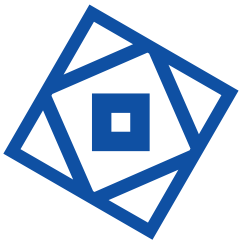

Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

LUCRARE DE LICENȚĂ

FitClub

Absolvent: Necula Bogdan Alexandru
Coordonator: Lect. univ. Bocu Răzvan

Brașov
Iulie 2022

Cuprins

Capitolul 1

Introducere

1.1 Descrierea succintă a temei

Această lucrare constă în dezvoltarea unei platforme web de social networking. Ideea acestei aplicații este de a împărtăși și de a găsi cele mai bune fotografii și videoclipuri. Fiecare profil de utilizator are un număr de urmăritori și urmăriți, reprezentând câte persoane urmăresc și câți alți utilizatori îi urmăresc.

Utilitatea aplicației se rezuma la posibilitatea de a partaja, în timp real, postări ce conțin text și chiar și imagini, însă beneficiază și de un sistem de reacții din partea următorilor, așadar un utilizator care urmărește un alt utilizator poate reacționa prin apreciere sau depreciere la postările acestuia.

Dacă un utilizator dorește să urmărească un alt utilizator, acest lucru este posibil din pagina de profil a respectivului utilizator. Ca și precondiții, este necesară autentificarea cu un cont de utilizator, prin care oricine poate găsi și vizualiza profilul, împreună cu postările recente de pe site. În caz contrar, vizualizarea profilului unui utilizator este în continuare posibila, însă accesul la resursele disponibile va fi limitat, precum faptul că postările respectivului utilizator nu vor putea fi vizibile.

Din punct de vedere al implementării pe partea de server, am ales framework-ul Spring[1], deoarece oferă un model cuprinzător de programare și configurare pentru aplicațiile moderne bazate pe Java, fiind, de asemenea, bine documentat, iar dezvoltarea unei aplicații nu necesita cumpărarea unei licențe.

Din punct de vedere al implementării pe partea de client, am ales să

folosesc React[2], deoarece dispune de diverse instrumente preinstalate care pot fi utilizate direct pentru dezvoltare. De asemenea, React beneficiază și de o comunitate numeroasa, iar acest lucru încurajează rezolvarea eventualelor blocaje tehnice în timpul dezvoltării aplicației.

1.2 Motivarea alegerii temei

Transformarea digitală are implicații profunde pentru companii, și pentru societate în ansamblu. Această revoluție digitală este atât de importantă, încât unii experți o compară cu nașterea tiparului în urmă cu mai bine de cinci secole. În această nouă eră, canalele digitale se înmulțesc, iar utilizările lor cresc.

Rețelele sociale au devenit instrumente de comunicare esențiale. În prezent, companiile trebuie să profite de oportunitățile digitale pentru a-și dezvolta reputația, pentru a-și adapta cultura și pentru a-și fideliza clienții. Creșterea vizibilității mărcii sau împărtășirea noutăților sunt câteva dintre numeroasele avantaje pe care le poate oferi o platformă de social media unei companii.

Dornici de web, de noi forme de consum, datorită internetului, clienții compară, împărtășesc, recomandă un produs/serviciu, iar acest lucru modifică strategiile de marketing, comunicare și vânzare.

Pentru mine, aceste fapte reprezintă un interes deosebit, și totodată, acestea au stat la baza alegerii temei, precum și dorință de a realiza o lucrare cât mai practică, și de actualitate.

Platforma se intitulează FitClub și se adresează clienților unei săli de fitness, oferindu-le posibilitatea de a comunica rapid și eficient.

1.3 Structura lucrării

• **Capitolul 1 -** Capitolul curent, cu rol introductiv în ceea ce privește tema aleasă, motivarea alegerii temei, dar și structura lucrării.

• **Capitolul 2 -** Acest capitol prezintă aspecte teoretice ale lucrării precum și o prezentare detaliată a tehnologiilor folosite la implementarea proiectului folosind atât exemple cât și explicații.

