r '.teams[] | .squad[] += { "teamId" : .id, "leagueId": 2021} | .squad |
  flatten[] | with_entries(if .key == "dateOfBirth" then .key = "birthDate"
    else . end) | with_entries(if .key == "position" and .value=="Offence" then
    .value="FW" else if .key == "position" and .value=="Midfield" then .value
    ="MF" else . end end)' premier.json
```

5.4 Ariketa

1. Jatorrizko `premiere.json` fitxategitik abiatuta, jq-rekin egindako aldaketa guztiak JavaScript erabiliz egin itzazu.

6 Jokalari misterioak sortzen

Doako API bat erabiliz behar ditugun JSON guztiak sor daitezkeela ikusi ondoren, atal honetatik aurrera WhoAreYa jokoaren programazioarekin hasiko gara.

Deskargatu `fullplayers.json` eta `solution.json` fitxategiak. Lehenengo fitxategian (`fullplayers.json`) Big5 ligatan (Premiere, Bundesliga, LaLiga, League 1, Serie A) jokatzen duten jokalariek guztien datuak agertzen dira: jokalariaen identifikatzailea, izena, jaiotze-data, herritartasun, taldearen identifikatzailea, posizioa zelaian, elastikoaren zenbakia eta ligaren identifikatzailea.

Bigarren fitxategian (`solution.json`), jokalariek batzuen identifikatzaileak array batean gordetzen dira. Array horretan dauden jokalariaiek eguneko soluzioak izango dira. Nola kalkulatu da egungo soluzioa? Arraya horrela indexatuz: `arraya[differenceInDays(new Date("08-19-2022"))-1]`.

6.1 Ariketak

1. `differenceInDays(oinarria)` funtzioa programatu. Funtzio honek data bat jasoko du parametro gisa³ (`oinarria`) eta gaur eta `oinarria` arteko diferentzia egunetan itzuliko du. Adibidez, Martxoaren 1, 2022 pasatzen badiogu, eta gaur 2022ko irailak 1 bada, 185 zenbakia **itzuli**

³Date motako objektu bat

beharko luke. Berriz, 2022ko Abuztuaren 18a pasatzen badiogu eta gaur 2022ko irailak 2 bada, 16 zenbakia **itzuli beharko luke**⁴ (bi kasuetan azkenengo eguna ere kontatzen da)

2. `fetchJSON(file)` funtzioa inplementatu. Funtzio honek parametro gisa zein JSON fitxategi irakurri nahi den jaso eta fitxategi horren edukia itzuliko du JSON objektu batean.
3. Kargatu `fullplayers.json` eta `solution.json`⁵. Inplementatu `getSolution(players, solutionArray, differenceInDays)` izeneko funtzioa. Funtzio honek hiru parametro jasoko ditu (jokalarien JSON arraya, soluzioen JSON arraya, gaur eta 2022/08/18 dataren artean egun kopurua) eta gaurko egunari dagokion jokalaria objektua itzuliko du. Adibidez, gaur irailak 2 bada, distantzia 16 egunekoa da eta beraz, `solutions[16-1]` kalkulatzeko soluzioa 159208 id-a duen jokalaria izango litzateke (jokalaria izena: Danilo) eta funtzioak jokalaria hori itzuliko du (objektu osoa, JSON formatuan)



Aurreko ariketan jokalaria objektua itzultzeaz gain interesgarria litzateke kontsolan ere bistaratzea, horrela garatzaileak argibide on bat izango du jakiteko ea kodea ondo programatuta dagoen.

4. Fijatu zaitez 1. irudian. Bertan jokalaria aurpegia lausotuta (*blurred*) dago. Zeintzuk dira `main.html` fitxategian (`id="mystery"` duen geruzan) egin beharreko ezabaketak irudia lausotu gabe agertzeko? (ezin da ezer **gehitu** fitxategiari, soilik estiloak **ezabatuz** lor daiteke). Ariketa honen soluzioa *unblur* izeneko branch batean gorde zure GitHubeko proiektuan.



Aurreko ariketak egin ondoren, GitHubeko *whoareyou* proiektutik `milestone1` zip fitxategia jaitsi^a, eta bertan dauden funtzioak bete itzazu (3 funtzio daude **YOUR CODE HERE** komentarioekin, `loaders.js#fetchJSON`, `main.js#differenceInDays` eta `main.js#getSolution`). Dena ondo egin eta programatu baduzu, 1. irudian dagoen emaitza jaso beharko zenuke.

