ACADEMIA DE STUDII ECONOMICE BUCURESTI

FACULTATEA DE CIBERNETICA STATISTICA SI INFORMATICA ECONOMICA

MASTER SIMPRE

SISTEM DE GESTIUNE A RELATIEI CU CLIENTII IN TELECOMUNICATII

COORDONATOR STIINTIFIC MASTERAND:

CONF.DR. CARMEN TIMOFTE STAN NICOLAE CRISTIAN

BUCURESTI

2019

**Cuprins**

**1.Introducere**

1.1 Ce este CRM-ul si de ce este important ?................................................... 3

1.2 Tipuri de soluții CRM ……..……………………………………………………………………….4

1.3 Avantaje ale CRM-ului …….………………………………………………………………………4

1.4 Produse similare pe piata si studiu comparativ……………………………………….5

**2. Tehnologii utilizate**

2.1 Limbajul de programare Java……………………………………………………………………9

2.2 Sistemul de gestiune al bazelor de date relationare MySQL………………………………………….14

**3. Implementarea solutiei**

3.1 Schema logica……………………………………………………………………………………………………………….19

3.2 Schema bazei de date…………………………………………….……………………………………………………..20

3.3 Descrierea aplicatiei……………………………………………………………………………………………………….21

**Concluzii**………………………………………………………………………………………….30

**Anexe** ………………………………………………………………………….……………….31

**Bibliografie**……………………………………………………………………………………..32

**1. Introducere**

**1.1** **Ce este CRM-ul si de ce este important ?**

CRM sau Customer Relationship Management este o strategie pentru gestionarea relațiilor și interacțiunilor unei organizații cu clienții și potențialii clienţi. Un sistem CRM ajută firmele să rămână în legatură puternică cu clienți, să îmbunătățească procesele de afaceri și profitabilitatea organizaţiei.Acesta este mai importantă acum decât oricând pentru companii, deoarece ajută la câștigarea de noi clienți și păstrarea celor existenți.

Atunci când oamenii vorbesc despre CRM, ei se referă de obicei la un sistem CRM, un tool care este folosit pentru gestionarea contactelor, managementul vânzărilor, productivitate și multe altele. Scopul unui sistem CRM este simplu: Îmbunătățirea relațiilor de business.

Acesta îi ajută pe utilizatori să se concentreze pe relațiile companiei cu persoanele individuale, inclusiv clienți, utilizatori de servicii, colegi sau furnizori.

Când oamenii vorbesc despre CRM, ar putea însemna oricare din cele de mai jos:

CRM ca tehnologie: Acesta este un produs tehnologic, adesea prezent în cloud, pe care echipele îl utilizeaza pentru a înregistra, raporta și analiza interacțiunile dintre organizaţie și utilizatori. Acesta este numit un sistem sau o soluție CRM.

CRM ca o strategie: aceasta este o filozofie a afacerilor cu privire la modul în care trebuie gestionate relațiile cu clienții și potențialii clienți

CRM ca proces: Gândiți-vă la acest lucru ca pe un sistem pe care o organizaţie îl impelementează pentru a gestiona aceste relații.

Software-ul CRM înregistrează informații de contact ale clienților, cum ar fi e-mail, telefon, profil social media al site-ului web și multe altele. De asemenea, poate atrage automat alte informații, cum ar fi știrile recente despre activitatea companiei și poate stoca detalii cum ar fi preferințele personale ale clientului în domeniul comunicațiilor.

Sistemul CRM organizează aceste informații pentru a arata o evidență a persoanelor și a companiei, astfel încât să puteți înțelege mai bine relația oraganizaţiei în timp.

Software-ul îmbunătățește managementul relațiilor cu clienții, făcând o viziune de 360 de grade asupra clientului, captând interacțiunile cu afacerea și afișând informațiile necesare pentru a avea relaţii mai bune cu clienții.

Acesta a apărut la sfârșitul anilor '90 ca o strategie de afaceri centrata în jurul colectării și manipularii informațiilor referitoare la relaţiile cu clienţi și tranzacțiile acestora, susținute de o soluţie informatica.

CRM este o soluție suplimentară pentru cele mai obișnuite aplicații de bază pentru extragerea de valoare adăugată de la datele existente ale clienților.

**1.2** **Tipuri de soluții CRM**

Colaborativ: vizează schimbul de date privind diferitele tipuri de interacțiuni pe care clienții le au cu compania și departamentele sale, fie prin interacțiuni directe, prin mail sau prin fax.

Analitic: Funcțiile prezinta analiza performanțelor afacerilor. CRM-ul analitic vă permite să lucrați cu toate informațiile specifice despre cum vă place diferitele segmente de clienți a fi trataţi.

Operațional: aplicația interacționează direct cu clientul prin integrarea front officeului, a back officeului. Acest tip include domeniile funcționale ale automatizării forței de vânzări, managementului contactelor, automatizării activități de marketing și servicii pentru clienți.

**1.3 Avantaje ale CRM-ului**

Necesitatea folosirii acestuia este datorita urmatoarelor avantaje aduse:

Învăţare. CRM ajută companiile să învețe despre clienții lor, inclusiv cine sunt și de ce achiziționează produsele, precum și tendințele în istoria achizițiilor clienților. Acest lucru permite firmelor să anticipeze mai bine nevoile clienților și prin urmare, să le îndeplinească. Utilizarea eficientă a managementului relațiilor cu clienții poate oferi un avantaj strategic. Datele organizate bine despre clienț ajută organizaţiile să selecteze destinatarii pentru promoții și oferte noi.

Organizarea. CRM permite companiilor să devină mai eficiente prin organizarea și automatizarea anumitor aspecte ale afacerii. De la procese de vânzări la campanii de marketing și analiză de afaceri, precum și date despre clienți, CRM automatizează și simplifică aceste procese pentru companii. Acest lucru permite companiilor să organizeze aceste procese în date mai simple, mai ușor de înțeles.

Optimizare. În cele din urmă, software-ul CRM permite companiilor să își optimizeze interacțiunile cu clienții. Prin simplificarea și eficientizarea multora dintre procesele mai complexe de interacțiune cu clienții, CRM mărește satisfacția clienților.

Eficiența și productivitatea. Un CRM bine implementat este mai mult decât probabil să vă îndrepte ineficiențele operaționale, mai ales datorită capacității sale de descoperire a datelor și integrărilor puternice cu infrastructura software actuală.

Disponibilitatea datelor. Datele sunt cele mai prețioase și strategice pentru CRM, astfel este cel mai bun pariu pentru a strânge aceste date într-un singur loc în care vor fi analizate. Ideea nu este doar de a vedea datele, ci de a înțelege despre ce este vorba, iar CRM-ul face acest lucru cu ajutorul a o mulțime de instrumente analitice integrate.

Îmbunătățirea responsabilității. Atunci când relația cu clienții nu merge bine, asta indică faptul că restul in organizaţie nu funcţionează bine. Aici CRM-urile sunt cele mai atractive - ele simplifică comunicarea cu clienții și fac din afacere să fie mai de încredere și mai responsabilă. Ceea ce face este să-i ajute pe angajați să-și înțeleagă îndatoririle, dar mai ales să înțeleagă greșelile organizaţiei.

O mai bună colaborare. Foile de calcul nu sunt funcționale. Cu toate acestea, suntem de acord cu anumite funcționalități pe care trebuie să le oferim, deoarece nu avem instrumentul potrivit pentru a face acest lucru. CRM-urile, pe de altă parte, sunt bazate pe cloud, ceea ce este suficient pentru a garanta faptul că informațiile despre clienți vor fi accesibile fiecărui departament.

Îmbunătățirea comunicării cu clienții. CRM-urile sunt unul dintre puținele sisteme care se dovedesc a fi booster-ul deținerii clienților, deoarece urmăresc relația companiei cu fiecare client în parte.

**1.4 Produse similare pe piata si studiu comparativ:**

**Saleforce** se remarca prin:

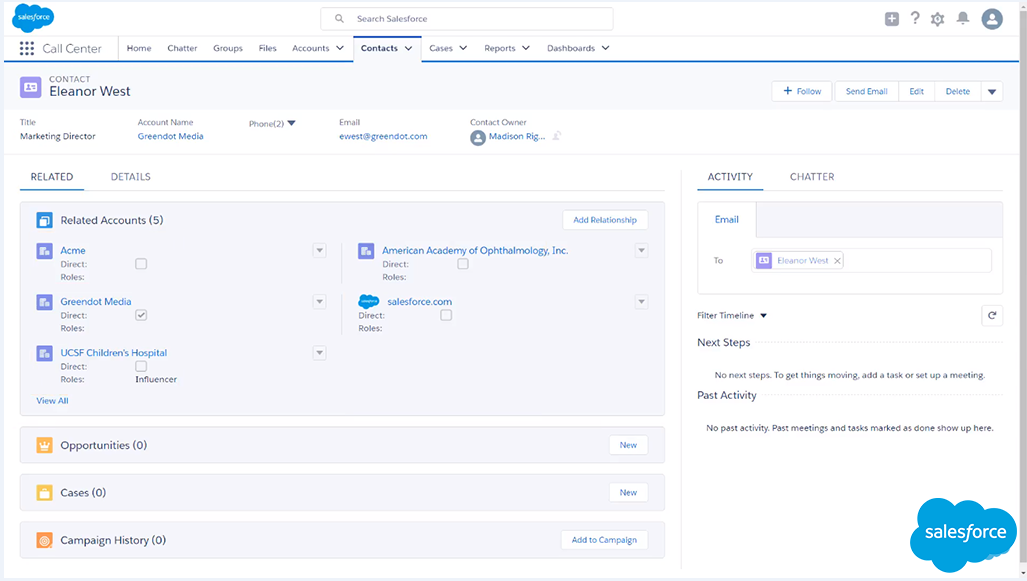
Gestionarea oportunităților foarte competitive. Salesforce Essentials încorporează un modul special de colaborare în vânzări și un modul de gestionare a vânzărilor, ambele dezvoltate pentru a ajuta stabilirea obiectivelor bazate pe metrici și pentru amplificarea performanțelor reprezentanților de vânzări pentru a motiva o competiție sănătoasă în echipa.