• **Capitolul 3 -** În acest capitol vor fi prezentate detalii cu privire la implementarea propriu-zisă a aplicației, prezentarea modulelor acesteia, cu explicații sugestive ale codului folosit.

• **Capitolul 4 -** Acest capitol va cuprinde prezentarea aplicației în sine, cu accent pe interfața de utilizator. Capitolul va cuprinde, de asemenea, și un ghid de utilizare al aplicației, cu diverse exemple și instrucțiuni de interacțiune.

• **Capitolul 5 -** Concluziile și funcționalități adiționale pentru a spori complexitatea aplicației, în viitor, vor fi cuprinse în cadrul acestui capitol.

Capitolul 2

Tehnologii folosite

2.1 Descrierea unei aplicații web

O aplicație web poate fi manipulată direct online cu ajutorul unui browser web și nu trebuie instalată. Aceasta poate fi plasată și pe un server, iar accesul este simplu și se face prin intermediul unei rețele de calculatoare (internet, rețea locală etc.).

O aplicație web dinamică constă într-un set de pagini statice și dinamice al căror conținut este parțial sau total determinat. Conținutul final al unei pagini este determinat numai atunci când utilizatorul solicită o pagină de la serverul web, iar aceasta variază de la o cerere la alta, în funcție de acțiunile utilizatorului.

Spunem că o pagină web este statică dacă această pagină are un conținut fix și afișează acest conținut către toți utilizatorii.
În opozitie, o pagină web dinamică este mai complexă din punct de vedere tehnic, deoarece utilizează baze de date pentru a încărca informații, iar conținutul este actualizat de fiecare dată când utilizatorul se conectează la aplicație.

Există multe limbaje de programare pentru dezvoltarea de aplicații web. Asadar, pentru a programa această aplicație, am ales unul din cele mai populare framework-uri Java Enterprise Edition (Java EE), și anume, Spring, pe care îl voi prezenta ulterior.

2.2 Generalități despre HTML

HTML[1] este un limbaj de codare utilizat pentru a crea pagini pe care un browser web le poate afișa. Fișierele de tip HTML sunt alcătuite din elemente pereche, unul marcând deschiderea iar perechea sa, închiderea. Site-urile web sunt așadar compuse din diverse pagini HTML legate între ele, stocate undeva pe un server. De asemenea, acesta este motivul pentru care HTML este frecvent numit și "coloana vertebrală a internetului".

Browser-ul folosit de către utilizatori este capabil, astfel, să interpreteze acest limbaj, fară să țină cont de majuscule sau minuscule, afișând rezultatul prin intermediul unei ferestre web.

Organizarea unui document HTML:

- **<html> </html>**

  Acest tag reprezintă elementul de nivel superior sau rădăcina oricărui document HTML. Browser-ul înțelege prin intermediul acestui tag că este vorba de un fișier HTML pentru a-l putea afișa. De asemenea, toate celelalte tag-uri trebuie să fie descendente ale acestui tag.

- **<head> </head>**

  Între aceste tag-uri sunt conținute informații care pot fi citite automat (metadate) despre document precum titlul, script-urile, iar acestea vor fi prelucrate de browser.

- **<title> </title>**

  Aceste tag-uri sunt folosite pentru a da un titlu documentului. Astfel, acest text se va afișa în bara de titlu a browser-ului.

- **<body> </body>**

  Într-un document HTML, poate exista o singură astfel de pereche de tag-uri, ce reprezintă conținutul documentului care va fi afișat pe ecran.

[1] Hyper Text Mark-up Language

2.3 HTML5

A fost prezentat oficial publicului pe 22 ianuarie 2008, având următoarea actualizare majoră în octombrie 2014. Obiectivele acestui update au fost de a îmbunătăți limbajul cu suport pentru cele mai recente caracteristici multimedia și alte caracteristici noi, de a menține limbajul ușor de înteles atât de către oameni, cât și de către computere, fără rigiditatea XHTML-ului, dar și de a rămâne compatibil cu software-ul mai vechi.