^a<https://github.com/juananpe/whoareyou/releases/tag/milestone1>

7 Jokalariak asmatzen

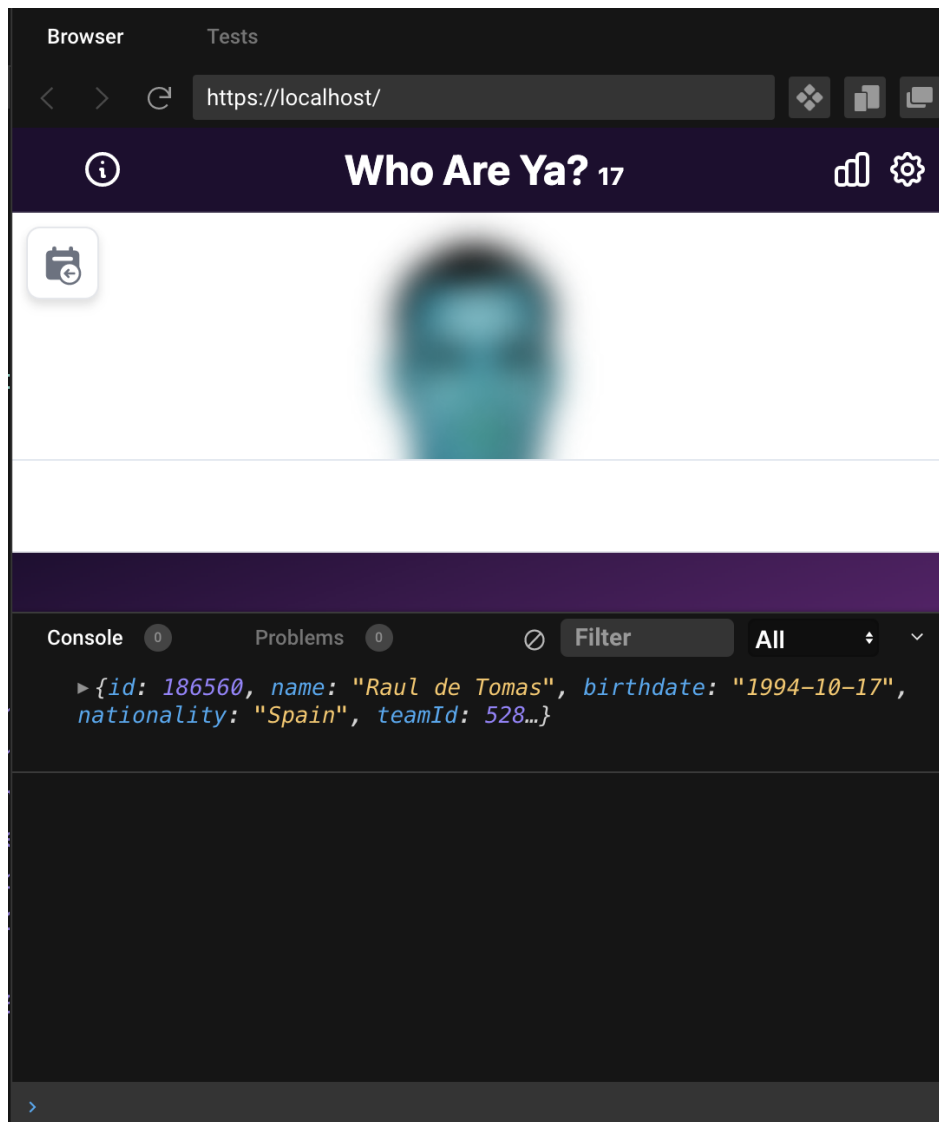
Behin ordenagailuak jokalaria misteriotsua pentsatu ondoren, erabiltzaileak jokalaria hori asmatu beharko du, asko jota 8 saiakeretan. Aplikazioak asmatutako jokalaria bakoitzaren atributuak (adina, liga, postua...) misteriotsuaren atributuen kontra erkatu beharko ditu. Horretarako, hainbat funtzio programatu beharko ditugu datozen ariketetan.

7.1 Ariketak

1. `js/rows.js` fitxategian `getAge(dateString)` funtzioa programatu. Parametroan jokalaria baten jaiotze data bat jasoko du, string arrunt gisa (urtea-hilabetea-eguna formatuan) eta emaitza gisa jokalaria horren adina emango du. Adibidez, `getAge('1999-01-14')` funtzioaren emaitza 23 izango da.

⁴<https://www.timeanddate.com/date/durationresult.html?d1=18&m1=8&y1=2022&d2=2&m2=9&y2=2022&ti=on>

⁵Ikus: <https://github.com/juananpe/whoareyou/blob/main/js/main.js#L29>



1. irudia: Milestone1 ariketak bete ondoren, soluzioa horrela ikusi beharko litzateke nabigatzailean. Ohar zaitez kontsolan jokalaria misterioitsuaren datuak agertzen direla, programatzeko laguntza gisa ondo etorriko zaigu... baina bukatzean ez ahaztu log hori ezabatzeaz.

