Proiect GlobalWaves - Etapa 3 - Analytics & Recommendations

- **Responsabili:** Robert Grancsa [mailto:robert.grancsa2002@gmail.com], Tudorache Ioana-Alina [mailto:alina.tudorache872@gmail.com], Alexandru Gheorghe [mailto:tccz658@gmail.com]
- **Colaboratori:** Sorina-Anamaria Buf [mailto:sorinabuf@gmail.com], Ștefan Cocioran [mailto:stefancocioran@gmail.com], Florian-Luis Micu [mailto:miculuis1@gmail.com]
- **Data publicarii:** 18.12.2023, ora 13:30
- **Deadline HARD:** ~~14.01.2024~~ 16.01.2024, ora 23:55
- **Ultima modificare a enuntului:**
  - 11.01.2024, ora 23:23 - added details for design patterns
  - 06.01.2024, ora 14:25 - details for updateRecommendations
-
- **Ultima modificare a testelor:** 29.12.2023, ora 14:54 - removed all songs from the library
- **Schelet:** GitHub [https://github.com/oop-pub/oop-project-2023/tree/main/etapa3]
- **History**
  - 28.12.2023, ora 02:37 - simplified complex tests (6, 15, 16) and other fixes for date, recommandation and duplicate songs
  - 27.12.2023, ora 11:05 - fixed index for random song
  - 26.12.2023, ora 14:13 - replaced topPodcasts and recalculted mostProfitableSong
  - 24.12.2023, ora 13:25 - clarifications for wrapped
  - 24.12.2023, ora 13:25 - fixed createPlaylist in test 12
  - 24.12.2023, ora 13:25 - removed field from test 10 and fixed message for notifications
  - 23.12.2023, ora 14:10 - updated notifications info
  - 23.12.2023, ora 14:10 - updated notifications messages
  - 22.12.2023, ora 19:50 - updated search specifications
  - 22.12.2023, ora 22:33 - added hosts to input
  - 19.12.2023, ora 23:11 - removed bad field from refs

- **Atenție!** Pentru partea de citire / afișare nu este necesară folosirea adnotărilor din librăria Jackson. Puteți utiliza o metodă similară cu cea prezentată de noi în schelet, și anume să vă folosiți de clasa ObjectMapper. **Nu există o metodă corectă sau greșită de a lucra cu input-ul / output-ul.** Scopul nostru în cadrul acestui proiect nu este să vă testăm abilitatea de a lucra cu fișierele, ci să vă testăm atât înțelegerea cunoștințelor de bază dobândite în cadrul cursurilor și a laboratoarelor, cât și modul de gândire într-un limbaj de programare orientat obiect.
- **Este obligatoriu să folosiți cel puțin 4 design pattern-uri.** Trebuie sa precizati in README ce design pattern ati folosit, si motivatia & locatia unde a fost folosita
- **Depunctari lipsa design pattern-uri**: Pentru fiecare design pattern lipsa (din cele 4), veti avea o depunctare de -2.5p, astfel pentru:
  - 0 design pattern-uri - **-10p**
  - 1 design pattern-uri - **-7.5p**
  - 2 design pattern-uri - **-5p**
  - 3 design pattern-uri - **-2.5p**

Dacă decideți să folosiți Generative AI (ex. ChatGPT) pentru implementare sau alte aspecte ale codului, treceți în README exact unde ați folosit această metodă.

De asemenea, scrieți în README și dacă ați folosit ca schelet rezolvarea oficială a etapei II.

Obiective

- Implementarea a cel puțin 4 design pattern POO.
- Refactorizarea codului pentru a permite adăugarea funcționalităților noi.

