Buna ziua, eu ma numesc Trifu Rares Alexandru, sunt student la Facultatea de Automatica si Calculatoare si va voi prezenta o documentatie a aplicatiei create de mine, utilizand React JS si Spring Boot.

Aceasta aplicatie este structurata in mai multe parti, asadar voi face o documentatie a acesteia:

1. Prezentare generala

Aceasta aplicatie are ca si scop crearea unei platforme pentru utilizatori pentru a îndeplini și urmări obiectivele într-un mod distractiv și interactiv.

Obiectivele acestei aplicatii sunt:

- de a permite unui user sa se inregistreze pentru a salva progresele acestuia

- de a raspunde unor intrebari, sub forma de quest-uri, dar totodata si de a le crea, totul desfasurandu-se intr un mediu distractiv

- de a furniza un leaderboard pentru a crea un fel de competitie intre utilizatori

Pentru a crea aceasta aplicatie, am utilizat:

- **Spring Boot**, un framework Java care este folosit pentru a crea aplicatii web, pentru back-end

- **React**, o biblioteca JS, utilizată pentru partea de front-end a aplicației

- **Docker**, pe care l am folosit pentru a pune aplicatia mea in containere si de ag gestiona niste port-uri

- **Tailwind CSS**, un framework folosit pentru crearea de interfețe

- **Node.js**, folosit pentru a rula aplicația

- **PostgreSQL**, baza de data facuta in acesta

2. Mod de utlizare

Pentru a rula aplicatia mea, urmeaza acesti pasi:

- Cloneaza repository-ul din Github care contine partea de front-end si back-end

- Asigura-te ca sunt instalate si configurate PostgreSQL, Node.js, Docker Desktop, Tailwind

- Ruleaza cele 2 containere din Docker, 5050, respectiv 5432

- Porneste aplicatia din Sprin

- Foloseste comanda "npm start" din VSCode

- O fereastra se va deschide si acceseaza "localhost:3000/register"

3. Arhitectura

Aceasta aplicatie este de tip client-server, sugerand faptul ca front-end-ul si back-end-ul comunica intre ele prin intermediul API.

Componente:

- **Front-end**: Reprezinta partea vizuala a aplicatiei si este scrisa in React. In cadrul acesteia, se regasesc toate paginile pe care

le poate accesa user-ul.

- **Back-end**: Reprezinta componenta server-side a aplicatiei, scrisa in Java, foosing framework-ul Spring Boot. Aceasta se ocupa cu

procesarea datelor si comunicarea cu baza de date.

- **Baza de date**: Aceasta se ocupa cu gestionarea datelor, folosing PostgreSQL.

Fluxul de date:

- User-ul are acces la interfata vizuala

- Front-end-ul face apeluri catre API-ul din back-end, folosing Axios pentru a obtine/trimite date

- Back-end proceseaza aceste data, efectueaza metodele necesare si le retrimite spre front-end

- Datele returnate sunt afisate in interfata vizuala prin intermediul front-end-ului

4. Implementare

Autentificare: Aplicatia foloseste un mecanism de autentificare bazat pe cookie-uri. Dupa autentificare, acel cookie se va afla

in response-urile si de fiecare data cand voi face request-uri, acel cookie va fi folosit.

Baza de date: Aplicatia foloseste o baza de data relationala pentru a stoca informatiile utlizatorilor si a quest-urilor.

Quest-urile: Acestea sunt definite intr o baza de date si sunt disponibile user-ilor in functie de raspunsurile acestora sau

de numarul de tokens. Sunt accesate prin endpoint-uri.

In cazul aplicatiei mele, am folosit o arhitectura **MVC** (Model-View-Controller), care presupune faptul ca am impartit aplicatia in

3 componente.

Modelul este responsabil de stocarea și manipularea datelor.

Controller-ul este responsabil pentru gestionarea cererilor HTTP primite si alege ce raspuns sa trimita inapoi.

Serviciile sunt responsabile pentru logica aplicatiei. Ele preiau datele din repository si aplica

operatiile si transformarile necesare pentru a satisface nevoile aplicatiei.

Lista cu fiecare endpoint disponibil in aplicatie:

-privind **User**-ul:

-/account: acest endpoint poate fi folosit pentru a obține informații despre contul utilizatorului curent,

cum ar fi numele de utilizator, adresa de email sau numarul de telefon.