Generarea activă a calitatii de conducere. Cu Salesforce Essentials se poate reduce durata perioadei de vanzare. Acest sistem permite urmărirea clickurilor și crearea de campanii inteligente oferiind aplicații dedicate pentru automatizarea marketingului.

Procesele de afaceri personalizate. Viteza de lucru Visual Sales Essentials vă permite să editați cu ușurință procese de afaceri, inclusiv oferte, discounturi și cheltuielile. Același instrument poate fi folosit pentru a partaja, sincroniza fișierele și a urmări schimbările în timp real astfel îmbunătățindu-se comunicarea intre membrii echipei.

Analiza detaliata și raportarele. Ca și în cazul majorității produselor Salesforce, dashbordurile inteligente ale CRM sunt capabile să colecteze și să acumuleze date pentru rapoarte și analize detaliate. Folosind modelul potrivit, veți avea posibilitatea de a gestiona teritorii, de a stabili granițe și volume ale clienților. În figura 1 se poate observa principalele meniuri ale aplicaţiei.

Implicați clienții oriunde v-ați afla. Salesforce Essentials oferă aplicații dedicate utilizatorilor Android și iOS, permițând să ajute clienții și să utilizeze oportunitățile chiar și atunci când nu se află la birou. Fiind un sistem cu adevărat optimizat pentru dispozitive mobile, Salesforce Essentials nu impune restricții asupra tipurilor de dispozitive sau a sistemelor de operare pe care membrii echipei le vor utiliza.

Figura 1 - Interfata Saleforce

|  |  |  |  |
| --- | --- | --- | --- |
| **PREZENTARE**  **GENERALA** |  |  |  |
| **NUME PRODUS** | **SAP CRM** | **Salesforce** | **Oracle Customer Experience** |
| **ACCES API** |  |  |  |
| **CONTACT MANAGEMENT** |  |  |  |
| **CONTRACT MANAGEMENT** |  |  |  |
| **DATABASE MANAGEMENT** |  |  |  |
| **INTEGRARE** |  |  |  |
| **RAPORTARE**  **PERFORMANTA** |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTROLUL PRODUSELOR& PRETURILOR** |  |  |  |
| **ASIGNARE**  **TASKURI** |  |  |  |
| **CONFIGURARE**  **OFERTA SI PRET** |  |  |  |
| **MARKETING AUTOMATION** |  |  |  |
| **SOCIAL RELATIONSHIP MANAGEMENT** |  |  |  |
| **NUMARUL DE CLIENTI** |  |  |  |
| **MIC (1-50 UTILIIZATORI)** |  |  |  |
| **MEDIU (51-200 UTILIZATORI)** |  |  |  |
| **ALTE**  **TRASATURI** |  |  |  |
| **VALUTA DIFERITA** |  |  |  |
| **PERSONALIZABIL** |  |  |  |
| **MOBILE FEATURES** |  |  |  |
| **IOS APP** |  |  |  |
| **ANDROID APP** |  |  |  |
| **NATIVE**  **WEB APP** |  |  |  |

**2. Tehnologii utilizate**

**2.1 Limbajul de programare Java**

**2.1.1 De ce Java?**

Am ales Java pentru ca se pot dezvolta cu uşurtintă aplicaţii desktop datorită funcţionalităţilor aduse de AWT si Swings. AWT vine de la Abstract Window Tookit şi este folosit pentru a dezvolta aplicaţii desktop User Interface. Putem dezvolta orice tip de aplicaţie desktop cu ajutorul celor doua pachete precizate. IDEul Netbeans are incorporat şi drag and drop ce face realizarea de aplicaţii mult mai uşoară.

Pentru dezvoltarea aplicațiilor Enterprise, Java este prima opțiune datorită caracteristicilor sale puternice care se potrivesc cerințelor. În lumea de astăzi, cea mai mare parte a aplicaţiilor enterprise se bazează pe aplicațiile Java datorită faptului că este cel mai sigur, puternic si scalabil limbaj de programare. Aceste aplicaţii necesitată prea multă securitate pe care o poate îndeplinii doar acest limbaj. Java are caracteristici puternice de gestionare a memoriei, șterge automată a memoriei neutilizată, care îmbunătățește performanța aplicatiilor. Aplicațiile Java pot fi ușor scalabile în cazul creșterii numărului de persoane de pe aplicație pentru întreprinderi. Java este limbajul cel mai sigur, motiv pentru care majoritatea aplicațiilor bancare sunt dezvoltate pe această platforma. Acesta oferă un mediu multithreaded care ajută la rularea mai multor fire de executie, ajutând la îmbunătățirea performanței unui sistem, acesta fiind motivul pentru care acest limbaj este superior altor limbaje de programare (|Educba2019|).

**2.1.2 C# vs Java**

S-ar mai fi putut alege C# fiind un limbaj de programare orientat pe obiecte, funcțional, generic. Acesta a fost dezvoltat de Microsoft cu inițiativa .NET.

C# este folosit pentru a construi o varietate de aplicații Este deosebit de puternic pentru a construi aplicații desktop și jocuri Windows. Dezvoltarea web poate fi făcută eficient și cu C# și devine tot mai popular pentru aplicatiile mobile. Deci se face o alegere excelentă pentru orice programator care dorește să facă aplicaţii web și dezvoltare de gaming. Există diferite instrumente cross-platform disponibile care permit aplicațiilor scrise în C# să potă fi utilizate pe mobil și pe desktop (|Educba2018|).

**2.1.3 Despre Java**

Doar câteva limbaje de programare au redesenat fundamental esența programării. În acest grup de elită, Java se remarcă prin faptul că impactul a fost rapid și larg răspândit. Versiunea originală lanstată Java 1.0 în 1995 de către Sun Microsystems a provocat o revoluție în programare. Această revoluție a transformat radical Web-ul într-un mediu foarte interactiv. În acest proces, Java stabilește un nou standard în proiectarea limbajelor de programare (|SCHI2014|).

De-a lungul anilor, Java a continuat să crească, să evolueze și să se imbunatatească. Spre deosebire de multe alte limbaje de programare, care sunt lente în încorpora de noi caracteristici, Java a fost adesea prima in ceea ce priveşte dezvoltarea limbajelor de programare. Un motiv pentru aceasta este cultura inovării și a schimbării care a venit în jurul Java. Ca rezultat, Java a trecut prin mai multe upgrade-uri unele relativ mici, altele mai semnificative.

Creșterea Internetului și a World Wide Web a modificat fundamental computerele. Înainte de Web, peisajul a fost dominat de PC-uri stand alone. Astăzi, aproape toate computerele sunt conectate la Internet. Internetul, în sine, a fost transformat - inițial oferind o modalitate convenabilă de a partaja fișiere și informații. Astăzi este un univers vast. Cu aceste schimbări a venit o nouă modalitate de programare: Java.

Java nu este doar un limbaj de programare al Internetului este mult mai mult de atât. Java a revoluționat programarea, schimbând modul în care ne gândim atât la forma, cât și la funcționalitatea unui program. A fi un programator profesionist implică în prezent capacitatea de a programa în Java.

Inovarea în limbajul de programare a fost realizată de doi factori: îmbunătățiri în arta programării și schimbări în mediul de calcul. Java nu face excepție. Bazându-se pe moștenirea bogată moștenită de la C și C ++, Java adaugă rafinamente și caracteristici care reflectă stadiul actual al tehnicii în programare. Răspunzând la creșterea mediului online, Java oferă caracteristici care simplifică programarea pentru o arhitectură distribuită.

Java a fost conceput de James Gosling, Patrick Naughton, Chris Warth, Ed Frank și Mike Sheridan la Sun Microsystems în 1991. Java a fost denumit inițial "Oak", dar a fost redenumit "Java" în 1995. Oarecum surprinzător nu Internetul a dus la realizarea limbajului de programare. În schimb, motivația principala a fost nevoia unui limbaj independent de platformă care ar putea fi folosit pentru a crea software pentru a fi încorporate în diverse dispozitive electronice de consum, cum ar fi prăjitoare de pâine, cuptoare cu microunde și telecomenzi. După cum probabil puteți ghici, multe tipuri de procesoare sunt folosite ca controlleri. Problema a fost că (la vremea respectivă) majoritatea limbajelor de programare au fost concepute pentru a fi compilate pentru o anumită țintă. De exemplu, să luam în considerare C ++. Deși a fost posibilă compilarea unui program C ++ pentru aproape orice tip de procesor, pentru a face acest lucru a fost necesar un compilator C ++ orientat pentru acel procesor. Problema, totuși, este că compilatoarele sunt costisitoare și consumatoare de timp pentru a fi create. Într-o încercare de a găsi o soluție mai bună, Gosling și alții au lucrat pe un limbaj portabil, cross-platform, care ar putea produce cod care să ruleze pe o varietate de procesoare în medii diferite. Acest efort a dus în cele din urmă la realizarea Java.

Despre momentul în care au fost elaborate detaliile despre Java, a apărut un al doilea factor și în cele din urmă mai important care ar juca un rol crucial în viitorul Java. Această a doua forță a fost, desigur, World Wide Web. Dacă Web-ul nu ar fi luat forma la aproximativ același timp în care a fost implementat Java, Java ar fi rămas un limbaj folositoar, dar obscur, pentru programarea electronicii de consum. Cu toate acestea, odată cu apariția webului, Java a fost propulsat în prim planul designului de limbaj computerizat, pentru că și Webul a solicitat programe portabile.

Majoritatea programatorilor învață devreme în carieră că limbajele de programare portabile sunt la fel de evazive pe cât sunt de dorit. În timp ce căutarea unei modalități de a crea programe eficiente, portabile (independente de platformă) este aproape la fel de veche ca și disciplina de programare în sine, ea a luat locul din spate pentru alte probleme mai presante. Cu toate acestea, odată cu apariția internetului și a web-ului, vechea problemă a portabilității s-a întors ca o răzbunare. La urma urmei, Internetul constă într-un univers divers, distribuit, populat cu mai multe tipuri de calculatoare, sisteme de operare și CPU-uri.