Descriere

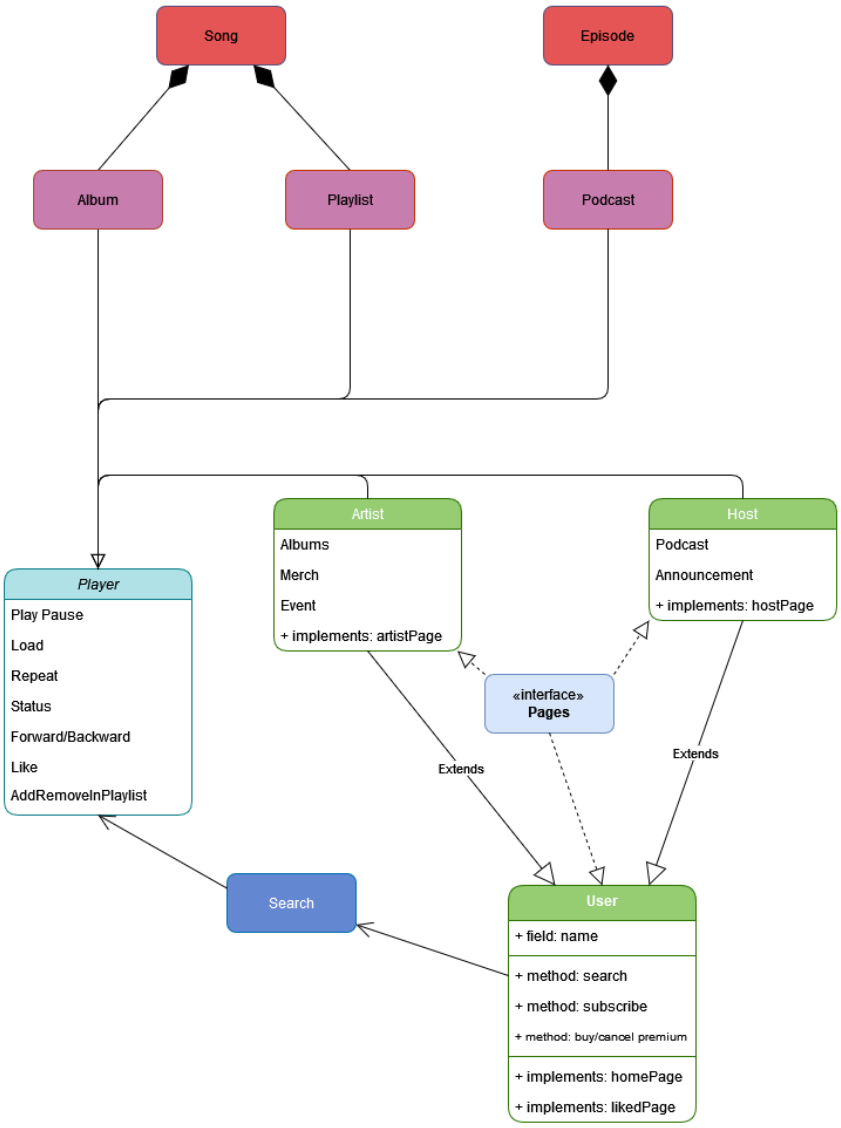
Pentru acest proiect o să aveți de implementat o aplicație asemănătoare ca funcționalități cu Spotify, simulând diferite acțiuni făcute de utilizatori, acum cu feature-uri care ne adauga detalii despre monetizare si cat de mult foloseste un user platforma noastra. Pe langa asta,

vom avea si modalitati de a adauga playlist-uri generate pe baza preferintelor userilor, sau a preferintelor fanilor de la un anumit artist. Iar pentru fiecare obiect adaugat nou pe platforma, utilizatorii vor primi notificari de la platforma cu noutatiile aparute.

Comenzile de la etapa I, etapa II și formatul input-ului și al output-ului nu se modifică. Tot ce ați implementat în primele etape, rămâne valid și în cadrul etapei curente. În această etapă va trebui să extindeți soluția de la etapa II cu funcționalități noi.

Recap de la etapele anterioare

În cadrul primelor etape, ați avut de implementat structura platformei noastre, care a constat în definirea entitatilor si obiectelor de mai jos.



Clarificari functionalitati de search:

Filterele de search **nu sunt case sensitivite**, asadar "Highway to Hell" == "Highway To Hell"

Ordinea albumelor filtrate vor fi dupa **ordinea inscrieri artistului** pe platforma, apoi dupa **ordinea adaugarii albumelor** de pe platforma

GlobalWaves Wrapped

GlobalWaves Wrapped este o statistica a platformei GlobalWaves care oferă utilizatorilor un rezumat personalizat al activității lor muzicale, inclusiv topul melodiilor ascultate, genuri preferate și alte statistici relevante, pe parcursul rularii programului.

Aceasta comanda poate fi apelată de pe orice pagină, si va afisa statistici de la timestamp 0 pana la timestamp-ul current. Se vor afisa o structura sub forma de `${nume}: ${count}`, si vor fi sortate initial dupa count, iar in caz de egalitate dupa nume lexicografic.

Un listen este definit ca orice melodie care a inceput sa ruleze pentru userul respectiv, oricat de mult a trecut din ea.

Atata timp cat nu este specificat altfel, top-urile vor consta din **primele 5 obiecte** din fiecare categorie.

Statistici pentru Useri

La rularea comenzii de catre un user, o sa se afiseze urmatoarea configuratie:

- topArtists - artistii dupa numarul de listen-uri
- topGenres - genre-urile dupa numarul de listen-uri
- topSongs - melodiile dupa numarul de listen-uri
- topAlbums - albumele dupa numarul de listen-uri
- topEpisodes - episoadele de podcast dupa numarul de listen-uri

Mesaje posibile pentru aceasta comanda:

- `"No data to show for user ${username}."`

Orice melodie stearsa, dar **deja ascultata** de user o sa ramana in statisticile lui

Daca aveti 2 melodii cu acelasi artist si aceasi nume, listen-urile lor se vor combina la afisare, chiar daca se afla in albume diferite

Input

```
{
     "command": "wrapped",
     "username": "alice22",
     "timestamp": 4000
}
```