HTML4 vs HTML5 :

În comparație cu ediția anterioară de HTML, respectiv HTML4, au fost modificate în mod direct structura, echivalentă marcării, caracteristicile care permit redarea (culori, font etc.), echivalente cu directivele destinate stilului paginilor web și partea de conținut.

HTML5 oferă totodată și o mai bună gestionare a erorilor și are o coerență ridicată în cazul documentelor malformate. De asemenea, versiunea nouă oferă suport în ceea ce privește stocarea de cantități consistente, descărcate de pe web pe spațiul local de memorie, permițând astfel utilizarea offline unor aplicații web.

2.4 Bootstrap

Bootstrap este un framework de dezvoltare web gratuit și open-source, dezvoltat de Twitter în 2011 și lansat pe GitHub în același an, care reprezintă o colecție cu numeroase fragmente de cod reutilizabile și versatile scrise în CSS, HTML și JavaScript pentru a dezvolta rapid și ușor pagini web reactive.
O pagină web reactivă este capabilă să se adapteze automat pentru a arăta mai bine în funcție de rezoluția sau dimensiunea ecranului ori aspectul de imagine.

Așadar, folosind Bootstrap, se pot crea dropdown-uri, alerte, pop-up-uri, navigări etc.

2.5 CSS

CSS[2] permite un control exact asupra aspectului elementelor HTML în browser, și astfel, se pot crea interfețe de utilizator reprezentative, fiecare pagină web putând avea un design unic. În CSS se poate defini culoarea, dimensiunea și poziția textului și a altor etichete HTML, în timp ce fișierele HTML definesc conținutul și organizarea acestuia.

HTML poate defini atât structura, cât și prezentarea, dar WWW[3] recomandă utilizarea CSS, pentru a separa structura unui document HTML de prezentarea sa, deoarece în loc să definească stilul fiecărui tabel și al fiecărui bloc de text în HTML-ul unei pagini, CSS ajută dezvoltatorii front-end să creeze un aspect uniform pe mai multe pagini ale unui site web, definindu-le o singură dată într-un fișier/document CSS, având extensia specifică ".css".

2.6 Funcționarea unei aplicații web

Funcționarea unei aplicații web are la bază anumite mașini care comunică în rețea folosind un limbaj comun. Dintre aceste mașini, se face o distincție între cele care oferă resurse, serverele, și cei care le folosesc, și anume utilizatorii finali sau clienții. Resursele pot fi, de exemplu, documente HyperText, imagini, fișiere XML sau chiar programe (PHP, Java) responsabile pentru generarea lor la cerere.

Un server este un proces care execută o operație la cererea unui client și transmite răspunsul către client, care la rândul său este un procedeu care solicită executarea unei operații de la un web server process prin trimiterea unui mesaj care conține o descriere a operației care urmează să fie efectuată și așteaptă răspunsul la această operație printr-un mesaj de răspuns. Atunci când un client accesează o resursă (consultarea unui document, modificarea datelor stocate pe server etc.), acesta trebuie să utilizeze protocolul de comunicare HTTP pentru a ajunge la aceasta: acesta definește o semantică foarte simplă (GET, POST și alte comenzi) care permite formularea de cereri care sunt interpretate pe server de un program specific, și anume, serverul web.

Un server web este un simplu server de fișiere. Clienții i se adresează prin intermediul protocolului HTTP pentru a obține o resursă. Atunci când serverul web primește o astfel de cerere HTTP, acesta extrage pur și sim-

[2]Cascading Style Sheets
[3]World Wide Web

plu numele resursei solicitate din cerere, apoi extrage resursa de pe disc și o împachetează într-un răspuns HTTP, pentru a o transmite clientului, fără a o prelucra înainte de a o transmite acestuia. Prin urmare, acesta poate transfera către un client o pagină HTML, o imagine, un sunet sau chiar un fișier executabil. Tipul de conținut al resursei solicitate este total irelevant pentru aceasta. Atunci când un server web primește o cerere pentru o pagină web statică, acesta transmite această pagină direct către browserul care o solicită.

Cu toate acestea, atunci când serverul web primește o cerere pentru o pagină web dinamică, reacționează diferit: transmite pagina către un software special care este responsabil pentru finalizarea paginii. Acest software special se numește server de aplicații.