2. *check(theKey, theValue)* funtzioa programatu. Bi parametro jasoko ditu, theKey eta theValue. theKey erabiltzaileak pentsatu duen jokalaria baten edozein atributu izan daiteke eta theValue, atributu horren balioa. Jokalaria misterioitsuaren balioarekin bat egiten badu, “correct” itzuliko du, bestela “incorrect”. Jaiotze-data bat bada (*birthdate*), adin berekoak badira “correct” itzuliko du. Misterioitsuaren adina handiagoa bada, “higher”, eta txikiagoa bada, “lower”.

Adibidez, demagun jokalaria misterioitsuaren datuak hauek direla, besteak beste:
name: 'Declan Rice', birthdate: '1999-01-14', nationality: 'England', teamId: 1

Orduan:

`check('nationality', 'Spain')` deiak 'incorrect' itzuliko luke (jokalaria misterioitsua ez delako Espainakoa)

`check('birthdate', '1998-01-01')` deiak 'lower' itzuliko luke (jokalaria misterioitsua gazteagoa delako)

Gauza bera gainontzeko atributu guztiekin (**OHARRA**: soluzio orokor bat eskatzen da, alegia, jokalaria atributu berri bat sartzen badiogu, *check* funtzioak ondo funtzionatzen jarraitu beharko luke inolako aldaketarik egin beharrik gabe)

Gogoratu jokalaria misteriot suaren datuak *game* izeneko datu egituran daudela, *solution* atributuan.

3. *getPlayer(playerId)* funtzioa programatu. Funtzio honek jokalaria baten IDa jaso eta jokalaria horri dagokion JSON objektua itzuliko du (*game.players* datu egitura begiratuta)
4. *leagueToFlag(leagueId)* funtzioa programatu. *leagueId* zenbaki bat jaso eta dagokion string-a itzuliko du. Zehazki: 564 -> es1, 8 -> en1, 82 -> de1, 384 -> it1, 301 -> fr1. **Garrantzitsua** (ariketa interesgarria egiteko): EZIN DIRA **if** edo **switch** klausulak erabili.
5. *stringToHTML* funtzioa horrela definituta dago:

```
const stringToHTML = (str) => {  
  var parser = new DOMParser();  
  var doc = parser.parseFromString(str, 'text/html');  
  return doc.body  
};
```

Sar ezazu funtzio hori *fragments.js* fitxategian eta esportatu. Jarraian *rows.js* fitxategitik inportatu.

6. *rows.js* fitxategian *setupRows* funtzioa esportatu (ohar zaitez maila goreneko funtzio bat dela, alegia funtzio bat itzultzen duen funtzioa). *main.js* fitxategian *setupRows* funtzioa inportatu. Jarraian, *main.js* fitxategian, *myInput* testu eremuan erabiltzaileak jokalaria baten zenbakia sartu (momentuz zenbakiekin egingo dugu lan, aurrerago moldatuko dugu izenekin lan egiteko) eta 'Enter' tekla sakatzen badu, *setupRows* itzultzen duen funtzioari deituko diogu. Adibidez:

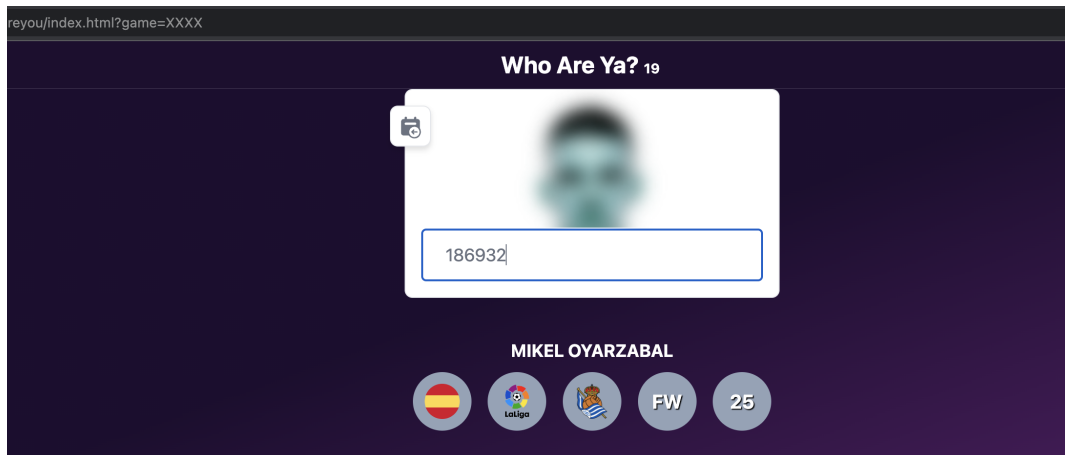
```
let addRow = setupRows( /* parametro bat beharko duzu hemen */ );  
...  
addRow(jokalariarenIDa);  
...