Output

```
{
  "command": "wrapped",
  "user": "alice22",
  "timestamp": 4000,
  "result": {
    "topArtists": {
      "The Beatles": 17,
      "Ed Sheeran": 2,
      "Pharrell Williams": 2,
      "Led Zeppelin": 1
    },
    "topGenres": {
      "rock": 18,
      "pop": 4
    },
    "topSongs": {
      "Freedom": 2,
      "Happier": 2,
      "Because - Remastered 2009": 1,
      "Carry That Weight - Remastered 2009": 1,
      "Come Together - Remastered 2009": 1
    },
    "topAlbums": {
      "Abbey Road (Remastered)": 17,
      "Divide": 2,
      "G I R L": 2,
      "Led Zeppelin IV": 1
    },
    "topEpisodes": {
      "The Power of Design": 2,
      "The Science of Us": 1
    }
  }
}
```

Statistici pentru Artisti

La rularea comenzii de catre un artist, o sa se afiseze urmatoarea configuratie:

- topAlbums - albumele dupa numarul de listen-uri
- topSongs - melodiile dupa numarul de listen-uri
- topFans - userii, dupa numarul de listen-uri pentru artistul curent
- listeners - numarul useri care au ascultat artistul curent

Mesaje posibile pentru aceasta comanda:

- `"No data to show for user ${username}."`

Orice melodie stearsa, dar **deja ascultata** de user o sa ramana in statisticile lui

Input

```
{
  "command": "wrapped",
  "username": "The Beatles",
  "timestamp": 12000
}
```

Output

```
{
  "command" : "wrapped",
  "user" : "The Beatles",
  "timestamp" : 12000,
  "result" : {
    "topAlbums" : {
      "Abbey Road (Remastered)" : 34,
      "1 (Remastered)" : 27
    },
    "topSongs" : {
      "Because - Remastered 2009" : 2,
      "Carry That Weight - Remastered 2009" : 2,
      "Come Together - Remastered 2009" : 2,
      "Golden Slumbers - Remastered 2009" : 2,
      "Her Majesty - Remastered 2009" : 2
    },
    "topFans" : [ "alice22", "bob35" ],
    "listeners" : 2,
  }
}
```

Statistici pentru Host

La rularea comenzii de catre un host, o sa se afiseze urmatoarea configuratie:

- topEpisodes - episoadele dupa numarul de listen-uri
- listeners - numarul useri care au ascultat artistul curent

Mesaje posibile pentru aceasta comanda:

- `"No data to show for user ${username}."`

Orice melodie stearsa, dar **deja ascultata** de user o sa ramana in statisticile lui

Input

```
{
  "command": "wrapped",
  "username": "Guy Raz",
  "timestamp": 12000
}
```

Output

```
{
  "command" : "wrapped",
  "user" : "Guy Raz",
  "timestamp" : 12000,
  "result" : {
    "topEpisodes" : {
      "The Power of Design" : 2,
      "The Science of Us" : 1
    },
    "listeners" : 1,
  }
}
```

Monetization

Artisti de pe platforma au protestat pentru ca nu sunt platiti, asa ca introducem sectiunea de monetization. La fiecare play al melodiei, un user va plati catre artist o valoare calculata pe baza unor euristici. Pe langa asta, cumpararea de merch va transfera bani in contul artistului. La finalul fiecarea rulari se vor calcula si afisa banii pe care il primeste fiecare artist pe baza calculelor de mai jos.

Toate operatile de impartire, adunare, scadare se vor realiza cu tipul de date **Double**

Doar la afisare se va folosi rotunjire la ultimele 2 zecimale. Pentru a face asta puteti sa va folositi de formula

```
songRevenue = Math.round(value * 100.0) / 100.0
```

Formatul afisarii va fi `${artist}: ${date}`, unde datele vor avea urmatoarea forma:

- songRevenue - cati bani au scos de pe toate listen-urile de pe platforma
- merchRevenue - cati bani au scos de pe vanzarile de merch de pe platforma
- ranking - ranking-ul artistului de pe platforma dupa vanzari (indexate de la 1)
- mostProfitableSong - melodie pe care artistul a scos cei mai multi bani (bazat pe suma revenue-urilor)

Nu aveti nici un input pentru aceasta comanda, la finalul fiecarei rulari se va adauga automat acest output

Fiecare artist care a avut macar un play sau un merch cumparat o sa fie afisat in final, chiar daca are 0 income din cantece. In cazul acela, mostProfitableSong o sa fie "N/A"

Output

```
{
    "command": "endProgram",
    "result": {
      "Eminem": {
        "songRevenue": 74.2,
        "merchRevenue": 150.0,
        "ranking": 1,
        "mostProfitableSong": "Lose Yourself"
      },
      "Led Zeppelin": {
        "songRevenue": 18.2,
        "merchRevenue": 70.0,
        "ranking": 2,
        "mostProfitableSong": "Stairway to Heaven"
      },
      "...": {}
    }
}
```

Merch

Buy merch

Un utilizator poate sa cumpere merch al unui artist. Dupa cumpararea lui, artistul va primi pretul final al produsul in contul sau, care va fi adaugat in field-ul `merchRevenue`.