Ceea ce a fost odată o problemă iritantă, dar cu prioritate scăzută, a devenit o necesitate de mare interes. Până în 1993, a devenit evident pentru membrii echipei de proiectare Java că problemele de portabilitate întâlnite frecvent la crearea codului pentru controllere încorporate se găsesc și atunci când încercați să creați cod pentru Internet. Această realizare a determinat concentrarea Java să treacă de la electronice de consum la programare pe Internet. Deci, în timp ce a fost dorința pentru un limbaj de programare neutru în arhitectură, care a oferit scânteia inițială, Internetul a dus în final la succesul Java la scară largă.

Java este în stransă legatură atât cu C, cât și cu C ++. Java își moștenește sintaxa de la C. Modelul de programare orientat obiect este adaptat din C ++. Relația Java cu C și C ++ este importantă din mai multe motive. În primul rând, mulți programatori sunt familiarizați cu sintaxa C / C ++. Acest lucru îl face ușor pentru un programator C / C ++ să învețe Java și invers, pentru un programator Java pentru a învăța C / C ++.

În al doilea rând, designerii Java nu au "reinventat roata". În schimb, ei au mai imbunătăţit paradigma de programare deja de mare succes. Epoca modernă a programării a început cu C. S-a mutat la C ++, iar acum Java. Prin moștenirea și construirea acestui patrimoniu bogat, Java oferă un mediu de programare puternic, logic și consistent, care ține de cele mai bune caracteristici din trecut și adaugă unele noi cerute de mediul online. Poate cel mai important, datorita asemănărilor dintre C, C ++ și Java, acestea definesc un framework comun pentru programatorii profesionisti astfel acestia nu se confruntă cu o rupture majora atunci când trec de la un limbaj de programare la altul.

Una dintre filosofiile de proiectare centrale ale lui C și C ++ este aceea că programatorul are puterea! Java de asemenea moștenește această filozofie. Cu excepția acelor constrângeri impuse de Internet, Java oferă programatorului controlul deplin.

Java are un alt atribut în comun cu C și C ++: a fost proiectat, testat și rafinat de programatori din mediu de lucru. Este un limbaj axat pe nevoile și experiențele oamenilor care au conceput-o. Nu există o modalitate mai bună de a produce un limbaj de programare mai bun. Datorita asemănărilor dintre Java și C ++, în special suportul lor pentru programarea orientată pe obiect este tentant să credem că Java este pur și simplu "versiunea de Internet a C ++". Cu toate acestea, ar fi o greșeală.

Java are diferențe practice și filozofice semnificative. Deși Java a fost influențat de C ++, nu este o versiune îmbunătățită a C ++. Desigur, asemănările cu C ++ sunt semnificative, și dacă ești un programator C ++, vă veți simți ca acasă cu Java. Un alt punct: Java nu a fost conceput pentru a înlocui C ++. Java a fost conceput pentru a rezolva un anumit set de probleme. C ++ a fost conceput pentru a rezolva un set diferit de probleme. Limbajele vor coexista mulți ani.

**2.1.4 Coexistenta dintre Java si C#**

La câțiva ani după crearea Java, Microsoft a dezvoltat limbajul C#. Acest lucru este important deoarece C# este strâns legat de Java. De fapt, multe dintre caracteristicile C# sunt la fel cu cele ale limbajului Java. Atât Java cât și C # au aceeași sintaxă generală în stilul C++, suportă programarea distribuită și utilizează același model de programare obiect. (|WILL2005|)

Există, desigur, diferențe între Java și C#, dar aspectul general al acestor limbaje de programare este foarte asemănător. Aceasta înseamnă că dacă cunoașteți deja C#, atunci învățarea Java va fi deosebit de ușoară. În schimb, dacă C# este în viitorul tău, atunci cunoștințele tale despre Java vor fi foarte folositoare. Având în vedere asemănarea dintre Java și C#, s-ar putea întreba în mod firesc: "Va înlocui C# pe Java?" Răspunsul este nu. Java și C# sunt optimizate pentru două tipuri diferite de medii de calcul. La fel cum C ++ și Java vor coexista pentru o lungă perioadă de timp, asa și C# și Java.

Toate limbajele de programare oferă abstractizare. Se poate argumenta că complexitatea problemelor pe care le puteți rezolva este direct legată de natura și calitatea abstractizării. Limbajul de asamblare este o mică abstractizare a mașinii de bază. Multe limbaje așa-numite "imperative" care au urmat (precum FORTRAN, BASIC și C) au fost abstracții ale limbajului de asamblare.

Aceste limbaje sunt îmbunătățiri majore față de limbajul de asamblare, însă abstractizarea lor primară necesită în continuare să vă gândiți mai degrabă la structura calculatorului decât la structura problemei pe care încearci să o rezolvi. Programatorul trebuie să stabilească asocierea dintre modelul mașinii (în spațiul "soluție", locul în care implementați acea soluție, cum ar fi un computer) și modelul problemei care este de fapt rezolvată (în " spațiu problemă ", care este locul în care există problema, cum ar fi o afacere). Efortul necesar pentru a efectua această asociere produce programe care sunt greu de scris și costisitor de întreținut.

Abordarea orientată pe obiect merge mai departe prin furnizarea de instrumente pentru ca programatorul să reprezinte elementele în spațiul problemei. Această reprezentare este destul de generală încât programatorul nu este constrâns de un anumit tip de problemă. Referirea se face la elementele din spațiul problemei și la reprezentările lor în spațiul soluției ca "obiecte". Ideea este că programului ii este permis să se adapteze la probleme prin adăugarea de noi tipuri de obiecte, așa că atunci când citiți codul care descrie soluția, citiți cuvintele care exprimă și problema. Aceasta este o abstractizare mai flexibilă și mai puternică decât cea pe care am avut-o înainte. Astfel, programarea orientate obiect vă permite să descrieți problema în termenii ei, mai degrabă decât în ceea ce privește computerul în care va funcționa soluția. Există încă o conexiune înapoi la computer: fiecare obiect pare destul de mic ca un mic computer - are o stare și are operații pe care le puteți cere să le îndeplinească. Totuși, aceasta nu pare a fi o analogie rea cu obiectele din lumea reală - toate au caracteristici și comportamente.

**2.1.5 Avantajele Java**

Simplu: Java a fost conceput pentru a fi ușor de folosit, scris, compilat, depanat și învățat fata de alte limbje de programare. Java este mult mai simplu decât C ++, deoarece Java utilizează alocarea automată a memoriei și folosirea garbage collectionului.

Object-Oriented: Programarea orientată obiect este asociată cu concepte precum clasă, obiect, moștenire, încapsulare, abstracție, polimorfism care permite crearea de programe modulare și cod reutilizabil. Se pot declara clase, crea obiecte în interiorul clasei și interacționa între două obiecte.

Platformă Independenta: Java oferă confortul programului de a scriere o singură dată și rulează pe orice platformă hardware și software și orice browser compatibil Java. Aceasta oferă posibilitatea de a se deplasa cu ușurință de la un sistem de calcul la altul.

Sisteme distribuite: Java are o capacitate mare de conectare la rețea, este concepută pentru a face computerele distribuite cu capacitatea de rețea care este în integrată în ea.

Secure: Java este primul limbaj de programare care include securitatea ca parte integrantă a designului. Compilatorul Java, interpretul și mediul de execuție au fost dezvoltate fiecare în funcție de securitate. Java Virtual Machine are un identificator unic care identifică octetul și îl verifică înainte de al rula.

Alocare: Java are caracteristica sistemului de alocare a stivei. Urmează LIFO (Last in First Out), care ajută datele să fie stocate și recuperate cu ușurință.

Multithreaded: Java este unul din limbile de programare care suportă Multithreading. Multithreading este capacitatea unui program de a efectua simultan mai multe sarcini în cadrul unui program.

API-uri bogate: Java oferă diverse API-uri pentru dezvoltarea aplicațiilor. Java API (Application Programming Interface) este setul de comenzi sau metode de comunicare între diverse activități cum ar fi conexiunea bazei de date, rețele, I/O, parsarea XML, utilitare și multe altele.

Instrumente de dezvoltare rapidă Opensource: În cursul anului, mai multe instrumente de dezvoltare open source, adică IDE-uri precum Eclipse și Netbeans, au fost create cu Java ca bază care face Java mai puternic pentru dezvoltarea aplicațiilor. IDE-urile simplifică dezvoltarea aplicațiilor cu caracteristici puternice de codificare și depanare.

Robust: Java este unul dintre cele mai robuste limbaje de programare. Compilatoriul Java poate detecta orice eroare în codificare. Există caracteristici cum ar fi manipularea excepțiilor și garbage collectionul care face Java să fie mai robust.

**2.2 Sistemul de gestiune al bazelor de date relationare MySQL**

**2.2.1 Despre bazele de date**

Bazele de date moderne au apărut în anii 1960 datorită cercetării de la IBM, printre alte companii. Cercetarea s-a axat în principal pe automatizare în special pe automatizarea sarcinilor de stocare și indexare a datelor, care anterior necesitau o mare parte a muncii manuale. Puterea de calcul și stocarea au devenit mult mai ieftine, ceea ce face ca utilizarea computerelor pentru indexarea și stocarea datelor să fie o soluție viabilă. Un pionier în domeniul bazei de date a fost Charles W. Bachman, care a primit premiul Turing în 1973 pentru o activitate de pionierat în tehnologia bazelor de date. În 1970, un cercetător de la IBM numit Ted Codd a publicat primul articol despre bazele de date relaționale.

Deși IBM a fost lider în cercetarea bazei de date, Honeywell Information Systems, Inc. a lansat un produs comercial în 1976 pe baza acelorași principii ca și sistemul de informații IBM.

La începutul anilor 1980, primul sistem de baze de date SQL au apărut de la companii cum ar fi Oracle, cu Oracle Version 2 și ulterior SQL/DS de la IBM, precum și o serie de alte sisteme de la alte companii.

O bază de date constă din seturi de date structurate, ceea ce înseamnă că o bază de date conține colecții de date. De exemplu, baza de date ar putea conține detaliile scorurilor de golf ale unchiului Bob sau date despre toate cărțile dintr-o bibliotecă. Probabil că nu doriți să amestecați aceste două colecții de date, altfel când doriți să găsiți date despre o carte pe care ar trebui să le analizați sa fie irelevante din cauza rezultatelor de la golf. Pe scurt, bazele de date vă ajută să vă organizați datele. O bază de date stochează colecțiile de date în tabele.