Un server de aplicații este radical diferit de serverul web, deoarece resursele care îi sunt oferite nu sunt doar fișiere statice, ci coduri pe care le va executa în numele clienților care le solicită. Atunci când un server de aplicații primește o cerere HTTP, acesta analizează și această cerere pentru a determina ce resursă este solicitată. De obicei, cererea este pentru un cod executabil găzduit de server. Spre deosebire de un server web aflat în aceeași situație, acesta nu transferă codul către client, ci îl execută, iar rezultatul acestui cod va fi returnat clientului.

Server-ul de aplicații citește codul din pagină, completează pagina în conformitate cu instrucțiunile cuprinse în cod și ulterior, elimină codul. Server-ul de aplicații trimite înapoi la server-ul web rezultatul acestui proces care constă într-o pagină statică. La rândul său, trimite această pagină la browser-ul solicitant. Browser-ul primește, în consecință, doar cod HTML pur atunci când îi este trimisă pagina, pe care îl prelucrează corespunzător.

Figura 2.1 prezintă o imagine de ansamblu a acestui proces:

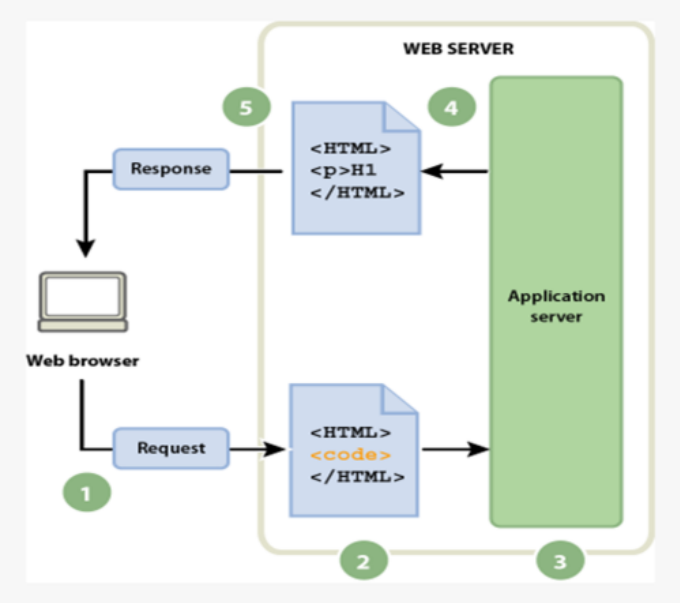

Figura 2.1: Funcționarea unei aplicații web

Observație: O resursă este identificată prin url (șir de caractere). Atunci când clientul dorește să ajungă la o resursă la distanță, acesta trimite o cerere HTTP în care menționează adresa URL a resursei.

2.7 Arhitectura unei aplicații web

2.7.1 Arhitectura pe două niveluri (client/server)

Arhitectura pe două niveluri numită și arhitectura "client-server" nu este deloc complicată. Pe de o parte se află clientul, iar pe de altă parte server-ul, așa cum este reprezentat și în Figura 2.2. Acest tip de arhitectură poate fi realizat pe orice tip de arhitectură hardware interconectată.