Userul nu are nici un wallet intern, asa ca fiecare buy va adauga revenue doar pentru artist, nu se va scadea nimic de la user.

Merch-ul poate fi cumparat doar atunci cand este pe pagina artistului.

Mesaje posibile pentru aceasta comanda:

- `"The username ${username} doesn't exist."`
- `"Cannot buy merch from this page."`
- `"The merch ${name} doesn't exist."`
- `"${username} has added new merch successfully."`

Input

```
{
  "command": "buyMerch",
  "username": "alice22",
  "name": "Metallica T-Shirt",
  "timestamp": 5
}
```

Output

```
{
  "command": "buyMerch",
  "username": "alice22",
  "timestamp": 5,
  "message": "alice22 has added new merch successfully."
}
```

See my merch

Se vor afisa toate obiectele cumparate de catre user, in ordine cronologica dupa timpul cumparari lor.

Daca obiectul a fost sters de catre artist, userul va avea in continuare acces la el.

Mesaje posibile pentru aceasta comanda:

- "The username ${username} doesn't exist."

Input

```
{
    "command": "seeMerch",
    "username": "alice22",
    "timestamp": 6
}
```

Output

```
{
    "command": "seeMerch",
    "username": "alice22",
    "timestamp": 6,
    "result": [ "Metallica T-Shirt", "AC/DC Cap", "I Am Kenough Hoodie" ]
}
```

Tipuri de utilizatori

Premium

Adaugare utilizator premium

Mesaje posibile pentru aceasta comanda:

- "The username ${username} doesn't exist."
- "${username} is already a premium user."
- "${username} bought the subscription successfully."

```
{
    "command": "buyPremium",
    "username": "alice22",
    "timestamp": 10
}
```

```
{
    "command": "buyPremium",
    "username": "alice22",
    "timestamp": 10,
    "message": "alice22 bought the subscription successfully."
}
```

Renuntare utilizator premium

La renuntarea de la premium, toate melodiile ascultate atunci cand a fost el premium vor primii banii impartiti dupa formula de mai jos

Mesaje posibile pentru aceasta comanda:

- "The username ${username} doesn't exist."
- "${username} is not a premium user."
- "${username} cancelled the subscription successfully."

```
{
    "command": "cancelPremium",
    "username": "alice22",
    "timestamp": 15
}
```

```
{
    "command": "cancelPremium",
    "username": "alice22",
    "timestamp": 15,
    "message": "alice22 cancelled the subscription successfully."
}
```

Calcul monetizare

Un user premium nu poate fi scos niciodata de pe platforma

Un utilizator poate cumpara subscriptie premium la pretul unic de **1.000.000**, care va semnifica cat credit are pentru a plati artisti. La finalul fiecarei rularii, toti utilizatori care sunt premium vor adauga bani in contul artistilor de la care a ascultat. Formula pentru a afla cat primeste fiecare artist este urmatoarea:

$$val = \frac{10^6}{song_{total}} * song_{artist}$$

unde:

- val = valoarea de bani care se duce la artistul respectiv
- $song_{total}$ = numarul de cantece ascultate de catre userul premium
- $song_{artist}$ = numarul de cantece ascultate de la artistul

Exemplu: Un user premium care a ascultat 10 melodii:

- 3 de la Led Zeppelin
- 5 de la Eminem
- 2 de la Queen

O sa aiba distributia urmatoare a baniilor:

- $10^6/10 * 3$ pentru Led Zeppelin
- $10^6/10 * 5$ pentru Eminem
- $10^6/10 * 2$ pentru Queen

Free

Ads

Pentru ca un utilizator free nu plateste deloc ca sa foloseasca aplicatia, va primi la momente predefinite ad-uri. Acestea vor fi adaugate in coada userului atunci cand va aparea comanda "adBreak" de la input. In acest caz, dupa terminarea melodiei curente, utilizatorul va avea de ascultat obligatoriu un ad care este unskippable.

Ad-ul este un cantec definite in library, si ar trebui sa functioneaze la fel ca un cantec normal doar ca nu se poate da skip la el. Totusi, un load il va suprascrie mereu.