Bazele de date sunt de obicei asociate cu software-ul care permite actualizarea și interogarea datelor. Exemplele de software de baze de date din viața reală includ Microsoft Access, Oracle 10g, MySQL, MySQL AB și Microsoft SQL Server. De multe ori aceste programe sunt denumite baze de date, dar strict vorbind, sunt sisteme de gestionare a bazelor de date (SGBD). O bază de date este seturile (colecții de date conexe) grupate într-o singură entitate. S-ar putea crea o bază de date Access, cu numele MyDatabase, să se includa diferite colecții de date în interiorul acelei baze de date și să se gestioneze cu software-ul MS Access.

La fel ca și în cazul bazei de date Access, o bază de date simplă ar putea fi doar un fișier cu multe înregistrări, fiecare înregistrare fiind împărțită în câmpuri. Dar ce sunt înregistrările și câmpurile? Un câmp este un singur element de date despre un anumit lucru. Un lucru ar putea fi o persoană și un singur element de date despre o persoană ar putea fi data nașterii. Sau lucru ar putea fi adresa unei case și un anumit element de date ar putea fi strada ei. Folosind exemplul de carte, anul în care a fost publicată o carte este un element specific de date care poate fi stocată într-un câmp. Un alt câmp ar putea fi titlul cărții; un altul ar putea fi numele autorului. În mod colectiv aceste câmpuri sunt cunoscute ca înregistrări. Fiecare carte are o înregistrare proprie, iar toate înregistrările sunt stocate colectiv într-o bază de date în ceva numit tabel. O singura bază de date poate conține unul sau mai multe tabele.

Baza de date relațională este o bază de date care conține date organizate și legate între ele. Toate înregistrările dintr-o bază de date sunt organizate în tabele. Datele înrudite, cum ar fi detaliile persoanelor de vânzări, sunt grupate într-un singur tabel.

Cele mai multe sisteme de gestionare a bazelor de date în aceste zile sunt relaționale, denumite sistem relațional de gestionare a bazelor de date (RDBMS). Aceste sisteme fac ca stocarea datelor și returnarea rezultatelor să fie mai ușoară și mai eficientă. Acestea permit să se pună întrebări diferite ale bazei de date - chiar întreabări pe care proiectantul original al bazei de date nu se aștepta să fie puse.

**2.2.2 Avantajele bazelor de date**

Principalul avantaj al bazelor de date este recuperarea rapidă și eficientă a datelor. O bază de date vă ajută să vă organizați datele în mod logic. Sistemele de gestionare a bazelor de date sunt reglate pentru regăsirea rapidă a datelor dorite în modul dorit. De asemenea, bazele de date vă permit să returnati datele in diferite moduri. Preluarea datelor dintr-o bază de date se numește interogare. Interogarea SQL înseamnă pe scurt orice cod SQL care extrage datele din baza de date.

Bazele de date relaționale au avantajul suplimentar de a vă permite să specificați modul în care datele se leaga intre ele.

De asemenea, bazele de date vă permit să configurați reguli care să asigure că datele rămân consistente atunci când adăugați, actualizați sau ștergeți date. Imaginați-vă o companie de vânzări care are doi agenți de vânzări numiți Julie Smith.

Puteți crea o bază de date pentru a vă asigura că fiecare agent de vânzări are un cod unic, numit un identificator unic (astfel încât Julii să nu se amestece); în caz contrar, spunând cine a vândut care mașina s-ar dovedi imposibil.

Alte sisteme de stocare a datelor, cum ar fi fișiere text sau foi de calcul, nu au aceste tipuri de verificări și vă permit din pacate să stocați date gresite. De exemplu, puteți specifica faptul că numărul de securitate social al unui angajat trebuie să fie unic în baza de date. Sau dacă o mașină este vândută și este listată ca fiind vândută de un angajat cu un ID de 123, puteți adăuga un cec pentru a vedea că detaliile complete ale angajatului 123 sunt păstrate într-unul din tabelele bazei de date.

Bazele de date stochează date brute - doar faptele, ca să spunem așa, și nici o informație. O bază de date despre vânzările de mașini ar putea să conțină marca, modelul și prețul fiecărei mașini, dar în mod normal, nu veți stoca numărul mediu de mașini vândute într-o lună, deoarece puteți calcula din informațiile despre vânzările mașinilor.

Cu toate acestea, o foaie de calcul poate conține date prelucrate, cum ar fi mediile și analiza statistică. O bază de date stochează pur și simplu datele și, în general, lasă procesarea datelor către un program frontend sau interfața pe care utilizatorul o vede. Exemple de programe de tip frontend includ o pagină Web care își extrage datele dintr-o bază de date sau un program care se conectează la datele bazei de date și permite utilizatorului să o vizualizeze.

Partajarea datelor este, de asemenea, mult mai ușoară folosind o bază de date. Puteți partaja date între un număr de utilizatori pe același computer sau între utilizatori de pe diferite computere conectate printr-o rețea sau pe Internet. Dacă compania de vânzări de mașini de exemplu are sucursale în New York, Washington și Boston, puteți seta un computer care să conțină o bază de date într-o singură locație, accesibilă tuturor birourilor prin intermediul unei rețele. Acest lucru este posibil deoarece bazele de date au o structură clar definită și aplică normele care protejează datele conținute. Acestea permit mai multor persoane să acceseze simultan baza de date și să schimbe datele stocate; sistemul de gestionare a bazelor de date gestionează modificările simultane. Imaginați-vă potențialul haos dacă ați folosit o foaie de calcul Excel și doi agenți de vânzări schimba datele simultan. Doriți să păstrați ambele seturi de modificări, dar cel care salvează foaia de calcul ultima este persoana ale cărei modificări sunt stocate, suprascriind orice modificări anterioare.

De asemenea, bazele de date fac schimbul de date între diferite sisteme mult mai ușor decât utilizarea formatelor de date proprietare - adică un format specific unui anumit program, producător sau sistem de operare. O foaie de calcul Excel, de exemplu, este ușor de citit pe o mașină Windows cu MS Office, dar este mai mult o provocare de citit pe o mașină UNIX, Macintosh sau Linux, deoarece aceste computere se ocupă de date într-un mod diferit. Chiar și pe o mașină Windows, trebuie să aveți instalat MS Office. Puteți găzdui o bază de date pe un computer central, puneți sistemul de gestionare a bazelor de date acolo și apoi activați accesul prin intermediul unei rețele locale sau prin Internet.

Ca alternativă la bazele de date, fișierele text și foile de calcul au un mare avantaj, care este și slăbiciunea lor: flexibilitatea. Fișierele text nu au reguli reale. Puteți introduce orice date de text doriti oriunde iti place. În mare măsură, foile de calcul sunt aceleași. Puteți solicita utilizatorilor să adauge date într-o structură predefinită, dar nu aveți nicio modalitate reală de a impune o astfel de solicitare. Utilizarea bazelor de date limitează accesul utilizatorilor doar la date și nu permite utilizatorilor să schimbe structura.

Un ultim avantaj semnificativ al bazelor de date este securitatea. Majoritatea sistemelor de gestionare a bazelor de date vă permit să creați utilizatori pentru a specifica diferite niveluri de securitate. Înainte ca cineva să acceseze baza de date, el sau ea trebuie să se conecteze ca utilizator specific. Fiecare utilizator are diferite drepturi și limite. Cineva care întreține baza de date are capacitatea deplină de a edita date, de a schimba structura bazei de date, de a adăuga și de a șterge utilizatori și așa mai departe. Alți utilizatori pot avea abilitatea de a vizualiza datele, dar nu le pot schimba, sau chiar sa se limiteze datele pe care le pot vizualiza. Multe sisteme de gestionare a bazelor de date oferă un nivel granular de securitate, adică sunt foarte specifice cu privire la ceea ce poate face un utilizator. Acestea nu sunt doar o abordare "totul sau nimic" în care utilizatorul are acces sau nu are acces.

Bazele de date sunt folosite aproape pretutindeni. Prelucrarea datelor a jucat un rol important în dezvoltarea computerelor și chiar și astăzi este unul dintre rolurile principale ale acestora. Aproape oricare plimbare normala sau plimbare de afaceri necesită o baza de date undeva de-a lungul drumului. Bazele de date sunt utilizate în mod obișnuit pe computerele personale pentru stocarea datelor utilizate la nivel local, iar bazele de date ale rețelelor de companii stochează și împărtășesc informații la nivelul întregii companii. Internetul a înregistrat o creștere semnificativă a bazelor de date utilizate pentru a face schimb de informații; cele mai multe magazine online de dimensiuni rezonabile utilizează bazele de date. Când vizitați magazine online de orice dimensiune semnificativă, o bază de date oferă, de obicei, toate informațiile privind mărfurile vândute. Mai degrabă decât fiecare pagină creată manual, comercianții mari utilizează un șablon pentru detaliile despre cărți sau CD-uri, iar SQL preia informațiile din baza de date.

Bazele de date sunt excelente pentru stocarea datelor, sistemul de gestionare a bazelor de date oferă modalități de vizualizare a datelor și de obicei software-ul furnizat vă permite să vizualizați datele. Dar cum folosiți datele în afara software-ului de gestionare a bazelor de date? Sistemul de operare, indiferent dacă este vorba de Windows, UNIX, Linux sau Macintosh, oferă modalități de conectare la sistemul de gestionare a bazelor de date și extragere a datelor. Trebuie să scrieți codul de programare pentru a pune în interior o aplicație autonomă pe care utilizatorul o execută pe calculatorul său sau puteți configura o pagină Web pentru a extrage date. Nu sunteți limitat la anumite limbaje, atâta timp cât limbajul vă permite să vă conectați la software-ul de gestionare a bazelor de date.

**2.2.3 Sistemul de gestionare a bazelor de date**

Un sistem de gestionare a bazelor de date (DBMS) este un software de sistem pentru crearea și gestionarea bazelor de date. DBMS furnizează utilizatorilor și programatorilor o modalitate sistematică de a crea, prelua, actualiza și gestiona datele.

