**PERDELIYEVA AKJEMAL**

**CSC AN1, GRUPA B**

Obiectiv:

Obiectivul acestui proiect este familiarizarea cu șablonul architectural Client/Server. Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

Cerințe:

❖În faza de analiză se va realiza diagrama cazurilor de utilizare.

❖În faza de proiectare se vor realiza:

➢diagrama de clase corespunzătoare aplicației server,

➢diagrama de clase corespunzătoare aplicației client,

➢diagrama entitate-relație corespunzătoare bazei de date.

❖În faza de implementare se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:

➢proiectarea dată de diagrama de clase;

➢unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

❖Finalizarea temei va consta în predarea unui director ce va cuprinde:

➢Un fișier cu diagramele UML realizate;

➢Baza de date;

➢Aplicația soft;

➢Documentația (minim 15pagini)-un fișier care cuprinde:

▪numele studentului, grupa;

▪enunțul problemei;

▪instrumente utilizate;

▪justificarea limbajului de programare ales;

▪prezentarea etapelor de dezvoltare a aplicației informatice:

•etapa de analiză: descriere diagrame UML;

•etapa de proiectare: descriere diagrame UML;

•etapa de implementare: prezentarea interfețelor grafice și a funcționalităților implementate.

**Problema 15**

Dezvoltați o aplicație care poate fi utilizată de către o agenție de rezervare bilete de tren. Aplicația va avea 3 tipuri de utilizatori: călător, angajat și administrator.

Utilizatorii de tip călător pot efectua următoarele operații fără autentificare:

❖Vizualizarea listei trenurilor după stație de plecare, destinație, durată ordonată după numărul trenurilor;

❖Vizualizarea listei trenurilor dintre 2 locații, inclusiv preț și disponibilitate locuri libere;

❖Căutarea unui tren după număr.

Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

❖Toate operațiile permise utilizatorilor de tip călător și vânzarea unui bilet către un călător;

❖Operații CRUD în ceea ce privește persistența trenurilor și biletelor vândute;

❖Salvarea listelor cu informații despre trenuri în formatele csv, json, xml, txt;

❖Vizualizarea unor statistici legate de trenuri utilizând grafice (structură radială, structură inelară, de tip coloană, etc.).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖Toate operațiile permise utilizatorilor de tip călător;

❖Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare;

❖Vizualizarea listei utilizatorilor care necesită autentificare;

❖Notificarea fiecărui utilizator care necesită autentificare prin cel puțin 2 variante (email, SMS, WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente acelui utilizator.

1. **Instrumente utilizate**

Pentru dezvoltarea aplicației, am utilizat următoarele instrumente:

* Backend: Java Spring cu arhitectura MVC. Spring oferă un cadru robust pentru dezvoltarea aplicațiilor Java de tip enterprise, facilitând implementarea logicii de afaceri, gestionarea datelor și comunicarea cu baza de date. Arhitectura MVC (Model-View-Controller) asigură separarea clară între logica de prezentare, logica de afaceri și stocarea datelor.
* Frontend: React și JavaScript. React este o bibliotecă JavaScript populară pentru construirea interfețelor de utilizator interactive. Este eficientă în gestionarea stării aplicației și oferă o reprezentare virtuală a interfeței, facilitând actualizările selective și reutilizarea componentelor. JavaScript este limbajul de programare principal utilizat în dezvoltarea aplicațiilor web și este esențial pentru interacțiunea cu backend-ul.

1. Java Spring Java Spring este un cadru de dezvoltare open-source pentru aplicații Java de tip enterprise. Acesta oferă un set puternic de funcționalități și abstracții care facilitează dezvoltarea rapidă și eficientă a aplicațiilor robuste și scalabile. Iată câteva caracteristici importante ale Java Spring:

* Inversiunea controlului (Inversion of Control - IoC): Spring implementează principiul IoC, prin care controlul fluxului aplicației este transferat către cadru. Aceasta elimină dependențele strânse dintre componente și permite o structură modulară și flexibilă a aplicației.
* Injecția de dependențe (Dependency Injection - DI): Spring facilitează gestionarea dependențelor prin intermediul injecției de dependențe. Aceasta permite crearea modulară a componentelor și le oferă acces la dependențele necesare prin intermediul unor mecanisme flexibile de configurare.
* Spring MVC: Spring MVC (Model-View-Controller) este un modul al cadrului Spring care facilitează dezvoltarea aplicațiilor bazate pe arhitectura MVC. Acesta separă clar logica de prezentare (View), logica de afaceri (Controller) și stocarea datelor (Model), asigurând o structură coerentă și ușor de gestionat a aplicației.
* Gestionarea tranzacțiilor: Spring oferă suport puternic pentru gestionarea tranzacțiilor în aplicații. Acesta facilitează definirea și gestionarea tranzacțiilor prin intermediul unor mecanisme simple și declarative, asigurând integritatea și consistența datelor.
* Securitate: Spring Security este un modul al cadrului Spring care oferă facilități extinse pentru implementarea securității în aplicații. Acesta permite configurarea autentificării, autorizării și gestionarea sesiunilor utilizatorilor în mod flexibil și securizat.

