Tema 3

Nume:Bogdan Andrei

Grupa:30233

Contents

[Enuntul problemei: 3](#_Toc130429240)

[Instrumente utilizate: 4](#_Toc130429241)

[Justificarea limbajului de programare ales: 5](#_Toc130429242)

[descrierea diagramelor UML: 6](#_Toc130429243)

[Descrierea aplicației: 10](#_Toc130429244)

[Bibliografie: 12](#_Toc130429245)

# Enuntul problemei:

Dezvoltați o aplicație care poate fi utilizată într-un oraș pentru determinarea traseului optimutilizând transportul în comun. Aplicația va avea 3 tipuri de utilizatori: călător, angajat al firmei de transport în comun și administrator.Utilizatorii de tip călătorpot efectua următoarele operații fără autentificare:

❖Vizualizarea listei tuturor liniilor de transport în comun sortată după număr;

❖Vizualizarea traseului optim după specificareastației de plecare și a stației de sosire(reprezentare grafică);

❖Căutarea unei linii de transport în comun după număr.Utilizatorii de tip angajatpot efectua următoarele operații după autentificare:

❖Toate operațiile permise utilizatorilor de tip călător;❖Operații CRUD în ceea ce privește persistența liniilor de transport în comun;

❖Salvareliste cu liniile de transport în comunîn mai multe formate:csv, json, xml, txt;

❖Vizualizarea unor statistici legate de liniile de transport în comunutilizând grafice (structură radială, structură inelară, de tip coloană, etc.).Utilizatorii de tip administratorpot efectua următoarele operații după autentificare:

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

❖Vizualizarea listei utilizatorilor care necesită autentificare.Interfața grafică a aplicației va fi disponibilă în cel puțin 3 limbi de circulație internațională.

# Instrumente utilizate:

Am folosit ca IDE InteliJ deoarece imi este cel mai familiar deoarece toate proiectele anterioare au fost facute cu ajutorul lui.

Am folosit postgresql pentru baza de date fig 2.1:

A picture containing text

Description automatically generated

Fig2.1

Pentru legatura dintre java application si baza de date am folosit Hibernate,fig 2.2.Hibernate ORM (sau pur și simplu Hibernate ) este un instrument [de mapare obiect-relațională](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping) pentru limbajul de programare [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) . Acesta oferă un [cadru](https://en.wikipedia.org/wiki/Software_framework) pentru maparea unui model de domeniu [orientat pe obiecte la o](https://en.wikipedia.org/wiki/Object-oriented_programming)[bază de date relațională](https://en.wikipedia.org/wiki/Relational_database) . Hibernate tratează problemele [de nepotrivire a impedanței relaționale obiect](https://en.wikipedia.org/wiki/Object%E2%80%93relational_impedance_mismatch) prin înlocuirea acceselor directe și [persistente](https://en.wikipedia.org/wiki/Persistence_(computer_science)) la bazele de date cu funcții de manipulare a obiectelor de nivel înalt.

Hibernate este [un software gratuit](https://en.wikipedia.org/wiki/Free_software) care este distribuit sub [Licența Publică Generală Inferioară GNU](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License) 2.1.

Caracteristica principală a lui Hibernate este maparea de la clase Java la [tabelele bazei de date](https://en.wikipedia.org/wiki/Table_(database)) și maparea de la tipuri de date Java la tipuri de date [SQL .](https://en.wikipedia.org/wiki/SQL)Hibernate oferă, de asemenea, facilități de interogare și recuperare a datelor. Acesta generează apeluri SQL și scutește dezvoltatorul de manipularea manuală și conversia obiectelor setului de rezultate.

Graphical user interface

Description automatically generated with low confidence

Fig 2.2

Pentru testarea aplicatiei am folosit Junit 5.

JUnit este un frameworks [de testare unitară](https://en.wikipedia.org/wiki/Unit_testing) pentru [limbajul de programare Java](https://en.wikipedia.org/wiki/Java_(programming_language)) . JUnit a fost important în dezvoltarea  [test-driven development](https://en.wikipedia.org/wiki/Test-driven_development), și face parte dintr-o familie de frameworks [de testare unitară care este cunoscută în mod colectiv sub numele de](https://en.wikipedia.org/wiki/Unit_testing)[xUnit](https://en.wikipedia.org/wiki/XUnit) , care a apărut cu [SUnit](https://en.wikipedia.org/wiki/SUnit) .