Un DBMS permite utilizatorilor finali să creeze, să citească, să actualizeze și să ștearga datele dintr-o bază de date. DBMS funcționează în principal ca o interfață între baza de date și utilizatorii finali sau programele de aplicații, asigurând că datele sunt organizate în mod constant și rămân ușor accesibile.

DBMS gestionează trei lucruri importante: datele, engine-ul bazei de date care permite accesarea, blocarea și modificarea datelor și schema bazei de date, care definește structura logică a bazei de date. Aceste trei elemente fundamentale ajută la asigurarea concurenței, securității, integrității datelor și procedurilor uniforme de administrare. Operațiile tipice de administrare a bazelor de date suportate de DBMS includ managementul schimbării, monitorizarea/reglarea performanțelor, backup-ul și recuperarea.Multe sisteme de gestionare a bazelor de date sunt responsabile de returnări automate, reporniri și recuperări, precum și de înregistrarea și auditul activității.

DBMS este probabil cel mai util pentru a oferi o vizualizare centralizată a datelor care pot fi accesate de mai mulți utilizatori, din mai multe locații, în mod controlat. Un DBMS poate limita datele pe care le vede utilizatorul final, precum și modul în care acel utilizator final poate vizualiza datele, oferind numeroase vizualizări ale unei singure scheme de bază de date. Utilizatorii finali și programele software nu pot să înțeleagă unde sunt localizate fizic datele sau pe ce tip de suport de stocare se află, deoarece DBMS se ocupă de toate solicitările.

SGBD-ul folosit de mine este MySQL Workbench. MySQL Workbench este un instrument grafic pentru lucrul cu serverele și bazele de date MySQL.

**2.2.4 Funcționalitatea MySQL Workbench acoperă urmatoarele:**

Dezvoltare SQL: vă permite să creați și să gestionați conexiuni la serverele de baze de date. Înainte de a vă permite să configurați parametrii conexiunii, MySQL Workbench oferă posibilitatea de a executa interogări SQL pe conexiunile bazei de date utilizând editorul SQL încorporat.

Modelarea datelor (Design): vă permite să creați modele de schemă de bază de date, în mod grafic, reverse si forword engineer între o schemă și o bază de date live și să editați toate aspectele bazei de date folosind editorul de tabele complet. Editorul de tabel oferă facilități ușor de utilizat pentru editarea tabelelor, coloanelor, indexurilor, triggerilor, partiționării, opțiunilor, inserturilor și privilegiilor, rutinelor și viewurilor.

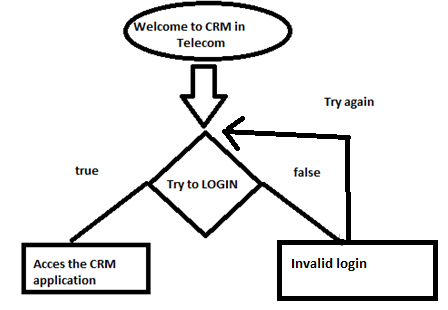
Administrare server: Permite administrarea instanțelor serverului MySQL prin administrarea utilizatorilor, efectuarea de backup și recuperare, inspectarea datelor de audit, vizualizarea sănătății bazei de date și monitorizarea performanței serverului MySQL.

Migrarea datelor: Vă permite să migrați de la Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL și alte tabele, obiecte și date RDBMS către MySQL. Migrarea de la versiunile anterioare ale MySQL la cele mai recente versiuni.

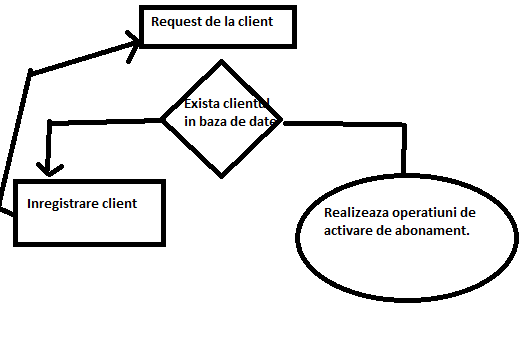
**3. Implementarea aplicatiei**

**3.1 Flow-ul pentru logarea în sistem şi activarea unui abonament**

Logarea în sistem presupune introducerea credidenţialelor şi compararea cu ce este în baza de date.



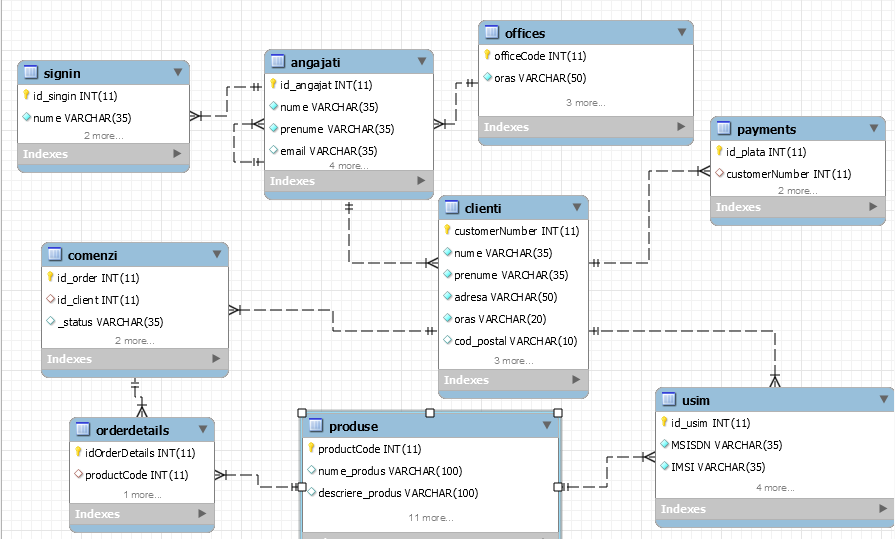
Aici putem observa ce se întâmpla în momentul în care se doreşte realizarea unei operaţiuni în baza de date.



**3.2 Schema bazei de date**

În figura 16 se poate observa schema bazei de date cu toate tabelele aplicatiei si legaturile dintre acestea.

Figura 16 – Schema bazei de date



**3.3 Descrirea aplicatiei**

Accesul la sistem se face prin intermediul unei interfeţe de Login prezente mai jos in figura 2. Se introduce numele si parola verificandu-se cu baza de date MySQL daca sunt corecte credidenţialele respective.

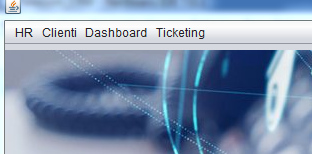
Figura 2 – Interfaţa de Logare



Daca s-au introdus corect credidenţialele se deschide o noua fereastră in care este prezent meniul principal care se poate observa si in figura 3. Aici vom observa mai multe functionalităţi ale aplicaţiei prezente în partea de sus pe MenuBar:

* HR
* Clienti
* Dashboard
* Ticketing

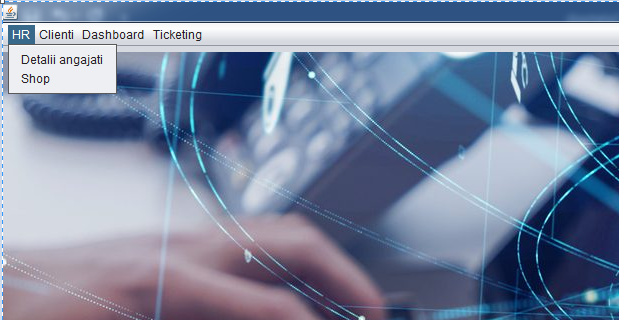
Figura 3 –Meniul principal



Pentru Modulul HR avem 2 opţiuni mai jos in figura 4:

* Detalii angajaţi
* Shop Menu

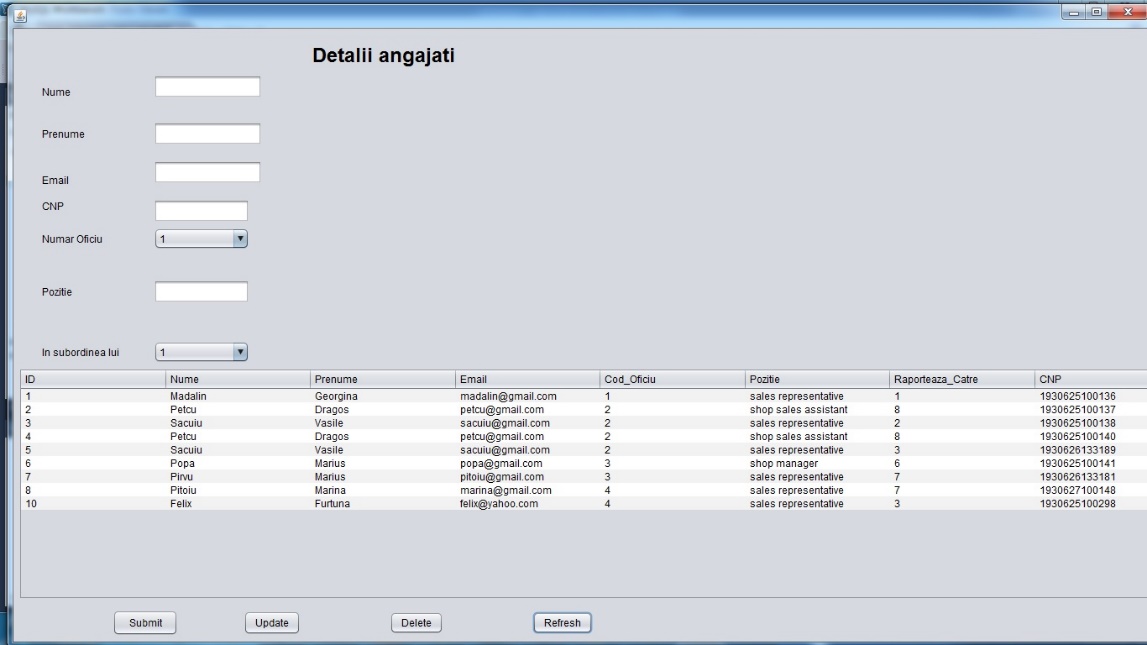
Figura 4 – Modulul HR



Funcţionalitatea Detalii angajati ne va permite sa realizam o serie de modificări referitoare la angajaţi realizând adăugarea, modificarea sau ştergerea acestora din companie.

