Teask-Descriere arhitectură

1.MVC

"Majoritatea aplicațiilor Web sunt dezvoltate conform MVC (Model-View-Controller)." (Curs Tehnologii WEB)

Model: se referă la datele ce vor fi stocate şi regulile, restrângerile asupra acestora. Oferă reprezentarea datelor existente şi se ocupă de validarea acestora. Pentru stocarea datelor ne vom folosi de MySQL**,** un sistem de gestiune a bazelor de date relaţionale. Am ales MySQL deoarece acesta este foarte des folosit împreună cu limbajul PHP.

Modelarea datelor şi proveniența lor: Există date ce provin din mai multe locuri. Vor exista date introduse de client, de exemplu pentru login, register, testele făcute de client spre a fi rezolvate de alţi utilizatori. În plus, aplicaţia noastră se va folosi de date provenite de pe site-uri precum GitHub şi site-uri de conferinţe, evenimente, acestea având ca obiectiv principal discuţia despre diferite limbaje de programare spre a îmbunătăţi înţelegerea participanţilor într-un anumit domeniu. Preluarea datelor din repo-urile de pe GitHub se va face cu ajutorul API-ului de la GitHub. Preluarea datelor de pe site-urile de evenimente şi integrarea în site-ul nostru se va face cu ajutorul unor http requests.Putem ataşa unor site-uri tag-uri care să ne ajute să îi arătăm fiecărui utilizator evenimente cât mai relevante. Totuşi cel mai uşor ar fi să folosim un recommendation engine pentru a oferi recomandări.În plus, pentru a localiza utilizatorul, vom folosi Geolocation API pentru a-i oferi recomandări în zone cât mai apropiate de acesta, iar pentru a genera statistici vom folosi Google API.

Pentru datele introduse de utilizator, vom folosi sistemul de baze de date relaţionale pentru a le stoca. Totuşi parola va fi criptată folosind o funcţie hash sau funţii special create în MySQL pentru criptarea datelor (https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html).

View: furnizează metode de reprezentare a datelor furnizare de model. Aceasta este interfaţa vizibilă utilizatorilor. Pentru view am folosit HTML şi CSS pentru a crea diferite pagini destinate utilizatorilor, pagini prezentate la componenta anterioară a proiectului.

Controller: se ocupă de cererile provenite de la client. Vom avea un server PHP Apache. Aplicaţia PHP va integra elemente şi din alte limbaje, precum JavaScript, pentru a genera, de exemplu, unele dintre statisticile existente în partea de FrontEnd a aplicaţiei.

2.Tehnologiile folosite

Pentru stocarea datelor vom folosi sistemul de gestiune a bazelor de date relaţionale MySQL. Ca limbaj de programare ne vom axa pe PHP şi vom integra elemente din alte limbaje precum JavaScript pentru a genera Google Charts.

API-uri preluate: Pentru a ne folosi de datele de pe GitHub vom prelua API-ul de la GitHub pentru a accesa informaţiile despre fiecare repository. Pentru a afla localizarea utilizatorului vom folosi Geolocation API, API utilizat pentru a oferi recomandari în zone cât mai apropiate de fiecare utilizator. Pentru generarea unor statistici ne vom folosi de API-ul de la Google, API pe care îl integrăm cu ajutorul unui script.

Tehnologii ce pot fi folosite: o altă metodă de afișare a recomandărilor poate fi prin folosirea unui recommendation engine şi un search engine(când utilizatorul dorește să caute anumite evenimente).

Alte API-uri: vom implementa un REST API. Rest Api utilizează limbajul javascript pentru a crea,citi, modifica şi şterge informaţii dintre diferite aplicaţii. De fiecare dată când o aplicaţie se poate conecta la internet şi are nevoie de alte aplicaţii pentru a modifica datele, un Rest Api este de cele mai multe ori folosit

Alte tehnologii ce pot fi folosite: putem folosi diverse librarii pentru a ne facilita munca, librarii precum php Markdown sau Buzz.

3.Funcţionalităţi-câteva dintre ele

Login: utilizatorii se vor putea autentifica imediat din pagina de pornire.După ce introduce username-ul şi parola, se va verifica dacă utilizatorul există în baza de date şi dacă există, i se va afişa pagina principală a utilizatorilor autentificaţi.

Register: utizatorii ce nu se află încă în baza de date, se pot înregistra. După ce introduc datele şi se verifică să nu mai existe aceeași adresă de e-mail sau același username, parola va fi criptată şi datele vor fi stocate în baza de date. Din acest moment, utilizatorul se poate autentifica.

Stabilirea nivelului la un anumit limbaj de programare: nivelul unui utilizator se va stabili pe baza informaţiilor de pe GitHub şi unor teste create pentru diferite limbaje de programare, fiecare limbaj având trei dificultăţi. Se va folosi o funcție ce va stabili punctele la un anumit limbaj. De exemplu testele contează 40% şi GitHub 60%. Răspunsurile la teste reprezintă un anumit punctaj,etc.

Generare statistici: există două moduri de generare a statisticilor. În primul rând sunt statisticile pe baza a ceea ce a făcut pe GitHub, apoi statisticile pe baza evoluţiei sale de când s-a înregistrat, până în momentul actual.Ultimele statistici vor fi generate pe baza a Google Charts.

Token: menţine utilizatorul conectat.

Requesturi http: folosite pentru a genera recomandări pentru utilizator, se vor folosi taguri pentru a face asocierea dintre limbajul cunoscut şi nivelul pentru acel limbaj.De obicei ar trebui să fie afişate conferinţe de acelaşi nivel sau un nivel puţin mai avansat, dacă utilizatorul este aproape să avanseze la următorul nivel.

Scrierea de teste: Un utilizator poate scrie teste pentru limbajele ştiute, la un nivel maxim egal cu al său. Testul va fi şters la un anumit număr de dislike-uri din partea altor utilizatori. Se va urmări ca unui utilizator să nu i se afişeze întrebări introduse de acesta.

Adăugare linkuri evenimente la care a participat+feedback: utilizatorul poate adăuga evenimente la care a participat şi să vizioneze paginile evenimentelor ce s-au sfârşit şi la care a luat parte.