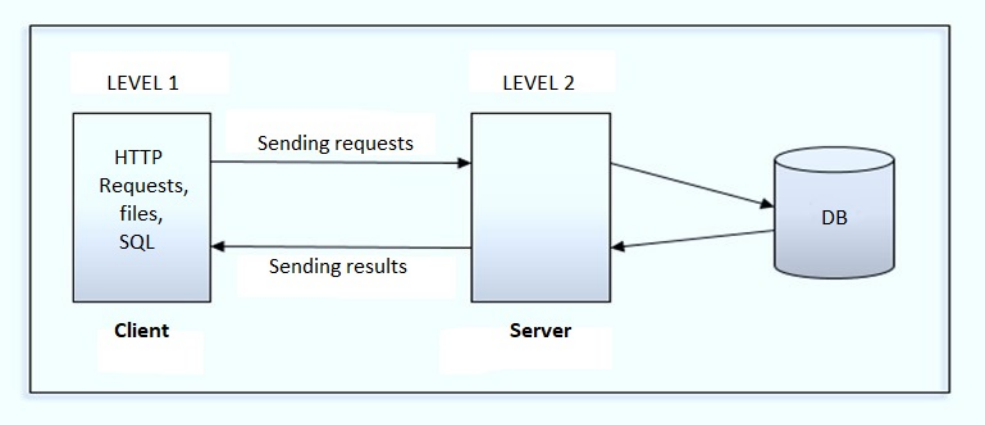

Figura 2.2: Arhitectura unei aplicații pe 2 niveluri

Punctele importante ale unei aplicații pe două niveluri:

– Clientul solicită un serviciu de la server, cum ar fi să i se afișeze o anumită pagină.

– Server-ul primește această cerere HTTP, efectuează procesarea și returnează resursa solicitată de client.

– Clientul primește resursa solicitată.

2.7.2 Arhitectura pe trei niveluri

În arhitectura pe trei niveluri (Figura 2.3), apare un nou nivel. Într-adevăr, mai avem încă primul nivel, care este clientul, care este foarte ușor, deoarece nu are niciun rol de procesare, utilizează aplicația, dar are doar o mică parte a aplicației pe dispozitivul său de lucru.

Toată procesarea se face apoi pe partea server-ului. La al doilea nivel se află server-ul de aplicații și, în sfârșit, la ultimul nivel, server-ul de baze de date, de unde se vor procesa anumite interogări SQL. Funcţia fundamentală a unui server de baze de date este reprezentată de: stocarea, regăsirea şi reactualizarea datelor. Pentru asigurarea acesteia, server-ul trebuie să ascundă față de utilizatorul care accesează site-ul, informațiile referitoare la implementarea fizică internă, cum ar fi detalii despre organizarea fișierelor, dar şi structurile de stocare.

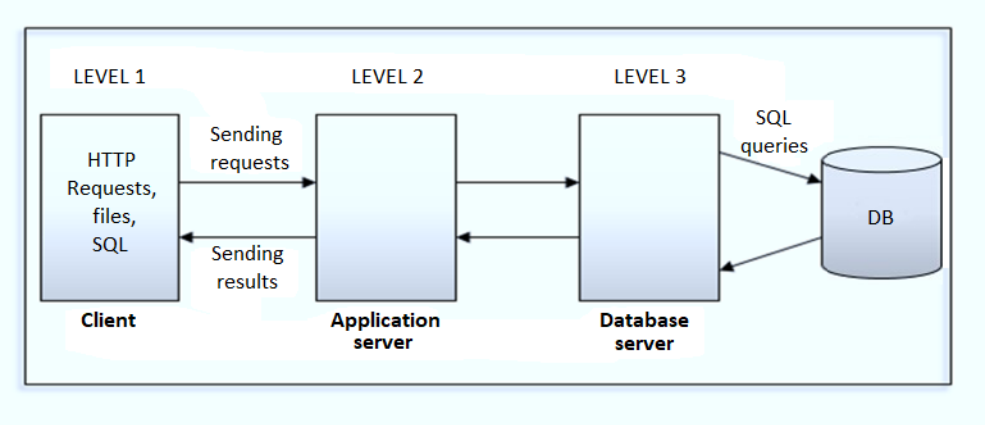

Figura 2.3: Arhitectura unei aplicații pe 3 niveluri

Punctele importante ale unei aplicații pe trei niveluri:

– Clientul este utilizat doar pentru a solicita și afișa răspunsurile serverului.

– Server-ul de aplicații se ocupă de calcule și chiar solicită servere suplimentare.

– Server-ul DB se ocupă de stocarea datelor.