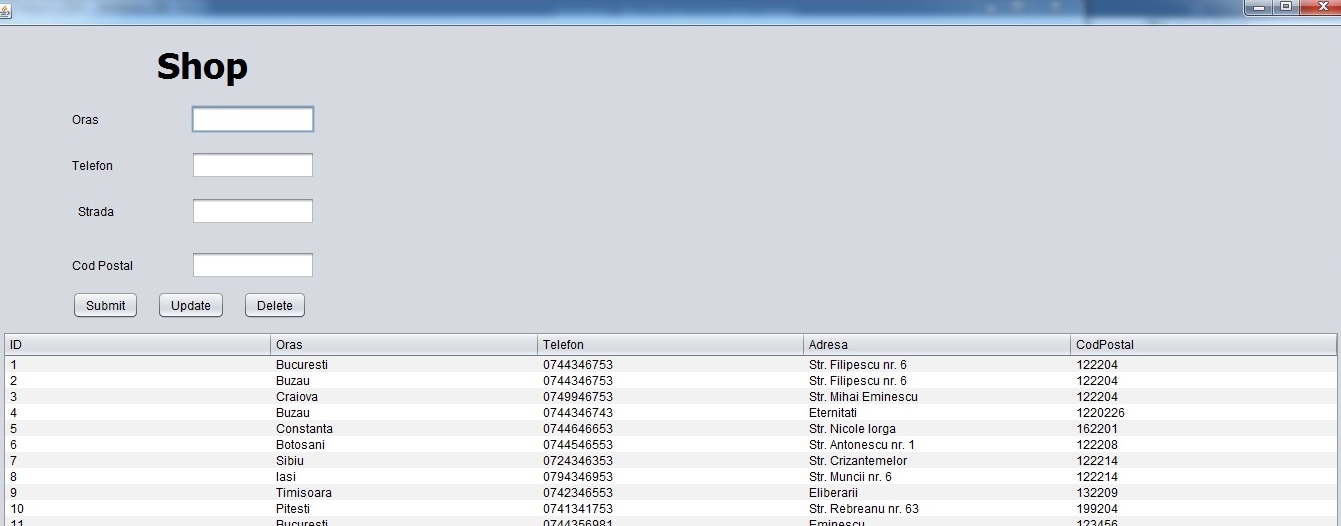
Desigur aceasta opţiune este recomandat să aiba acces doar persoanele de conducere, manageri si persoanele care se ocupa de recrutarea personalului.

Figura 5 – Detalii angajati



O altă funcţionalitate prezentă tot in meniu HR este aceea referitoare la magazine prezenta in figura 6.

Figura 6- Magazine

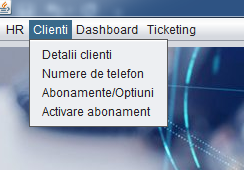


De aici putem adăuga noi magazine daca firma a deschis vreun sediu nou, se pot realiza mici modfiicări ale magazinelor prezente sau ştergerea din evidenţă a vreunui magazin daca nu mai este prezent în firma, a fost vândut sau din alte motive. De asemenea există şi un tabel unde putem afla mai multe informaţii despre magazinele prezente ale firmei cum ar fi adresa, numarul de telefon la care poate fi contactaţi angajaţii companieni din respectivul shop.

Pentru modulul Clienţi sunt prezente mai multe functionalităţi:

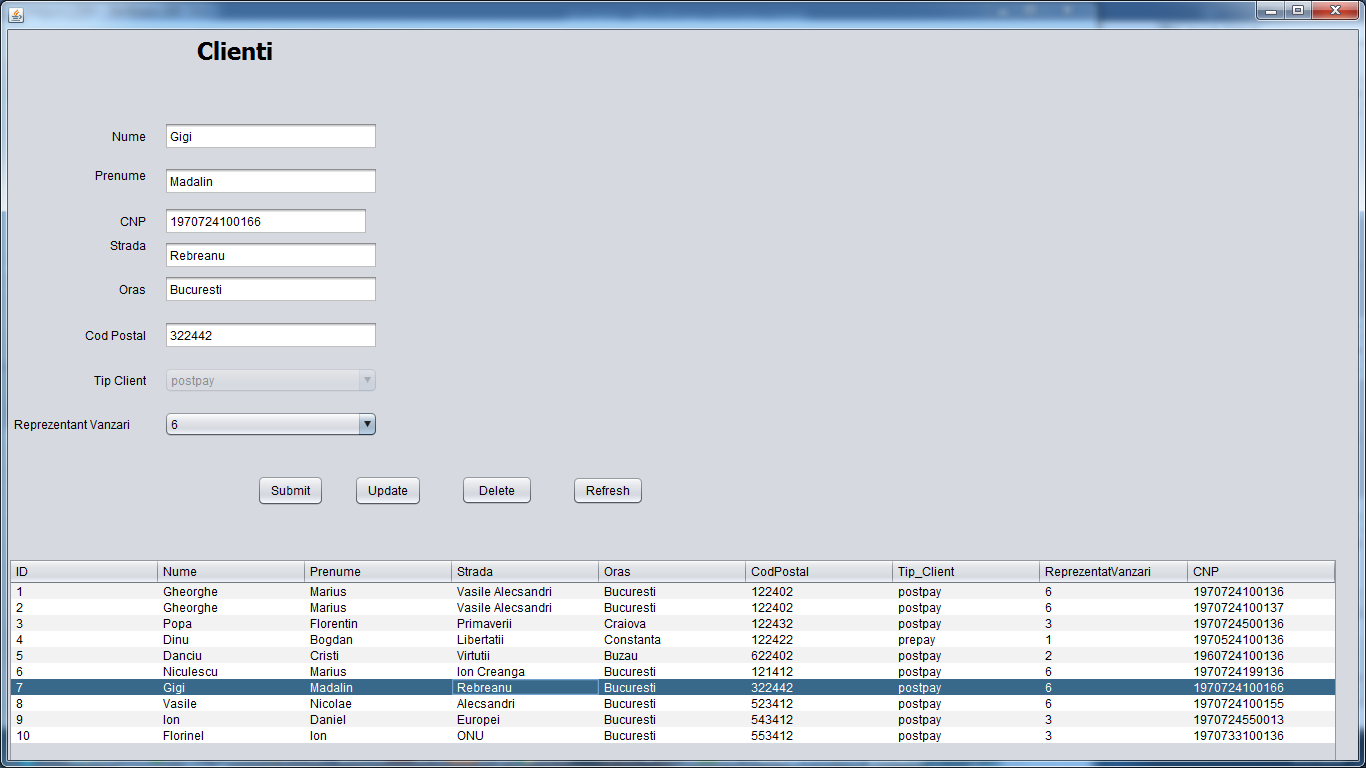
* Detalii clienti
* Numere de telefon
* Abonamente/Optiuni
* Activare abonament

Figura 7 Menu Bar clienti



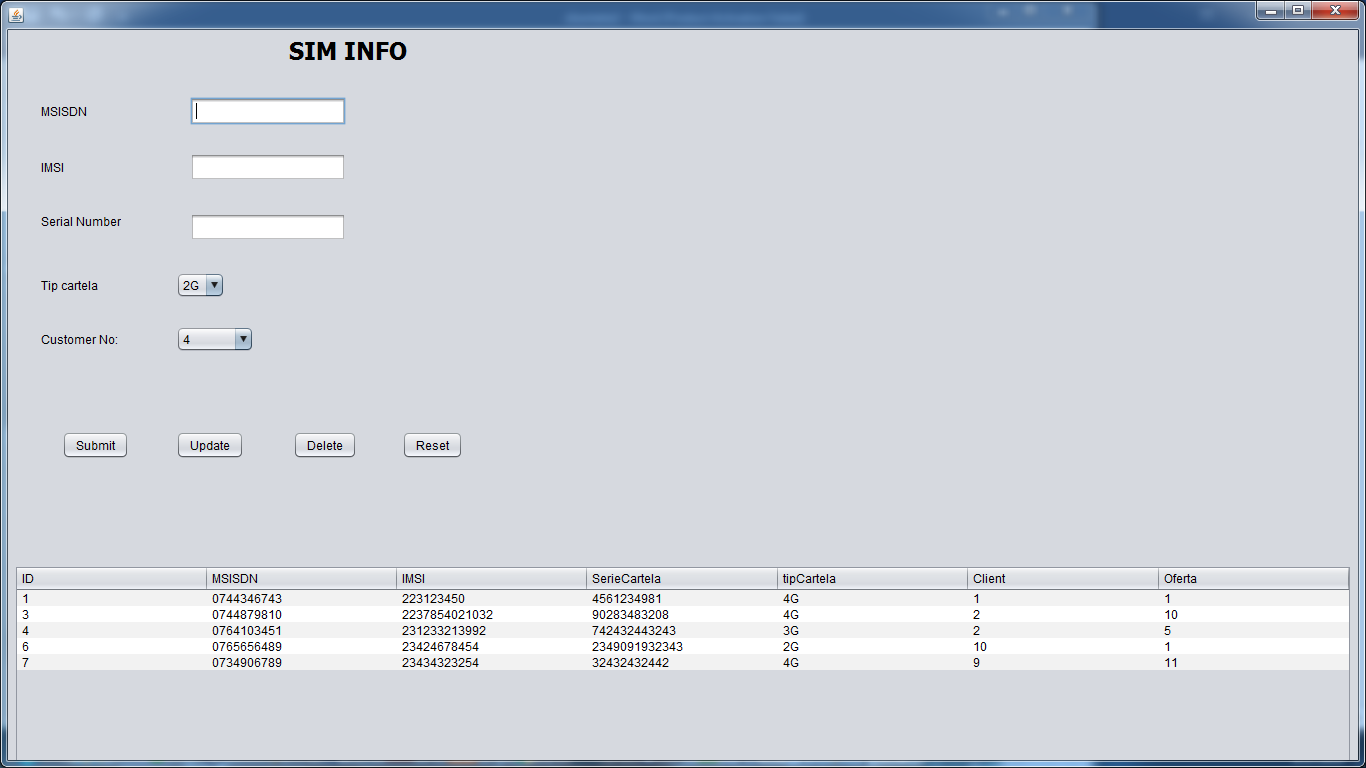
Pentru funcţionalitea **Detalii clienti** avem posibilitatea de a vedea clienţii existenţi in baza de date, iar ca operaţiuni putem insera, modicafica sau să-i ştergem.