Ad-urile nu vor contribui la spotify wrapped

Se garanteaza ca nu vor fi adaugate 2 ad-uri consecutive

Mesaje posibile pentru aceasta comanda:

- `"The username ${username} doesn't exist."`
- `"${username} is not playing any music."`
- `"Ad inserted successfully."`

```
{
  "command": "adBreak",
  "username": "alice22",
  "timestamp": 15,
  "price": 100
}
```

```
{
  "command": "adBreak",
  "username": "alice22",
  "timestamp": 15,
  "message": "Ad inserted successfully."
}
```

Calcul monetizare

Atunci cand ad-ul intra in player, se vor calcula ultimele melodii ascultate intre ad-ul curent, si ultimul ad primit (sau inceputul istoricului), si artistii ai caror melodii a ascultat acel user vor primii valoarea urmatoare:

$$val = \frac{price_{ad}}{song_{last}} * song_{artist}$$

unde:

- val = valoarea de bani care se duce la **artistul respectiv**
- $price_{ad}$ = cat valoareaza ad-ul respectiv
- $song_{last}$ = numarul de cantece ascultate de catre user de la ultimul adBreak pana la adBreak-ul curent

- $song_{artist}$ = numarul de cantece ascultate de la artist

Exemplu: Un user non-premium care a ascultat 5 melodii urmate de un ad, urmate de inca 2:

- 2 de la Led Zeppelin
- 3 de la Eminem
- Ad (cost 500)
- 2 de la Queen

O sa aiba distributia urmatoare a baniilor:

- $500/5 * 2$ pentru Led Zeppelin
- $500/5 * 3$ pentru Eminem
- 0 pentru Queen, deoarece inca nu a primit un ad care sa plateasca acele piese

Notifications

Veti avea de implementat un sistem de notificari pentru urmatoarele categorii:

- Playlist:
  - Owner-ul playlist-ului va primi notificare cand cineva se aboneaza la playlist-ul lui
- Creator de continut (Userul primeste notificare pentru orice operatie de mai jos daca este abonat la un Artist/Host):
  - Cand este adaugat un concert (actiune valida doar pentru Artist)
  - Cand este adaugat un album nou / podcast
  - Cand este adaugat merch
  - Cand este adaugat un announcement

Userul se va putea abona la oricare creator de continut, si incepand de la momentul acela el va primii notificari pentru oricare din actiunile enumerate mai sus. Notificarea pentru playlist-uri va fi activata pentru **fiecare playlist public** by default.

Subscribe

Pentru a se abona la un artist/host, utilizatorul trebuie sa se afle pe pagina artistului/host-ului. Daca utilizatorul va da de 2 ori la rand comanda subscribe pentru un artist/host, el se va dezabona de la acel user.

Mesaje posibile pentru aceasta comanda:

- `"The username ${username} doesn't exist."`
- `"To subscribe you need to be on the page of an artist or host."`
- `"${username} (subscribed/unsubscribed) (to/from) ${artist/host} successfully."`

Input

```
{
  "command": "subscribe",
  "username": "alice22",
  "timestamp": 15
}
```

Output

```
{
  "command": "subscribe",
  "username": "alice22",
  "timestamp": 15,
  "message": "alice22 subscribed to Led Zeppelin successfully."
}
```

Get Notifications

Fiecare notificare va fi declansata de unul dintre evenimentele descrise mai sus. La adaugarea unui album nou, fiecare utilizator abonat la artistul respectiv va trebui notificat cu un mesaj de forma `"New Album: New Album from ${artist}."` .

Un utilizator poate sa-si acceseze singur notificarile. O accesare afiseaza **toate notificarile**, si le **sterge**, astfel incat la urmatoarea accesare a notificarilor cele deja accesate **sa nu mai existe**.

Input

```
{
  "command": "getNotifications",
  "username": "alice22",
  "timestamp": 15
}
```

Output

```
{
  "command": "getNotifications",
  "username": "alice22",
  "timestamp": 15,
  "notificarions": [
    {
      "name": "New Album",
      "description": "New Album from Eminem."
    },
    {
      "name": "New Album",
      "description": "New Album from Ed Sheeran."
    },
    {
      "name": "New Merch",
      "description": "New Merch from The Beatles."
    }
  ]
}
```

Recomandari pentru Useri

Fiecare utilizator normal va primii recomandari pentru melodii si playlist-uri, care vor fi afisate pe HomePage-ul acestuia. Pentru a le reda, utilizatorul trebuie sa primeasca comanda de *loadRecommendations*.

Se poate ca pentru utilizatorul pentru care s-a cerut recomandari sa nu se gaseasca nicio recomandare.

Output-ul pentru printarea HomePage-ului utilizatorului va avea urmatorul format:

`"Liked songs:\n\t[songname1, songname2, …]\n\nFollowed playlists:\n\t[playlistname1, playlistname2, …]\n\nSong recommendations:\n\t[songname1, songname2, …]\n\nPlaylists recommendations:\n\t[playlistname1, playlistname2, …]"`

Exemplu :

```
{
  "user" : "melodicmind42",
  "command" : "printCurrentPage",
  "timestamp" : 150,
  "message" : "Liked songs:\n\t[Rhythmic Revolution]\n\nFollowed playlists:\n\t[]\n\nSong recommendations:\n\t[Rhythm Rise]\n\nPlaylists recommendations:\n\t[]"
}
```

Recomandarile sunt de 3 tipuri:

Random Song

- se ia melodia curenta pe care utilizatorul o asculta
- daca utilizatorul a ascultat 30s sau mai mult din melodie, atunci se va cauta o melodie random din acelasi genre
- alegerea random se va baza pe un **seed** reprezentat de cat a ascultat utilizatorul din melodia curenta

Cand se cere o recomandare, se considera ca utilizatorul asculta o melodie/playlist.