1. React React este o bibliotecă JavaScript pentru construirea interfețelor de utilizator interactive. A fost dezvoltată de către Facebook și este extrem de populară datorită abordării sale declarative și performanței ridicate. Iată câteva caracteristici cheie ale React:

* Componente reutilizabile: React se bazează pe conceptul de componente, care sunt unități izolate și reutilizabile ale interfeței de utilizator. Aceasta facilitează structurarea și organizarea codului și permite crearea unei ierarhii complexe de componente.
* Modelul de programare declarativ: React utilizează un model de programare declarativ, în care programatorul definește cum ar trebui să arate interfața în funcție de starea aplicației. Orice actualizare a stării va determina o reprezentare reactivă a interfeței, fără a necesita actualizări manuale.
* React Virtual DOM: React utilizează un Virtual DOM pentru a eficientiza actualizările interfeței. Acesta reprezintă o copie a structurii DOM a aplicației și se actualizează doar atunci când starea se schimbă. Astfel, React minimizează numărul de modificări necesare în DOM și optimizează performanța aplicației.
* Gestionarea stării aplicației: React oferă un mecanism simplu și eficient pentru gestionarea stării aplicației. Prin intermediul conceptului de stare locală a componentelor și utilizarea funcțiilor de stare și efecte, programatorul poate gestiona și actualiza starea aplicației într-un mod coerent și reactiv.
* Comunitatea și ecosistemul vast: React beneficiază de o comunitate puternică și activă, oferind o gamă largă de biblioteci și instrumente adiacente. Acestea facilitează dezvoltarea aplicațiilor complexe, precum gestionarea rutei (React Router), manipularea stării globale (Redux) și testarea unitară (Jest).

1. JavaScript JavaScript este un limbaj de programare dinamic, orientat pe obiecte și interpretat, care este esențial în dezvoltarea aplicațiilor web moderne. Este utilizat în principal pentru a oferi interactivitate și funcționalitate pe partea de client a aplicațiilor. Iată câteva caracteristici importante ale JavaScript:

* Programare orientată pe evenimente: JavaScript este bazat pe un model de programare orientat pe evenimente, ceea ce înseamnă că poate răspunde la evenimente declanșate de utilizatori (clicuri, introducerea datelor etc.) și poate executa acțiuni corespunzătoare.
* Manipularea DOM-ului: JavaScript oferă funcționalități puternice pentru manipularea și modificarea structurii și conținutului Document Object Model (DOM) al unei pagini web. Acest lucru permite actualizări dinamice ale interfeței și interacțiunea cu elementele paginii.
* Gestionarea asincronă: JavaScript suportă gestionarea asincronă a operațiilor prin intermediul funcțiilor de callback, promisiunilor și așteptărilor (async/await). Aceasta permite efectuarea de cereri către server, manipularea fișierelor și alte operații care necesită timp de execuție mai mare, fără a bloca interfața de utilizator.
* Ecosistemul bogat de biblioteci și framework-uri: JavaScript beneficiază de un ecosistem vast de biblioteci și framework-uri care facilitează dezvoltarea rapidă și eficientă a aplicațiilor web.

1. **Justificarea limbajului de programare ales**

4.1. Java pentru dezvoltarea serverului

Java a fost selectat ca limbaj de programare pentru dezvoltarea serverului în aplicația de rezervare bilete de tren datorită numeroaselor avantaje și caracteristici pe care le oferă.

Robustețe și stabilitate: Java este cunoscut pentru a fi un limbaj de programare robust și stabil, utilizat într-o varietate de aplicații critice. Este proiectat să fie rezistent la erori și să ofere mecanisme puternice pentru gestionarea excepțiilor. Această robustețe face din Java o alegere potrivită pentru un server care trebuie să rămână stabil și să funcționeze în mod eficient în fața cererilor multiple și complexe.