JUnit este conectat ca [JAR](https://en.wikipedia.org/wiki/JAR_(file_format)) la momentul compilarii. Cea mai recentă versiune a cadrului, JUnit 5, pachetul org.junit.jupiter. Versiunile anterioare JUnit 4 și JUnit 3.

Si ca arhitectura am folosit mvc.

# Justificarea limbajului de programare ales:

Java™ are avantaje semnificative față de alte limbi și medii care îl fac potrivit pentru aproape orice sarcină de programare.

Avantajele Java sunt următoarele:

Java este ușor de învățat.

Java a fost conceput pentru a fi ușor de utilizat și, prin urmare, este ușor de scris, compilat, depanat și învățat decât alte limbaje de programare.

Java este orientat pe obiecte.

Acest lucru vă permite să creați programe modulare și cod reutilizabil.

Java este independent de platformă.

Unul dintre cele mai semnificative avantaje ale Java este capacitatea sa de a trece cu ușurință de la un sistem computerizat la altul. Capacitatea de a rula același program pe multe sisteme diferite este crucială pentru software-ul World Wide Web, iar Java reușește în acest sens, fiind independent de platformă atât la nivel sursă, cât și la nivel binar.

Datorită ușurinței de utilizare, capabilităților multiplatformă și caracteristicilor de securitate Java, acesta a devenit un limbaj de alegere pentru furnizarea de soluții de internet la nivel mondial.

# descrierea diagramelor UML:

Diagrama de clase :

Diagram, schematic

Description automatically generated

Fig 4.1

Diagram, engineering drawing

Description automatically generated

Fig 4.2

In diagrama de clase se poate observa cu usurinta architectura MVC ( MODEL VIEW CONTROLER) astfel, in partea de sus de tot ,Fig4.1, am modelele Transport si Utilizator care tot odata sunt si tabelele din baza de date dar si clasa CompareTransport care se ocupa de compararea obiectelor de tip Transport.

Mai apoi ajungem la nivelul de repo-uri care fac legatura dintre aplicatie si baza de date RepoTransport si RepoUtilizator care implementeaza interfata comuna REPO prin care se comunica cu partea de Presenter.

Urmatorul nivel este cel cu controller, fig 4.2. Pentru fiecare pagina care poate fi deschisa in interiorul aplicatiei am create un controler care se ocupa de afisarea si manipularea datelor din partea de utilizator.

La ultimul nivel am partea de Views care sunt interfata utilizator.Fiecare utilizator implementeaza o interfaza prin care comunica cu presenters.

EERD:



Fig 4.3

Pe partea de baze de date ,fig4.3, avem 2 tabele unul care se ocupa cu persistenta datelor pentru utilizatori care necesita autentificare.

Si o tabela Transport care se ocupa cu persistenta datelor pentru linile de transport in comun. Aceasta tabela va suferi modificari in viitor cand o sa trebuiasca sa generez trasee optime.

User case fig 4.4:

Chart

Description automatically generated

Fig 4.4

Diagrama de activitati este prezentata in figura 4.5. In aceasta diagrama am reprezentat toate activitatiile care se pot efectua in timpul aplicatiei si modul in care modul in care se poate ajunge la ele.

Diagram

Description automatically generated

Fig 4.5

# Descrierea aplicației:

Aplicatia creata are scopul de a fi utilizata într-un oraș pentru determinarea traseului optim utilizând transportul în comun.

La rularea apicatiei utilizatorul-lui ii apare o fereasta in care poate allege daca vrea sa intre ca visitator sau daca vrea sa se autentifice.

Daca intra ca visitator el poate sa vada toate linile de transport public si sa le caute dupa numele lor pe partea de front end , pe partea de back-end se aperaza functia refresh ,fig 5.1, care returneaza un tabel cu toate linile disponibile, sau functia find ,fig 5.2, care cauta o linie specifica

Text

Description automatically generatedText

Description automatically generated

Fig 5.1 Fig 5.2

Daca alege sa se autentifice si foloseste credentialele corecte va fi redirectionat pe pagina care este destinata tipului de utilizator de care tine parte.

Daca este angajat el va putea sa faca toate operatiile crud pe tabela transport dar si sa vizualizele linile de transport in comun. Am ales sa deschid doua ferestre pentru a spori ergonomica aplicatiei.

Daca este manager se intampla exact la fel ca pe partea de angajat doar ca utilizatorul va putea aplica operatii crud pe tabela utilizatori

# Bibliografie:

<https://www.javatpoint.com/hibernate-tutorial>

https://www.postgresql.org/