2.8 Arhitectura Java EE

Arhitectura Java EE se bazează pe o arhitectură stratificată, care se bazează pe o împărțire în straturi, astfel încât vorbim de aplicații pe mai multe niveluri. Astfel, sunt reprezentate trei straturi principale, în care sunt distribuite diferite componente pe care le vom vedea în restul acestui capitol. Scopul acestei diviziuni este de a propune o mai bună distribuție a rolurilor (fiecare strat are un rol clar definit) și o mai bună mentenabilitate, având în vedere că fiecare strat este independent de celelalte.
Are loc, așadar, următoarea împărțirea în straturi:

- Stratul destinat afișării paginilor web
- Stratul destinat gestionării logicii de afaceri
- Stratul destinat salvării informațiilor din baza de date

Stratul destinat afișării paginilor web gestionează afișarea datelor și interacțiunile aplicației cu utilizatorul. Separarea acestui strat face posibilă oferirea mai multor prezentări pentru aceeași aplicație: același strat de procesare poate fi utilizat pentru o aplicație grea și pentru o aplicație ușoară.

Stratul destinat gestionării logicii de afaceri se referă atât la sarcinile care trebuie efectuate de aplicație asupra datelor, cât și la prelucrările necesare în urma unei acțiuni din partea utilizatorului: (verificarea stării utilizatorului: autentificat/neautentificat, diverse calcule etc.).

Stratul destinat salvării informațiilor din baza de date (sistemul informatic al aplicației) grupează mecanismele de stocare și de acces la date, astfel încât acestea să poată fi utilizate de aplicație la nivelul de procesare.

2.9 Arhitectura MVC

Majoritatea platformelor de dezvoltare a aplicațiilor web, inclusiv Java EE, nu impun o anumită ordine a codului, adică putem dezvolta în orice mod doriți. Problema este că, dacă dezvoltăm în orice mod, codul rezultat va fi nesatisfăcător din punct de vedere calitativ și va deveni dificil să găsim o bucată de cod sau o funcție pe care dorim să o modificăm.
Pentru a evita această problemă, este recomandat sa utilizăm bune practici de dezvoltare numite Design Patterns. Un model de proiectare permite descrierea liniilor principale ale unei soluții.

Modelul MVC descompune aplicația în straturi distincte și, prin urmare, are un impact puternic asupra organizării codului. MVC este acronimul de la Model - View - Controller. Practic, atunci când dezvoltăm o aplicație folosind arhitectura MVC, vom segmenta codul în trei părți sau straturi, fiecare strat având o funcție specifică.

2.9.1 Stratul Model

Aceasta este partea de cod care execută logica de afaceri a aplicației. Acest lucru înseamnă că este responsabil pentru recuperarea datelor, conversia lor în conformitate cu conceptele logicii aplicației, cum ar fi procesarea, validarea, asocierea și orice alte sarcini legate de manipularea datelor.

Este, de asemenea, responsabil pentru interacțiunea cu baza de date, știe cum să se conecteze la o bază de date și să execute interogări, folosind cele patru operațiuni de bază ale stocării persistente (CREATE, READ, UPDATE, DELETE) pe o bază de date.

2.9.2 Stratul View

Aceasta este partea de cod care se va ocupa de prezentarea datelor către utilizator, care returnează o vizualizare a datelor din model, cu alte cuvinte, este responsabilă pentru producerea interfețelor de prezentare ale aplicației din informațiile pe care le deține (de exemplu, pagina HTML).

Acest nivel nu se limitează însă doar la reprezentarea datelor în format HTML sau text, ci poate fi utilizat și pentru a oferi o mare varietate de formate, în funcție de ceea ce avem nevoie sa utilizăm pentru a implementa o anumită funcționalitate.

2.9.3 Stratul Controller

Acesta este stratul responsabil cu rutarea informațiilor, va decide cine va prelua informațiile și le va procesa. De asemenea, gestionează cererile utilizatorilor și returnează un răspuns cu ajutorul straturilor Model și View, descrise anterior.