Figura 8 – Clienti



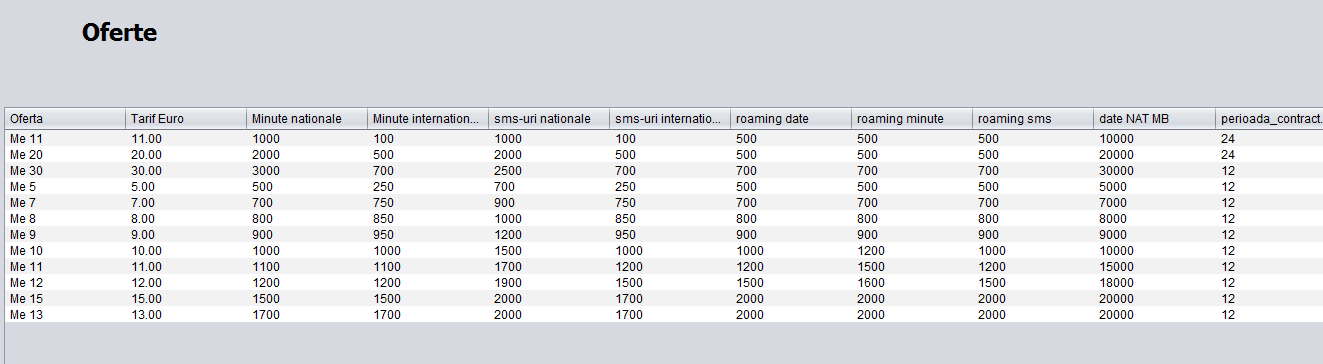
Fereastra cu **numere de telefon** ne permite să vizualizăm numerele din baza de date pe care le avem si daca au fost sau nu asignate unui client. De asemenea putem să adăugam noi numere de telefon daca avem cartele SIM in magazin.

Figura 9 – Informatii Cartele telefon



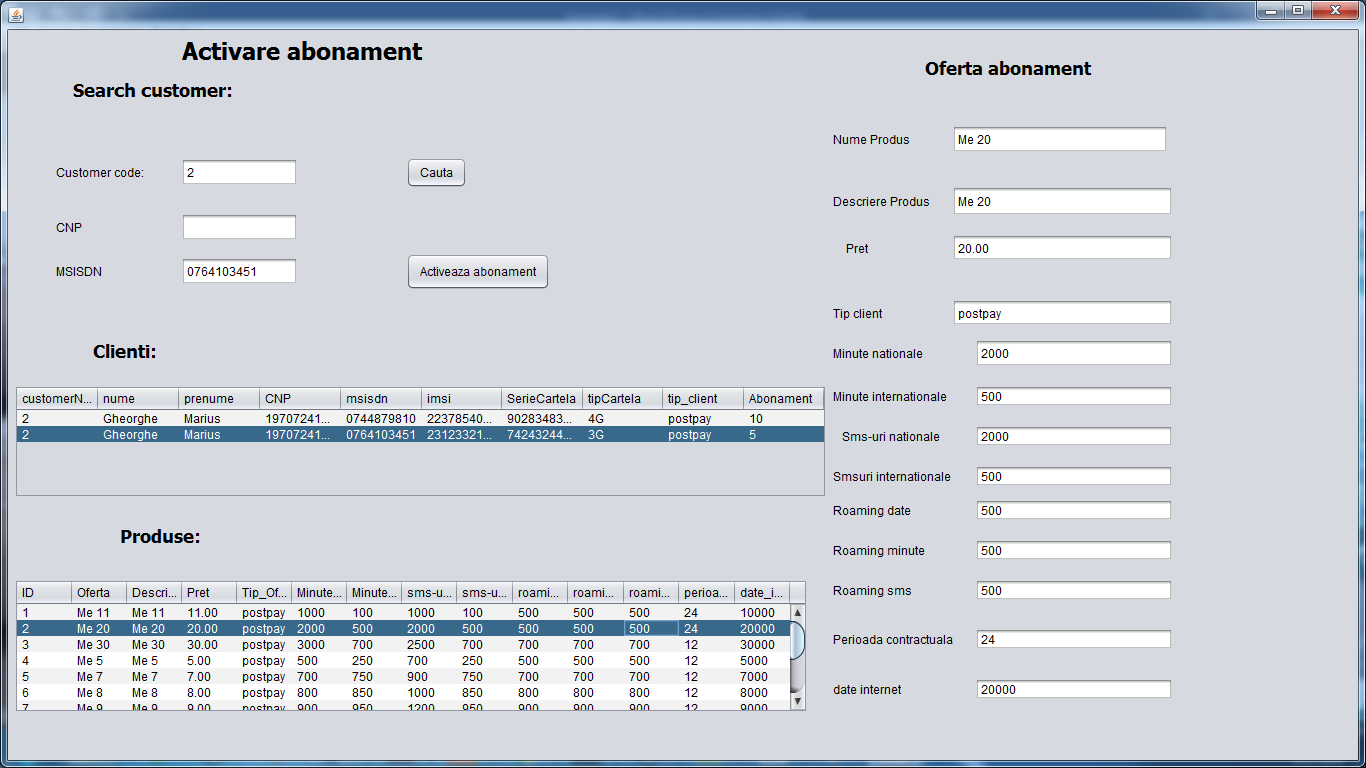
În opţiunea de **Oferte** avem informaţii despre ce putem noi să oferim clienţilor, abonamentele pe care acestia le pot activa impreuna cu ce ofera acestea minute, sms-uri, date de internet:

Figura 10 - Oferte



Fereastra cu Activare abonaent se realizeaza activarea abonamentului. De aici putem realiza o cautare dupa codul de abonat pentru a gasii clientul.

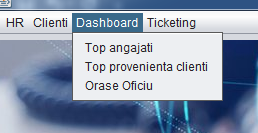
Figura 11 – Activare abonamente



Modulul de Dashboad prezinta urmatoarele:

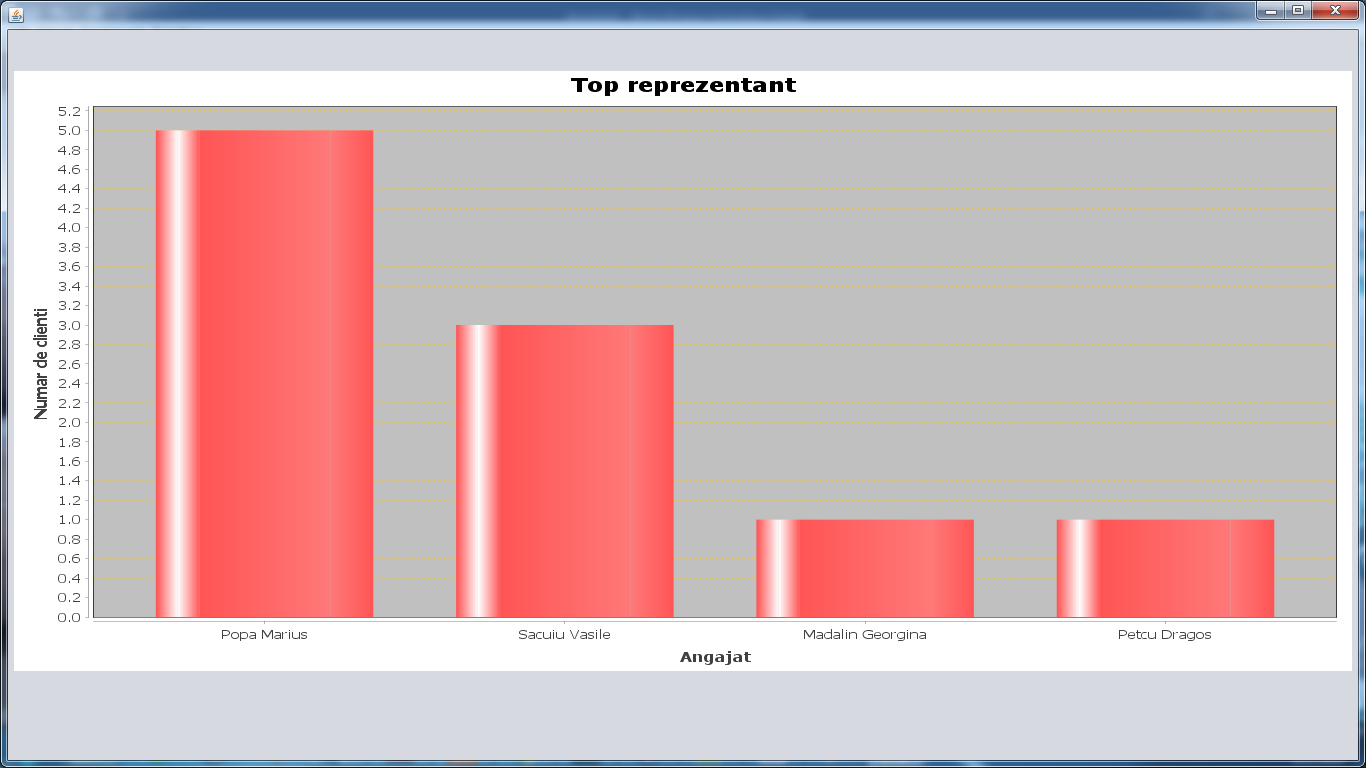
* Top angajati
* Top provenienta clienti
* Orase Oficiu