```

Dena ondo badoa, animazio sinple bat ikusiko duzu, jokalaria atributuak bistaratuz (berdez, misteriot suarekin bat egiten dutenak, grisez gainoinzekoak. Ikus 2. irudia)

7. Deskargatu *milestone2* GitHubetik eta bete itzazu *YOURCODEHERE* zuriuneak esandako funtzioekin.

8 ComboBox - Aukera zerrenda

Orain arte, erabiltzaileak jokalaria IDa idatzi behar du *myInput* testu eremuan. Eredu hori aukera zerrenda bihurtuko dugu (combobox bat). Horretarako, *main.js* fitxategitik zati hau ezabatu egingo dugu:



2. irudia: **Milestone2** ariketak bete ondoren, soluzioa horrela ikusi beharko litzateke nabigatzailean.

```
let addRow = setupRows(...);
...
...onkeydown = function (event) {...}
```

Horren ordean, funtzio honi deituko diogu

```
autocomplete(document.getElementById("myInput"), game)
```

Funtzio hori **autocomplete.js** fitxategitik inportatu beharko duzu.

Jarraian, **autocomplete.js** fitxategia moldatuko dugu *autocomplete combobox* bat inplementatu ahal izateko. Gure kodea, **hemengo erantzunan** oinarrituta dago.

Zer egiten du? Laburrean, hauek dira funtzioaren zereginak (kodea jarraituz):

1. **myInput** testu eremuan *eventListener* bat sortzen du, erabiltzaileak bertan teklatzen duena (jokalari baten izena) atzementeko.
2. **a** izeneko objetuan jokalarien izenak bistartzeko edukiontzi orokor bat prestatzen du
3. erabiltzaileak teklatu dituen karaktereak jaso eta jokalaria bakoitzeko funtzioak jokalaria baten izenarekin bat egiten duten egiaztatzen du.
4. Baiezkoan (jokalari baten izenean agertzen dira erabiltzaileak teklatu dituen karaktereak), **b** objektu bat sortzen du (*div* geruza bat) jokalaria baten izenarekin (eskubian) eta bere taldearen ikurrarekin (ezkerrean). Bat egin duten karaktereak beltzez margotzen ditu.
5. **b** objektuari listener bat esleitzen dio: jokalaria horren gainean klik egitean *addRow* metodoari deitzekeo
6. *combobox*-a teklatuarekin ere erabili daiteke. Hori lortzeko *keydown listener* bat esleitzen dio **b** objektuari
7. bukatzeko, erabiltzaileak *combobox*-aren kanpo klik egiten badu, aukera-zerrenda itxi egin behar da (*closeAllLists*)

8.1 Ariketak

Gogoratu `Milestone3` fitxategietan⁶ `YOUR CODE HERE` string-a bilatu behar duzula eta jarraian ariketetan eskatzen dena bete.

1. *Autocomplete* funtzioan (`autocomplete.js` fitxategian) falta diren hiru lerro bete itzazu. Lehenengoan: Jokalariaren izena erabiltzaileak tekleatu dituen hizkiekin hasten den kalkulatzen duen kodea.
2. Bigarrenean, jokalariaren izenaren barruan, erabiltzaileak tekleatu duen azpikatea beltzez margotu behar da.
3. Hirugarrenean, *addRows* funtzioari deitu erabiltzaileak combobox-ean hautatu duen jokalariaren ID-a pasatuz.
4. **Jokalariaren** adinaren kasua ez dugu guztiz bukatu. Jokoan, jokalaria misterioitsuaren adina ez badugu asmatzen, guk pentsatutako adina baino handiagoa edo txikiagoa den ere esaten digu. Oraitxe bertan ez da horrelakorik gertatzen. Ikus 2. irudia: bertan 25 zenbakia agertzen da Mikel Oyarzabal-en kasurako. Eta grisez ikusten dugu zenbaki hori. Baina jatorrizko jokoan, horrez gain, misterioitsuaren adina handiagoa (zaharragoa) edo txikiagoa (gazteagoa) den esan behar digu.