-/signup: acest endpoint este folosit pentru a permite utilizatorilor sa isi creeze un cont nou.

-/login: acest endpoint este folosit pentru autentificarea utilizatorilor. Cand un utilizator introduce numele de utilizator și parola corecte, serverul genereaza un token de autentificare si il trimite înapoi la client pentru a fi utilizat ulterior.

-/updateUser: acest endpoint poate fi folosit pentru a actualiza informatiile de cont ale unui utilizator existent, cum ar fi numele, adresa de email sau numarul de telefon.

-/getAllusers: acest endpoint poate fi folosit pentru a obtine o lista a tuturor utilizatorilor din baza de date.

-/leaderboard: acest endpoint poate fi folosit pentru a obține o lista a utilizatorilor ordonată după numărul de tokens obținute de fiecare dintre ei.

-/userdetails: acest endpoint poate fi folosit pentru a obține informatii detaliate despre un utilizator specific, cum ar fi numele, adresa de email sau numarul de telefon.

-/usernameTaken: acest endpoint poate fi folosit pentru a verifica dacă numele de utilizator introdus de un utilizator este deja utilizat de un alt utilizator.

-/emailTaken: acest endpoint poate fi folosit pentru a verifica dacă adresa de email introdusă de un utilizator este deja utilizată de un alt utilizator.

-/emailValid: acest endpoint poate fi folosit pentru a verifica dacă adresa de email introdusa de un utilizator este valida.

-/usernameValid: acest endpoint poate fi folosit pentru a verifica daca numele de utilizator introdus de un utilizator este valid.

-/phoneValid: acest endpoint poate fi folosit pentru a verifica dacă numărul de telefon introdus de un utilizator este valid.

-/passwordValid: acest endpoint poate fi folosit pentru a verifica daca parola introdusa de un utilizator este valida.

-/isUsernamesPassword: acest endpoint poate fi folosit pentru a verifica daca numele de utilizator și parola introduse de un utilizator sunt corecte.

-/getUserByUsername: acest endpoint poate fi folosit pentru a obține informații despre un utilizator specific utilizand numele de utilizator.

-/solved_quests: acest endpoint poate fi folosit pentru a obtine o lista a quest-urilor pe care utilizatorul le-a rezolvat deja.

-privind **Quest**-ul:

-/displayAll este responsabil pentru afișarea tuturor quest-urilor disponibile din aplicație.

-/addQuest este responsabil pentru adaugarea unui nou quest in aplicație.

-/Quests reprezinta o cale catre resursele referitoare la quest-urile aplicatiei.

Flow-ul aplicatiei:

1) Un utilizator acceseaza aplicatia si incearca sa se autentifice sau sa se inregistreze.

Aceste actiuni sunt realizate prin intermediul endpoint-urilor /login si /signup.

2) Dupa autentificare sau inregistrare, utilizatorul are acces la o pagina de profil,

unde poate vizualiza informatiile sale personale si poate accesa diversele obiective disponibile (quest-uri).

3) Pentru a accesa un obiectiv, utilizatorul trebuie să aibă suficiente tokens si sa indeplineasca alte cerințe specifice obiectivului (cum ar fi sa nu fi completat deja obiectivul respectiv).

4) Utilizatorii pot câștiga tokens prin completarea obiectivelor și prin realizarea altor acțiuni în aplicație.

5) Exista, de asemenea, un leaderboard care afișeaza utilizatorii in funcție de numarul de tokens acumulate.

6) Daca incercam sa accesam endpoint uri cum ar fi /user fara a fi logati, vom primi "Access Forbidden", fiind obligati sa ne intoarcem la pagina de logare.

5. Concluzie

Aplicatia dezvoltata foloseste tehnologii precum Spring Boot, ReactJS, Docker si NodeJS. Arhitectura sa este bazata pe un model de tip client-server.

Autentificarea utilizatorilor este gestionata de server prin intermediul endpointurilor de /login si /signup, iar datele lor sunt stocate intr-o baza de date relationala.

Utilizatorii pot castiga tokens, crea quest-uri si a le rezolva. In general, aplicatia reuseste sa indeplineasca obiectivele sale prin utilizarea tehnologiilor si a arhitecturii adecvate si implementarea corespunzatoare a componentelor sale majore. Cu toate acestea, ar putea fi imbunatatita prin simplificarea si optimizarea endpointurilor pentru o mai buna utilizare si performanta.