Adaugare recomandare Random Song

```
{
    "command": "updateRecommendations",
    "username": "melodicmind42",
    "timestamp": 35,
    "recommendationType": "random_song"
}
```

```
{
  "command" : "updateRecommendations",
  "user" : "melodicmind42",
  "timestamp" : 135,
  "message" : "The recommendations for user melodicmind42 have been updated successfully."
}
```

Random Playlist

- se va crea un playlist random din melodiile pentru top 3 genre
- top 3 genre se vor determina in functie de cate melodii are utilizatorul din acel gen in
  - melodiile la care utilizatorul a dat like
  - playlist-urile create de utilizator
  - playlist-urile la care utilizatorul a dat follow

Daca se obtin mai putin de 3 genre-uri, se vor lua rezultatele gasite.

- melodiile gasite pentru top 3 genre vor fi **DISTINCTE** si se vor sorta in functie de numarul de likes al melodiei
- pentru fiecare dintre cele 3 genre-uri se vor selecta:
  - top 5 melodii din primul gen
  - top 3 melodii din al 2-lea gen
  - top 2 melodii din ultimul gen
- numele playlistului va fi de forma `"${username}'s recommendations"`

```
{
    "command": "updateRecommendations",
    "username": "alice22",
    "timestamp": 165,
    "recommendationType": "random_playlist"
 }
```

```
{
  "command" : "updateRecommendations",
  "user" : "alice22",
  "timestamp" : 135,
  "message" : "The recommendations for user alice22 have been updated successfully."
}
```

Playlistul **NU** contine melodii duplicate

Fans Playlist

- se va crea un playlist pe baza recomandarilor celor top 5 fans
- pentru artistul melodiei curente, se iau top 5 fans
- pentru fiecare fan gasit, se vor lua top 5 melodii din lista de melodii la care utilizatorul a dat like
- melodiile sunt sortate in functie de numarul de likes
- cele top 5 melodii pentru fiecare fan vor fi adaugate la playlist
- numele playlistului va fi de forma `"${artistName} Fan Club recommendations"`

Adaugare recomandare Fans Playlist

```
{
    "command": "updateRecommendations",
    "username": "alice22",
    "timestamp": 15,
    "recommendationType": "fans_playlist"
}
```

```
{
  "command" : "updateRecommendations",
  "user" : "alice22",
  "timestamp" : 15,
  "message" : "The recommendations for user alice22 have been updated successfully."
}
```

Playlistul **NU** contine melodii duplicate

Daca se gasesc mai putini de 5 fani sau melodii, se vor lua rezultatele gasite.

Mesaje posibile pentru aceasta comanda:

- `"No new recommendations were found"`
- `"The username ${username} doesn't exist."`

- "${username} is not a normal user."
- "The recommendations for user ${username} have been updated successfully."

Flow comanda updateRecommendations

Consideram user-ul *alice22* care asculta melodia *"Amsterdam"* din genre-ul *"alternative rock"*. Deja au trecut 30s de cand a dat play la melodie. Se primeste comanda de *updateRecommendations* de tipul **random song**. Se verifica ca *alice22* a ascultat mai mult de 30s din melodie. Se cauta melodiile din genre-ul *"alternative rock"*. Se alege random una dintre melodiile gasite, unde ***seed*** = *passedTime* din melodia curenta. Aceasta va fi adaugata in recomandarile de melodii ale lui *alice22*, ce vor fi vizibile si in HomePage-ul acesteia.

```
{
  "command": "status",
  "username": "alice22",
  "timestamp": 90
},
{
  "command": "like",
  "username": "alice22",
  "timestamp": 130
},
{
  "command": "updateRecommendations",
  "username": "alice22",
  "timestamp": 135,
  "recommendationType": "random_song"
},
{
  "command": "printCurrentPage",
  "username": "alice22",
  "timestamp": 150
}
```

```
{
  "command" : "status",
  "user" : "alice22",
  "timestamp" : 90,
  "stats" : {
    "name" : "Amsterdam",
    "remainedTime" : 257,
    "repeat" : "No Repeat",
    "shuffle" : false,
    "paused" : false
  }
}, {
  "command" : "like",
  "user" : "alice22",
  "timestamp" : 130,
  "message" : "Like registered successfully."
}, {
  "command" : "updateRecommendations",
  "user" : "alice22",
  "timestamp" : 135,
  "message" : "The recommendations for user alice22 have been updated successfully."
}, {
  "user" : "alice22",
  "command" : "printCurrentPage",
  "timestamp" : 150,
  "message" : "Liked songs:\n\t[Amsterdam]\n\nFollowed playlists:\n\t[]\n\nSong recommendations:\n\t[Reset Me]\n\nPlaylists recommendations:\n\t[]"
}
```

Load Recommendations

Utilizatorul normal va putea asculta ultima recomandare primita prin comanda de *loadRecommendations*, ce este identica cu cea de *load* clasic, doar ca in loc de a lua ultima cautare, se va lua ultima recomandare.