`fragments.js` fitxategian bi konstante hauek txertatu eta esportatu:

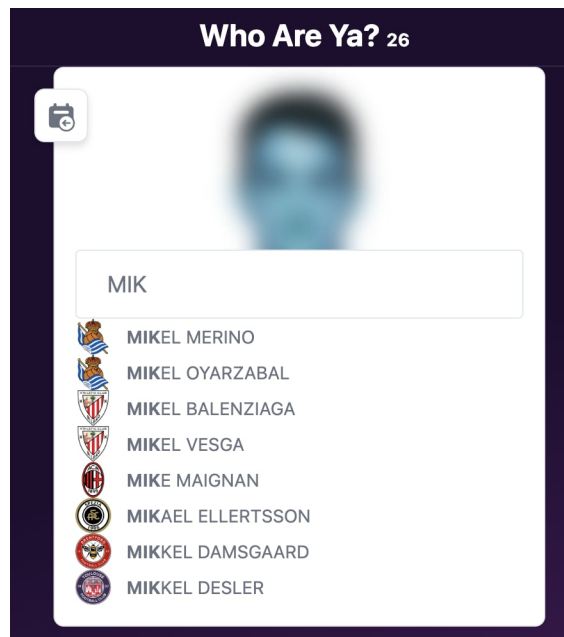
```
const higher = <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"
  fill="white" aria-hidden="true" width="25" style="margin-right: -8px;
  margin-left: -3px;"><path fill-rule="evenodd" d="M5.293 7.707a1 1 0
  010-1.41414-4a1 1 0 011.414 014 4a1 1 0 01-1.414 1.414L11 5.414V17a1 1
  0 11-2 0V5.414L6.707 7.707a1 1 0 01-1.414 0z" clip-rule="evenodd"></
  path></svg>;

const lower = <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"
  fill="white" aria-hidden="true" width="25" style="margin-right: -8px;
  margin-left: -3px;"><path fill-rule="evenodd" d="M14.707 12.293a1 1 0
  010 1.4141-4 4a1 1 0 01-1.414 01-4-4a1 1 0 111.414-1.414L9 14.586V3a1
  1 0 012 0v11.586L12.293-2.293a1 1 0 011.414 0z" clip-rule="evenodd"></
  path></svg>;

Jarraian, \hlc[lightgray]{rows.js} fitxategian setContent funtzioa
moldatu adina kontuan izan dezan dagokion lower edo higher ikurrak
bistaratz.
```

Dena ondo badoa, 3. irudian ikusten den bezala, combobox bat izan beharko zenuke jokalari baten izenaren hasierako hizkiak idazten hastean. Jokalaria hautatutakoan, bere atributuak misterioitsuaren atributuekin erkatu eta bistaratu beharko lirateke (eta adina, grisez edo berdez margotzeaz gain, gezi bat gora edo behera agertuko da - adina ez baduzu asmatzen -). Ohar zaitez asmatutako hizkiak beltzez agertzen direla izen guztietan. Bertan arazo bat dago: jatorrizko jokoan bilatutako hizkiak izenaren edozein tokitan egon daitezke (ez soilik aurrizki gisa). Hau konpondu beharko dugu...

⁶<https://github.com/juananpe/whoareyou/releases/tag/milestone3>



3. irudia: `Milestone3` ariketak bete ondoren, soluzioa horrela ikusi beharko litzateke nabigatzailean.

9 Saiakerak eta Game Over

Erabiltzaileak saiakera kopuru mugatu bat dauka: 8 izen eman ondoren, ez badu asmatu, jokoa amaitu behar da. JSON objektu batean (**egoera** deituko diogu) gordeko dugu saiakera array bat (erabiltzaileak esan dituen jokalarien identifikatzaileekin) eta soluzioarekin (jokalaria misterioetsuaren ID-a).

```
{ "guesses" : [],  
  "solution": IDa }
```

JSON objektu hori `localStorage` APIa erabiliz gordeko dugu. Zehazki, `WAYgameState` izeneko aldagai batean.

9.1 Ariketak

Milestone4⁷ kode txantiloia jaitsi eta hurrengo ariketak egin.

1. `stats.js` fitxategian `initState(what, solutionId)` funtzioa programatu. Parametro gisa bi balio hartuko ditu:
what parametroan, `localStorage`-tik jaso nahi dugun objektuaren identifikatzailea.
solutionId: jokalaria misterioetsuaren identifikatzailea

Erantzun gisa **array** bat itzuliko du, bi balioekin:

state: *what* izeneko objektua `localStorage`en existitzen bada, objektu hori bueltatuko du array-aren lehenengo posizioan.