Portabilitate: Java este un limbaj portabil care rulează pe mașini virtuale Java (Java Virtual Machines - JVM), ceea ce înseamnă că codul scris în Java poate fi rulat pe diferite sisteme de operare și arhitecturi hardware. Acest aspect este crucial în dezvoltarea aplicațiilor client-server, deoarece serverul poate fi implementat pe diferite platforme, oferind flexibilitate și interoperabilitate.

Ecologie și biblioteci bogate: Java beneficiază de un ecosistem bogat de biblioteci și framework-uri, care facilitează dezvoltarea rapidă a aplicațiilor. Există framework-uri precum Spring, care oferă suport pentru dezvoltarea serviciilor web RESTful, gestionarea dependențelor și crearea unei arhitecturi modulare și scalabile. Prin utilizarea acestor biblioteci și framework-uri, dezvoltatorii pot accelera procesul de dezvoltare și pot beneficia de soluții testate și robuste.

Performanță și scalabilitate: Java este recunoscut pentru performanța sa, iar JVM-ul optimizează codul în timpul execuției pentru a obține o performanță maximă. Acest aspect este esențial într-o aplicație client-server care trebuie să răspundă rapid și eficient cererilor utilizatorilor și să suporte un număr mare de conexiuni simultane. JVM-ul oferă, de asemenea, suport pentru gestionarea memoriei și colectarea automată a gunoiului, ceea ce contribuie la o utilizare eficientă a resurselor și la scalabilitate.

4.2. JavaScript pentru dezvoltarea frontend-ului

Pentru partea de frontend a aplicației, s-a ales utilizarea limbajului JavaScript și a framework-ului React datorită beneficiilor și funcționalităților pe care le oferă.

Interfețe de utilizator interactive și reactive: JavaScript este un limbaj de programare foarte versatil și puternic, care permite dezvoltatorilor să creeze interfețe de utilizator interactive și reactive. Prin intermediul JavaScript, se pot implementa funcționalități complexe și dinamice în interfața aplicației, cum ar fi validarea în timp real a datelor, animații fluide și interacțiuni interactive. Aceasta oferă o experiență mai bogată și mai captivantă pentru utilizatori.

Model de programare declarativ: React utilizează un model de programare declarativ, în care dezvoltatorii descriu cum ar trebui să arate interfața în funcție de starea aplicației. Aceasta elimină complexitatea de a manipula direct DOM-ul și facilitează actualizările eficiente ale interfeței în funcție de modificările de stare. Acest model simplifică dezvoltarea și întreținerea codului și permite dezvoltatorilor să se concentreze mai mult pe logica aplicației decât pe aspectele tehnice.

1. **Etapele de dezvoltare a aplicației**

Pentru dezvoltarea aplicației, am urmat următoarele etape:

a) Proiectarea și definirea cerințelor: Am stabilit cerințele funcționale și non-funcționale ale aplicației, inclusiv tipurile de utilizatori, operațiile permise și datele necesare.

b) Proiectarea arhitecturii: Am proiectat arhitectura aplicației utilizând arhitectura client-server, cu serverul dezvoltat în Java Spring și frontend-ul realizat în React și JavaScript.

c) Implementarea serverului: Am dezvoltat serverul folosind Java Spring, definind modelele de date, controlerele și serviciile necesare pentru gestionarea cererilor primite de la client.

d) Implementarea frontend-ului: Am creat interfața de utilizator utilizând React și JavaScript. Am definit componente reutilizabile și am gestionat starea aplicației pentru a facilita interacțiunea cu serverul prin intermediul endpoint-urilor definite.

e) Testare și depanare: Am efectuat teste pentru a verifica corectitudinea și performanța aplicației. Am rezolvat eventualele erori și probleme identificate în timpul testării.

f) Implementarea securității și autentificării: Am adăugat funcționalitatea de autentificare și autorizare pentru diferitele tipuri de utilizatori. Am asigurat că doar utilizatorii autorizați au acces la operațiunile specifice tipului lor.

**6.Descriere diagrame UML**

**O imagine care conține text, diagramă, captură de ecran, linie

Descriere generată automat**

**O imagine care conține text, captură de ecran, diagramă, alb și negru

Descriere generată automat**

**O imagine care conține text, captură de ecran, Notiță adezivă, Dreptunghi

Descriere generată automat**

O imagine care conține text, captură de ecran, Dreptunghi, pătrat

Descriere generată automat