Mesaje posibile pentru aceasta comanda:

- "No recommendations available."
- "Playback loaded successfully."
- "${username} is offline."

Input

```
{
    "command": "loadRecommendations",
    "username": "alice55",
    "timestamp": 275
}
```

Output

```
{
  "command" : "load",
  "user" : "alice55",
  "timestamp" : 155,
  "message" : "Playback loaded successfully."
}
```

Page Navigation

Un utilizator normal va putea da back/foward la pagini. De fiecare data cand utilizatorul primeste comanda de ChangePage, pagina respectiva va fi adaugata la istoricul de navigare al utlizatorului. Paginile prin care utilizatorul poate naviga sunt cele definite in cadrul etapei II - *HomePage, LikedContentPage, ArtistPage* si *HostPage*.

Intr-o succesiune de comenzi back/foward, daca apare o comanda de ChangePage, atunci istoricul pentru foward va fi resetat.

Noile optiuni pentru comanda changePage de `Host/Artist`, te vor directiona catre pagina host-ului, respectiv artistului al carui fisier este la acel moment in playerul userului.

Cand se primeste comanda de

- **nextPage** - utilizatorul va naviga la urmatoarea pagina din istoric, daca exista
- **previousPage** - utilizatorul va naviga la pagina anterioara din istoric, daca exista

Mesaje posibile pentru aceasta comanda:

- `"There are no pages left to go forward."`
- `"There are no pages left to go back."`
- `"The user ${username} has navigated successfully to the next page."`
- `"The user ${username} has navigated successfully to the previous page."`

Navigare la pagina anterioara

```
{
    "command": "previousPage",
    "username": "carol19",
    "timestamp": 25
}
```

```
{
  "command" : "previousPage",
  "user" : "carol19",
  "timestamp" : 25,
  "message" : "The user carol19 has navigated successfully to the previous page."
}
```

Navigare la pagina urmatoare

```
{
    "command": "nextPage",
    "username": "bob35",
    "timestamp": 30
}
```

```
{
  "command" : "nextPage",
  "user" : "bob35",
  "timestamp" : 30,
  "message" : "The user bob35 has navigated successfully to the next page."
}
```

Scheletul de cod

În rezolvarea etapei 3 veți putea folosi ca punct de pornire soluția oficială a etapei 2, dar vă sfătuim să lucrați pe codul scris de voi pentru a putea înțelege mai repede cum să implementați noile funcționalități fără nevoia de a înțelege principiile și interacțiunile din soluția oficială.

Pentru a întelege mai bine cum funcționează citirea/scrie în fișierele JSON vă recomandăm să citiți Json & Jackson [https://ocw.cs.pub.ro/courses/poo-ca-cd/laboratoare/tutorial-json-jackson].

În cadrul soluției oficiale este folosit Lombok [https://projectlombok.org/features/], un tool care simplifică codul prin eliminarea anumitor segmente explicite, cum ar fi getteri și setteri. Puteți folosi și voi aceste adnotări.

Output-ul nu trebuie formatat ca în ref-uri, fiindcă se verifică conținutul obiectelor și array-urilor JSON, nu textul efectiv. Cu toate acestea, dacă folosiți Jackson, vă recomandăm să utilizați **PrettyPrinter** Documentație PrettyPrinter [https://fasterxml.github.io/jackson-databind/javadoc/2.7/com/fasterxml/jackson/databind/ObjectMapper.html#writerWithDefaultPrettyPrinter()]. Totodată, pentru a înțelege cum se poate realiza **scrierea în fișierele JSON de output**, vă sugerăm să consultați JSON Array [http://makeseleniumeasy.com/2020/05/16/rest-assured-tutorial-27-how-to-create-json-array-using-jackson-api-objectmapper-createarraynode/].

Aveți în folder-ul **"lib"** toate dependințele necesare pentru rularea temei, mai exact bibliotecile Jackson.

Execuția temei

1. Se citesc listele cu useri, melodii si podcast-uri, in format JSON - e facuta deja citirea in schelet.
2. Se citesc comenzile si sunt puse in obiecte.
3. Se primesc secvențial comenzi și se execută pe măsură ce sunt primite.
4. După executarea unei comenzi, se afișează rezultatul ei în fișierul JSON de ieșire.
5. La terminarea tuturor comenzilor se termină și execuția programului.