funtzio anonimoa: funtzio honek parametro bakar bat jasotzen du, *guess* izenekoa. *guess* parametroan erabiltzaileak pentsatu duen jokalaria IDa etorriko da. Hori bera aurreko paragrafoetan zehaztu den *guesses* izeneko array-an txertatuko da eta `localStorage`-n JSON objektua eguneratuko da.

⁷<https://github.com/juananpe/whoareyou/releases/tag/milestone4>

2. `rows.js` fitxategian `setupRows` funtzioan, `initState` funtzioari deitu eta `[state, updateState]` balioak jasotzen dira. Beheko aldean, `addRow` funtzioan, `updateState()` funtzioa erabiltzen da, erabiltzaileak pentsatu duen jokalaria-aren IDa `localStorage`en gordetzeko. Erabiltzaileak pentsatu dituen ID guztiak `game.guesses` array-an agertzen dira.
3. Aurreko ariketan ez da ezer programatu behar, ulertu bakarrik. Baina ulertu ondoren, `rows.js` fitxategiko `resetInput` funtzioa programatu ahalko duzu. Bertan, `myInput` testu eremuan dagoen informazioa ezabatu eta "Guess X of 8" idatzi, non X uneko saiakera zenbakia izango den (adibidez, "Guess 3 of 8")
4. `gameEnded(lastGuess)` funtzioa programatu. Honek erabiltzaileak pentsatu duen jokalaria-aren ID-a jaso eta jokoa amaitu den edo ez bueltatuko du (jokoa amaituko da `lastGuess` jokalaria misterioaren IDa bada edo 8. saiakeran ez badugu jokalaria asmatu).
5. `addRow` funtzioan elementu berriak sartu dira... Zure soluzioan kopiatu. (Besteak beste, `unblur` funtzioa).
6. `addRow` funtzioan, `success()` eta `gameOver()` funtzioei deitzen zaie, soluzioa asmatu edo jokoa kale egitean, hurrenez hurren. Funtzio hauek programatu (momentuz oso-oso antzekoak dira, soilik `unblur` funtzioari deitzen diote, dagokion parametroarekin, jokalaria-aren irudia eta izena bistaratzeko).
7. Commit and Push egitean, ez ahaztu `Milestone4` etiketa esleitzeaz.

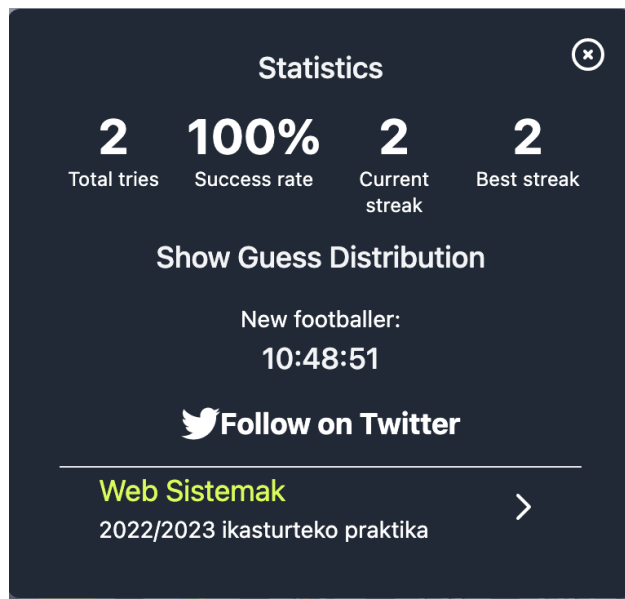
10 Estatistikak

Jokoa amaitzean estatistika leiho bat erakutsi nahi dugu (ikus 4. irudia). Bertan, besteak beste 4 zenbaki agertzen dira: Total tries, Success Rate, Current Streak, Best Streak.

- Total tries. Zenbat aldiz jokatu duzun.
- Best streak. Asmatze-bolada hoberena. Adibidez, azken 4 agertzen bada, 4 aldiz jarraian jokalaria misterioa zein den asmatu duzula (eta inoiz ez duzula hainbeste aldiz jarraian asmatu) esan nahi du.
- Current streak. Zenbat aldiz jarraian asmatu duzu zein den jokalaria misterioa azken kale egin zenuenetik.
- Success rate. Asmatze kopurua / Total Tries * 100 (portzentaia)

Datu horiek `localStorage`-n gordeko dira, horrelako JSON objektu batean:

```
{winDistribution: [0,0,0,0,0,0,0,0,0,0],
  gamesFailed: 0,
  currentStreak: 0,
  bestStreak: 0,
  totalGames: 0,
  successRate: 0
}
```

4. irudia: Jokoa amaitzean estatistikak bistaratu behar dira.