Figura 12 – Menu Bar Grafice



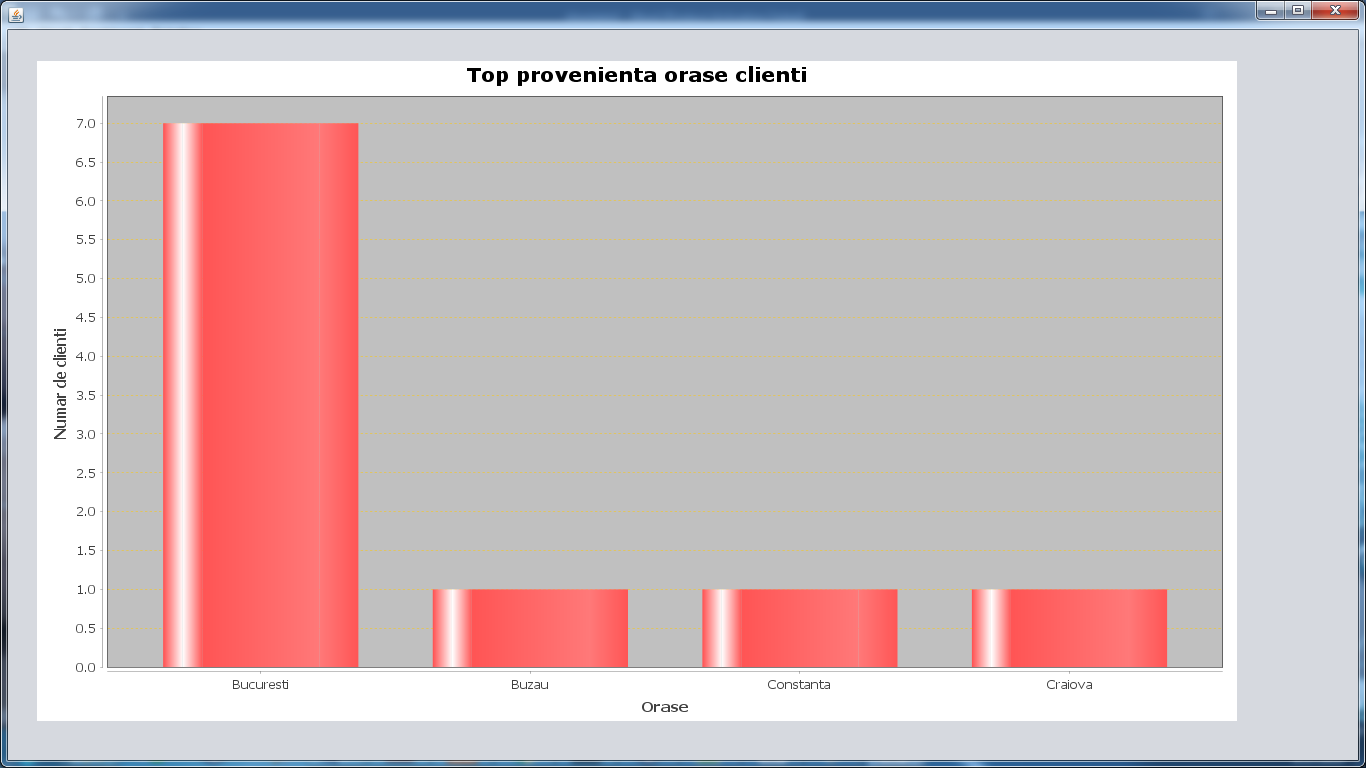
La partea de Top angajati avem o ierarhizare a acestora in functie de numarul de clienti pe care il are fiecare.

Figura 13 – Grafic Top reprezentant



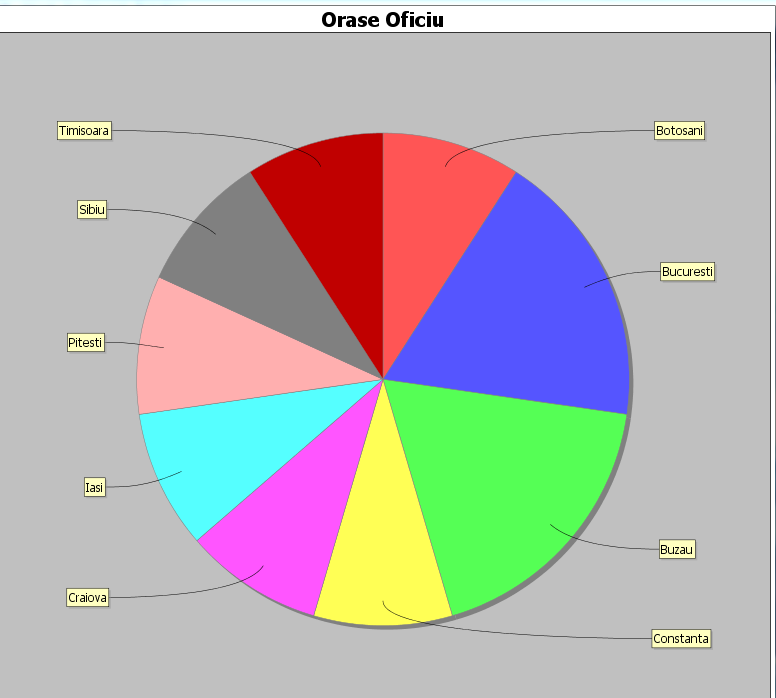
La tabul de Top provenienta orase putem observa de unde sunt cei mai multi clienti.

Figura 14 – Grafic Top provenienta orase clienti.



Iar la orase oficiu avem un grafic unde putem observa in ce oras sunt prezente cele mai multe magazine.

Figura 15 – Grafic Distribuiţie Oficii - orase



**3.4 Implementare**

**3.4.1 API-ul JDBC**

Pentru extragerea datelor de pe baza de date s-a folosit APIul JDBC.

JDBC reprezintă Java Database Connectivity, care este un standard Java API pentru conectivitate independentă de bazele de date între limbajul de programare Java și o gamă largă de baze de date (|TPoint2012|).

Biblioteca JDBC include API-uri pentru fiecare dintre sarcinile menționate mai jos, care sunt în mod obișnuit asociate cu utilizarea bazei de date.

Efectuarea unei conexiuni la o bază de date.

Crearea de instrucțiuni SQL sau MySQL.

Executarea interogărilor SQL sau MySQL în baza de date.

Vizualizarea și modificarea înregistrărilor rezultate.

Exemplu de folosire se poate observa in anexa nr.1.

**3.4.2 Singleton**

Pentru conexiuna cu baza de date s-a folosit singleton pattern. Acest model implică o singură clasă care este responsabilă pentru crearea unui obiect, asigurându-se în același timp crearea unui singur obiect unic. Această clasă oferă o modalitate de a accesa singurul său obiect care poate fi accesat direct fără a fi nevoie să se instanțieze obiectul clasei.

Implementarea se poate observa in anexe la anexa nr.2.

**Concluzii**

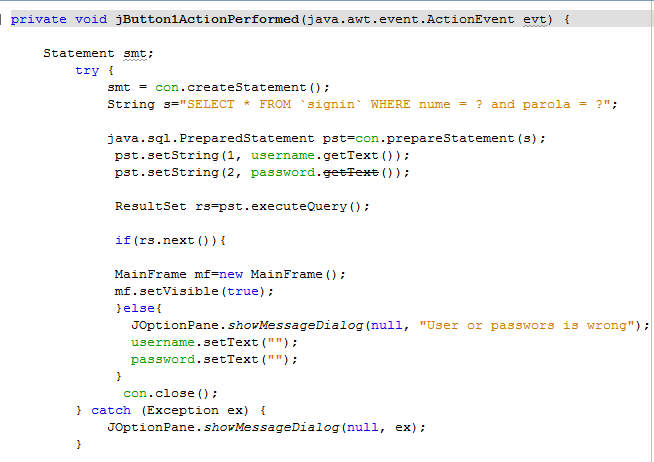
Lucrul cu clienții poate fi o sarcină dificilă. Păstrarea contactelor, diagnosticarea corectă a problemelor și asigurarea satisfacției, precum și interacțiunile pozitive se pot simți ca ponderea lumii pe umerii echipei de relații cu clienții. Software-ul de gestionare a relațiilor cu clienții (CRM) poate purta o mulțime de avantaje pentru procesele de afaceri și vânzări.

In acest sens pentru simplificarea muncii angajatilor se poate face si implementarea pe dispozitivele mobile. Toată lumea are un smartphone în aceste zile, deci este logic să se foloseasca tehnologia uimitoare din interiorul lor pentru a ajuta afacerea. Gestionarea relației cu clienții pe un dispozitiv de la distanță utilizând programe și aplicații CRM mobile. Personalul poate vizualiza date și primi alerte valoroase despre campaniile pe un dispozitiv mobil.

Opțiuni de personalizare. Personalizarea este una dintre caracteristicile principale ale software-ului CRM, dar are implicații foarte mari. Cu cât sunt mai personalizate, cu atât mai flexibilă este CRM. Este crucial să se cunoasca nevoile atunci când vine vorba de personalizare înainte de a face o achiziție. Va crește software-ul cu organizația ? Are capacitatea de a se adapta și de a se integra cu software-ul existent utilizat de organizația ? Poate să includă instrumente de productivitate desktop? Se poate schimba atunci când este nevoie? Toate aceste lucruri va duce la o flexibilitate mai mare a softwarelui.

**ANEXE**

Anexa numarul 1 – Implementare a API-ului JDBC exemplu:



Anexa numarul 2– Conectare la baza de date



**BIBLIOGRAFIE**

1. |ECK2006| - BRUCE ECKEL - Thinking in Java, Prentice Hall, 2006.

2. |SCHI2014| - HERBERT SCHILDT – Java A Beginner’s Guide, Oracle Press, 2014.

3. |WILL2005| - PAUL WILTON si JOHN W. COLBY - Beginning SQL, Wiley Publishing, 2005.

4. |Educba2019| - Educba, Introduction to Applications of Java, [Online] Disponibil la adresa: https://www.educba.com/applications-of-java/,2019, accesat la data de 03.04.2019.

5. |Educba2018|-Educba, Java vs C#, [Online] Disponibil la adresa:

https://www.educba.com/java-vs-c-sharp,2018, accesat la data de 03.04.2019.

6. |TPoint2012|- TutorialsPoint, JDBC Tutorial, [Online] Disponibil la adresa: https://www.tutorialspoint.com/jdbc/jdbc-introduction.htm, 2012, accesat la data de 10.04.2019.