Figura 2.4 prezintă o imagine de ansamblu a arhitecturii MVC:

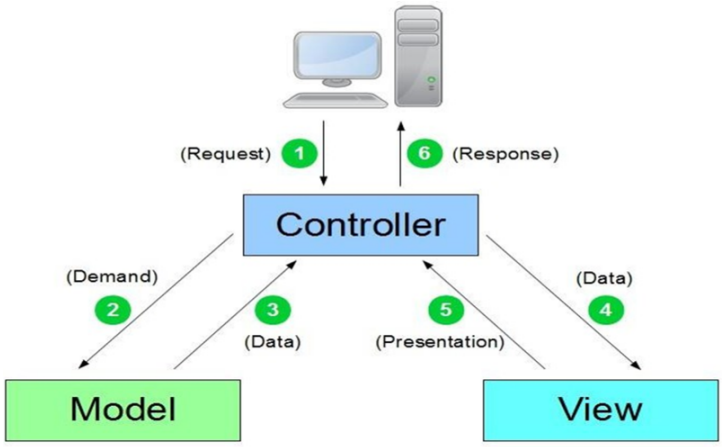

Figura 2.4: Arhitectura Model - View - Controller

Principiile arhitecturii MVC:

– Utilizatorul trimite cererea HTTP, o transmite server-ului de aplicații care o transmite direct către partea de cod numită Controller.

– Controller-ul se ocupă de direcționarea informațiilor, hotărând cine va prelua informațiile și apoi le va procesa, apelând, de fapt, modelul care conține informațiile structurate.

– Modelul va efectua calcule sau prelucrări pe baza acestor informații și le va trimite înapoi la Controller.

– Controller-ul va cere View-ului să genereze un View (pagină web).

– View-ul generează o pagină web așa cum a fost solicitată de controller și o trimite către Controller.

– Controller-ul primește pagina web pe care o va trimite utilizatorului, ca urmare a solicitării.

2.10 React

React este o bibliotecă JavaScript open-source dezvoltată de Facebook începând din 2013, ce s-a axat pe crearea de interfețe pentru utilizator declarative folosind un concept bazat pe componente, în care fiecare componentă are propria stare.
Dacă o componentă a paginii web tocmai a fost actualizată (de exemplu, dacă apăsăm pe un buton care declanșează apariția unui text pe pagină), acea parte este singura modificată de React fără a actualiza întreaga pagină. React utilizează "DOM virtual", care este o reprezentare a interfeței cu utilizatorul, stocată în memorie și sincronizată în mod constant cu "DOM-ul" real.

Așadar, React nu este un framework, ci mai exact o bibliotecă, deoarece se ocupă doar de redarea interfețelor de utilizare și rezervă multe lucruri la discreția proiectelor individuale, putând fi considerată "View-ul" în modelul MVC. De asemenea, bibliotecile externe pot ajuta dezvoltatorul să utilizeze atât funcționalitățile HTML, cât și cele CSS și să le construiască în JSX.

Observație: JSX este o sintaxă de extensie JavaScript utilizată în React pentru a scrie cu ușurință HTML și JavaScript împreună.

Totuși, React are anumite limitări:

- Adesea, devine necesar ca anumite pachete să fie descărcate pe stația de lucru a dezvoltatorului pentru a putea accesa diverse funcționalități implementate deja în React
- Deși React suportă o mulțime de biblioteci externe, există foarte puține biblioteci native.
- Navigarea datelor în React este complicată și complexă, deoarece manipularea paralelă a datelor nu este suportată.

2.11 MySQL

MySQL este un sistem de gestionare a bazelor de date care permite stocarea datelor într-un mod structurat (bază de date, tabele, câmpuri, înregistrări etc.). Nucleul acestui sistem permite accesul la informațiile stocate prin intermediul unui limbaj specific numit SQL.

Popularitatea MySQL este datorată, în special, de natura sa open-source, stabilitate, dar și de ușurința de utilizare și a performanței ridicate.

2.12 Git

Git este un software descentralizat de control al versiunilor creat de Linus Torvalds, autorul nucleului Linux. Git poate fi descris ca un instrument de urmărire a conținutului utilizat în principal pentru a stoca cod datorită celorlalte caracteristici pe care le oferă. Acest cod stocat continuă să se schimbe pe măsură ce se adaugă funcționalități sau se modifică cele deja existente.