10.1 Ariketak

Aurreko datuak izanik, programa itzazu hurrengo funtzioak `stats.js` fitxategian.

Milestone5⁸ kode txantiloia jaitsi beharko duzu ariketa hauek egiteko.

(Adi: *toggle* eta *headless* funtzioak ⁹ `fragments.js` fitxategian txertatu beharko dituzu)

1. Estatistikak eramaten dituen JSON objektua jaso eta `SuccessRate` balioa itzultzen du
2. `getStats(what)` funtzioa. Parametro gisa `localStorage`en dagoen eta estatistikak gordetzen dituen JSON objektuaren izena jasoko du eta baldin badago, objektu hori bera itzuliko du. Ez badago `localStorage`en, objektua sortu, hasieratu eta itzuli egingo du.
3. `updateStats(t)` funtzioa. Funtzio honi jokoa amaitzen denean deituko zaio (bai jokoa asmatu delako - orduan $t < 8$ edo jokoa ez delako asmatu, orduan $t \geq 8$). Parametro gisa uneko saiakera kopurua jaso eta `gamestats` objektuan atributu hauek kalkulatu ditu: `totalGames`, `currentStreak`, `gamesFailed`, `windistribution`, `bestStreak`, `successRate`. Bukatzeko, `gameStats` izenarekin `localStorage`en gordeko du.
4. `rows.js` fitxategian, `updateStats` funtzioari deitu (berez, hor dagoen komentarioa kentzearekin, nahikoa izan beharko litzateke)
5. `fragments.js` fitxategian `stats` izeneko zatia esportatu¹⁰ eta `rows.js` fitxategian inportatu (`showStats` funtzioan erabiliko da)
6. `gameOver` edo `success` funtzioetatik `showStats` funtzioari deitu, estatistikak bistartzeko
7. `stats` objektuan (`fragments.js` fitxategian), ohar zaitez geruza (`div`) bat dagoela `nextPlayer` identifikatzailearekin. Bertan, hurrengo jokalaria misteriozua noiz izango den esaten digu (4. irudian ikusten da 10 ordu 48 minutu eta 51 segundu barru izango dela - hurrengo eguneko

⁸<https://github.com/juananpe/whoareyou/releases/tag/milestone5>

⁹<https://gist.github.com/juananpe/fa0102544e680fd067b68a6e8ba7f3c4>

¹⁰Hemen aurki dezakezu <https://gist.github.com/juananpe/d224386fc436fcbe1d8449b99f8f7f52>

0:0:0 ordua betetzen denean). Denbora hori segunduro eguneratzen da, setInterval metodo baten laguntzaz (ikus interval objektua rows.js fitxategian). Denbora tarte hori kalkulatzeko duen kodea programa ezazu ¹¹.

8. GitHuben `milestone5` etiketa esleitu orain arte idatzi duzun kode bertsio honi.

11 Jokalari misteriotsuaren bilaketa hobetzen

Orain arte jokalaria misteriotsuaren izena bilatzean, soilik izenaren aurrizkian agertzen diren hizkiak aztertzen ditugu. Adibidez, Oyarzabal idazten badugu, combobox-an ez da ezer agertuko, ez baitago inolako jokalaririk bere izenaren aurrizkian Oyarzabal string-a duenik. Baina jakin badakigu Mikel Oyarzabal izeneko jokalaria bat dagoela. Hori konpondu nahi dugu atal honetan (izenaren edozein lekutan bilatu nahi ditugu erabiltzaileak teklatzen dituen hizkiak).

String baten barruan beste azpistring bat dagoen (eta baldin badago, zein posiziotan agertzen den) kalkulatzeko modulu bat [hemen](#)¹² aurki dezakegu (match eta parse funtzioen laguntzaz). [Hemen](#) ikus dezakezu modulu hori exekuzioan¹³ Arazoa da modulu hori nodeJS-ko modulu bat dela eta ezin dela zuzenean gure nabigatzailean (browser-an) exekutatu. Zer egin? Zorionez, browserify izeneko tresna bat daukagu eskuragarri. Tresna horren zeregina nodeJSrako moduluak browser batean erabiltzeko bihurtzeko egitea da.

Itzulpen hori egiteko komandoaren argibideak [hemen](#) aurki ditzakezu ¹⁴.

11.1 Ariketak

Adi!

Ariketa hauek (milestone6 eta milestone7) besteak baino zailtasun maila handiagoa dute eta berauek burutzeko ez da txantiloirik eskaintzen. Ez baduzu soluzioa asmatzen lasai hartu, ez dira derrigorrezkoak gainditzeko (nota hobetzeko soilik erabiliko dira).