- După ce clonați repo-ul de pe GitHub, vă rugăm să vă faceți un repository propriu privat în care să vă puneți doar conținutul folder-ului **"etapa1"** de pe repo-ul echipei de POO. Dacă nu puneți folder-ul cu tema la alta cale, **nu o să puteți** să faceți schimbări in Git, deoarece vă aflați în rădăcina repository-ului echipei de POO.
- Pentru ca checker-ul să funcționeze trebuie să deschideți tema din Intellij la calea unde se află folderele **"src"**, **"lib"**, **"ref"**, **"input"**. Aveți folder-ul **.idea** pregenerat ca să vă ajute in acest sens. De asemenea, fișier-ul **.iml** contine calea către bibliotecile Jackson. Dacă aveți probleme stergeți folder-ul .idea si fișierul .iml si generațile voi din nou din Intellij.
- **Citirea comenzilor si afisarea rezultatelor trebuie facuta de voi!!!**

Recomandări

- Pentru depanarea diferențelor dintre output-ul vostru si fișierele ref, vă recomandăm acest site [https://www.jsondiff.com/].
- Verificați periodic această pagină, deoarece scheletul/cerința pot suferi modificări în urma unor erori din partea noastră.

Evaluare

Punctajul constă din:

- 80p implementare - trecerea testelor
- 10p coding style (vezi checkstyle)
- 5p README clar, concis, explicații axate pe design (flow, interacțiuni)
- 5p folosire git pentru versionarea temei

Este obligatoriu să folosiți cel puțin 4 design pattern-uri din cele învățate la laborator, depunctarea este de 10 puncte în caz contrar!

Dacă decideți să folosiți Generative AI (ex. ChatGPT) pentru implementare sau alte aspecte ale codului, treceți în README exact unde ați folosit această metodă.

De asemenea, scrieți în README și dacă ați folosit ca schelet rezolvarea oficială a etapei II.

Pentru folosirea tool-ului **Git** vă punem la dispoziție un tutorial actualizat și amplu despre el la acest link și aveți de asemenea și un tutorial despre comenzile pe care puteți să le dați din IntelliJ la acest link.

Pe pagina Indicații pentru teme găsiți indicații despre scrierea readme-ului și depunctările generale pentru teme

Depunctările pentru **designul și organizarea codului** se vor scădea din punctajul testelor. Dacă vor apărea depunctări specifice temei în momentul evaluării, nemenționate pe pagina cu depunctări generale, ele se vor încadra în limitele de maxim 15 pentru design, 5p pentru readme. Dacă tema nu respecta cerințele, sau are zero design OOP atunci se pot face depunctari suplimentare.

Folosirea git pentru versionare va fi verificata din folderul .git pe care trebuie să îl includeți în arhiva temei. Punctajul se va acorda dacă ați făcut **minim 3 commit-uri relevante și cu mesaj sugestiv**. **NU** este permis să aveți repository-urile de git publice până la deadline-ul hard.

Bonusuri: La evaluare, putem oferi bonusuri pentru design foarte bun, cod bine documentat, dar și pentru diverse elemente suplimentare alese de voi.

- Temele vor fi testate împotriva plagiatului. Orice tentativă de copiere va duce la **anularea punctajului** de pe parcursul semestrului și **repetarea materiei** atât pentru sursă(e) cât și pentru destinație(ii), fără excepție.
- **Aveți grijă să nu puneți pe Vmchecker fișiere .idea sau .iml**.

Checkstyle

Unul din obiectivele temei este învățarea respectării code-style-ului limbajului pe care îl folosiți. Aceste convenții (de exemplu cum numiți fișierele, clasele, variabilele, cum indentați) sunt verificate pentru temă de către tool-ul checkstyle [https://checkstyle.sourceforge.io/].

Pe pagina de Recomandări cod găsiți câteva exemple de coding style.

Dacă numărul de erori depistate de checkstyle depășește 30, atunci punctele pentru coding-style nu vor fi acordate. Dacă punctajul este negativ, *acesta se trunchiază la 0*.

Exemple:

- `punctaj_total = 100` și `nr_erori = 200` ⇒ `nota_finala = 90`
- `punctaj_total = 100` și `nr_erori = 29` ⇒ `nota_finala = 100`
- `punctaj_total = 80` și `nr_erori = 30` ⇒ `nota_finala = 80`
- `punctaj_total = 80` și `nr_erori = 31` ⇒ `nota_finala = 70`

Upload temă

Arhiva pe care o veţi urca pe VMChecker [https://vmchecker.cs.pub.ro/ui/#POO] va trebui să conţină în directorul rădăcină:

- fișierul `README.md`
- folder-ul src cu pachetele și cu fișierele .java
- folderul .git

Resurse și linkuri utile

- Schelet de cod [https://github.com/oop-pub/oop-project-2023/tree/main/etapa3]
- Indicații pentru teme
- Recomandări coding style & javadoc

poo-ca-cd/teme/proiect/etapa3.txt · Last modified: 2024/01/14 14:39 by robert.grancsa