Git are un depozit la distanță unde sunt stocate toate aceste informații și un depozit local care este stocat pe computer-ul fiecărui dezvoltator. Acest lucru înseamnă că varianta completă a codului unei aplicații nu există doar pe depozitul central, ci este prezentă pe toate computer-ele dezvoltatorilor care lucrează la aplicația respectivă.

Așadar, un astfel de sistem de control al versiunilor ajută prin păstrarea unui istoric al modificărilor ce au avut loc de-a lungul dezvoltării aplicației.

2.13 Editoare de cod sursă

2.13.1 Visual Studio Code

În centrul său, Visual Studio Code oferă un editor de cod sursă foarte rapid, perfect pentru utilizarea zilnică. Cu suport pentru sute de limbaje, VS Code ajută prin caracteristicile sale, dar și prin mulțimea de extensii disponibile în Marketplace-ul integrat să menținem productivitatea la un nivel ridicat prin ilustrarea sintaxei, potrivirea corectă a parantezelor, indentare automată, selectarea fragmentelor de cod, a casetelor și multe altele. Scurtăturile de la tastatură sunt intuitive, datorită comunității numeroase de utilizatori care au contribuit cu astfel de sugestii, iar personalizarea ușoară a tuturor opțiunilor disponibile incluzând și schimbarea acestor scurtături de la tastatură, în funcție de propriile preferințe, permit navigarea extrem de ușoară și facilă prin codul aplicației.

Menționăm câteva dintre caracteristicile unice ale Visual Studio Code:

– Include un suport încorporat îmbogățit pentru dezvoltarea Node.js cu JavaScript, având instrumente excelente pentru tehnologii web, cum ar fi JSX/React, HTML, CSS, Bootstrap.

– Include suport încorporat pentru finalizarea codului, numit IntelliSense, folosit așadar pentru înțelegere și navigare semantică bogată a codului și refactorizare a codului.

– De obicei, suportă toate limbajele de programare, dar, dacă se dorește utilizarea unui limbaj de programare care nu este acceptat, atunci se poate descărca și utiliza o extensie.

– Include suport pentru Git, astfel încât se poate sincroniza în timp real cu sistemul de control al versiunilor fără a părăsi editorul, inclusiv vizualizarea modificărilor în așteptare.

2.13.2 IntelliJ IDEA

IntelliJ este unul dintre cele mai puternice și populare medii de dezvoltare integrată (IDE) pentru Java. Este dezvoltat și întreținut de JetBrains și este disponibil în edițiile Community și Ultimate. Acest IDE bogat în funcții permite dezvoltarea rapidă și ajută la îmbunătățirea calității codului. Algoritmul său predictiv poate presupune cu exactitate ceea ce un dezvoltator încearcă să tasteze și astfel, devine reprezentativă utilitatea acestei

funcționalități prin diversele sugestii ce sunt afișate, chiar dacă nu cunoaște numele exact al unei anumite clase, membru sau orice altă resursă.

Menționăm, așadar, câteva dintre caracteristicile productive de top ale IntelliJ IDEA:

– Suportă completarea inteligentă a codului bazată pe context. Oferă o listă cu cele mai relevante simboluri aplicabile în contextul curent.

– Suportă completarea codului în lanț care este o funcție avansată ce listează simbolurile aplicabile accesibile prin metode în contextul curent.

– Permite utilizarea de metode sau constante statice, și adaugă automat declarațiile de import necesare pentru a evita erorile de compilare.

– Găsește din mers fragmentele de cod duplicat și oferă utilizatorului o notificare/sugestie în acest sens.

– Capabil să detecteze în timp real o posibilă greșeală, fie că vorbim despre o greșeală de scriere, sau accesarea unei variable/metode inexistente sau orice altă situație ce poate genera eroare de compilare, caz în care o mică notificare cu beculeț apare pe aceeași linie.
Inspectând această notificare, se afișează o listă de sugestii prin care dezvoltatorul poate remedia rapid greșeala.

Bibliografie

[1] *Spring Framework*, URL: https://spring.io/projects/spring-framework.

[2] *ReactJS*, URL: https://reactjs.org/docs/getting-started.html.