1. `match.js` eta `parse.js` fitxategiak sortu (browserify exekutatu beharko duzu) Nabigatzailean (browser-an) erabiltzeko prest uzteko.
2. `autocomplete.js` fitxategia moldatu ¹⁵ match eta parse erabili dezan (erabiltzaileak jokalaria baten izenean agertzen diren azpistring-ak idaztean, comboboxean jokalaria horien izenak agertu behar dira. Adibidez, “Oyar” idaztean, Mikel Oyarzabal agertu behar da comboboxean)
3. Ariketa hauek egiten badituzu, GitHuben `milestone6` etiketarekin markatu bertsioa.

12 Berriro jokatu nahi?

Demagun jokatzeko hasi eta hainbat saiakera egin ondoren (demagun 3) nabigatzailea itxi egin behar duzula. Kafetxo bat hartu eta bueltatzean jokoa berriro irekitzen baduzu, nola gustatuko litzaizuke

¹¹ Argibidea: <https://stackoverflow.com/questions/66277554/how-to-use-interval-timer-function-from-date-fns>

¹² <https://github.com/moroshko/autosuggest-highlight>

¹³ <https://codesandbox.io/s/nr994o146m?file=/src/index.js>

¹⁴ <https://stackoverflow.com/questions/28696511/require-is-not-defined-error-with-browserify/41476832#41476832>

¹⁵ Gogoratu hemen duzula argibide on bat: <https://codesandbox.io/s/nr994o146m?file=/src/index.js>

aurkitzea? Ziur asko aurrekoan egin zenituen saiakerak berreskuratu nahiko zenituzkela eta egoera horretatik abiatu (alegia, 4. saiakeratik). Edo demagun jokalaria misterioitsua asmatu egin duzula. Nabigatzailea itxi, bazkaltzera joan zara eta bazkaldu ondoren, zure lagun bati erakutsi nahi diozu nola asmatu zenuen (zeintzuk izan ziren hasierako saiakerak). Bi kasu hauei erantzuna emateko, jokoaren egoera *localStorage* gordetzeko kodea programatu dezakegu. Jokoa hasi bezain pronto, *localStorage* begiratu beharko genuke ea jadanik hasita dagoen jokoren bat dagoen edo ez. Egotekotan egoera berreskuratu eta utzi zenuen puntu horretatik hasi beharko ginateke.

12.1 Ariketak

1. Joko baten egoera gordetzeko kodea programatu. Nabigatzailea itxi eta berriro jokoaren orria irekitzean *localStorage* begiratu eta hasita dagoen jokoren bat aurkitzen duzun. Aurkitzekotan, egoera berreskuratu. Gauza bera jokoa jada bukatuta badago (erabiltzaileak 8 saiakera egin dituelako asmatu gabe edo asmatu duelako).
2. Ariketa hau burutzen baduzu, GitHuben `milestone7` etiketarekin markatu bertsioa.

13 Hautazko ariketak


1. **Favicon.ico.** Nabigatzaileak Favicon.ico izeneko irudi bat bilatuko du erlaitz bakoitzaren ondoan jartzeko. Ez badu aurkitzen, kontsolan errore bat bistaratuko du. Hori ekiditzeko sor ezazu erro katalagoan (/) `favicon.ico` irudi bat.


✖ Failed to load resource: the server responded with a status of 404 (Not Found)

5. irudia: Nabigatzaileak Favicon.ico izeneko irudi bat bilatuko du erlaitz bakoitzaren ondoan jartzeko.

2. Bukatzeko, jatorrizko jokoan ez dira soilik jokalaria bakoitzeko 5 atributu bistaratzen (herrialdea, liga, taldea, posizioa, adina), 6 baizik, **elastikoaren zenbakia** ere kontutan hartzen baita (adina-rekin bezala, handiagoa edo txikiagoa den azalduz). Informazio hori jada jokalarien JSON fitxategian dugu, beraz, nahi izanez gero, kontuan izan dezakezu eta errenkada bakoitzean bistaratu (ikus 6. irudia)


Hautazko bigarren ariketa hau burutzen baduzu, GitHuben `milestone8` etiketarekin markatu bertsioa.








GUESS 2 OF 8


MIKEL OYARZABAL


**NAT**

**LGE**

**TEAM**

**POS**

**AGE**

**SHIRT**

6. irudia: Jokalariaren elastikoaren zenbakia (*shirt number*) bistaratzea ere ondo legoke...