Tehnici de programare fundamentale – Tema 3

Simulator al managementului unui magazin online

1. Obiectivul temei

Principalul obiectiv al acestei teme este de a proiecta și implementa un program care să simuleze gestiunea clienților, a produselor și a comenzilor dintr-un magazin online, mai exact felul în care se pot adăuga, edita, șterge și vizualiza atât date despre clienți, cât și despre produse, dar se pot introduce și comenzi noi, prin alegerea clienților si a produselor pe care se presupune că aceștia le comandă. Programul va avea interfață grafică dedicată, concepută într-o manieră intuitivă și atractivă pentru utilizator, atât pentru introducerea unor noi date în tabelele bazei de date cu care se face legătura, cât și pentru editarea, vizualizarea sau ștergerea de date din aceste tabele. De asemenea, se va genera un fișier text cu rol de confirmare a comenzii și de bon fiscal, pe care să apară datele de contact al clientului, numărul comenzii, produsele comandate (cu cantitatea corespunzătoare), prețul pe bucată și per total al fiecărui produs, dar și totalul de plată al comenzii.

Proiectul își propune să îi permită utilizatorului să:

- introducă datele unui nou client ce se dorește a fi introdus în baza de date
- introducă datele unui nou produs ce se dorește a fi introdus în baza de date
- introducă noile valori cu care să editeze câmpurile unor clienți, ori ale unor produse
- șteargă intrări din tabelul cu clienți sau cu produse
- vizualizeze într-un tabel datele tuturor clienților sau produselor existente în baza de date
- plaseze noi cereri de comenzi în numele unor clienți dintre cei existenți

Obiectivele secundare pe care dezvoltatorul și le propune se regăsesc în tabelul de mai jos, alături de secțiunile din prezenta lucrare în care vor fi adresate:

	Obiectiv secundar	Scurtă descriere	Secțiunea dedicată
1.	Analiza problemei	- stabilirea cerințelor la care programul trebuie să răspundă	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
2.	Modelare	- imaginarea modelului conceptual pentru programul ce urmează a fi dezvoltat	2. Analiza problemei, modelare, scenarii, cazuri de utilizare

3.	Scenarii posibile	- analiza exhaustivă a situațiilor în care programul poate fi pus de către utilizator (input valid, șirul nul ca nume de fișier, numere negative ca valori de preț ori cantitate, etc.)	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
4.	Cazuri de utilizare	- determinarea cazurilor pentru care utilizatorul dorește să folosească programul creat	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
5.	Proiectare	- definirea structurilor de date necesare, a claselor și interfețelor ce urmează a fi implementate, a tabelelor bazei de date ce va fi folosită în spatele programului	3. Proiectare
6.	Implementare	- descrierea claselor și a metodelor definite, explicarea funcționării și a utilității acestora; descrierea interfețelor grafice; descriere JavaDoc	4. Implementare
7.	Testare	- generarea fișierelor text cu rol de confirmare a comenzii și de bon fiscal	5. Rezultate

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

2.1. Analiza problemei

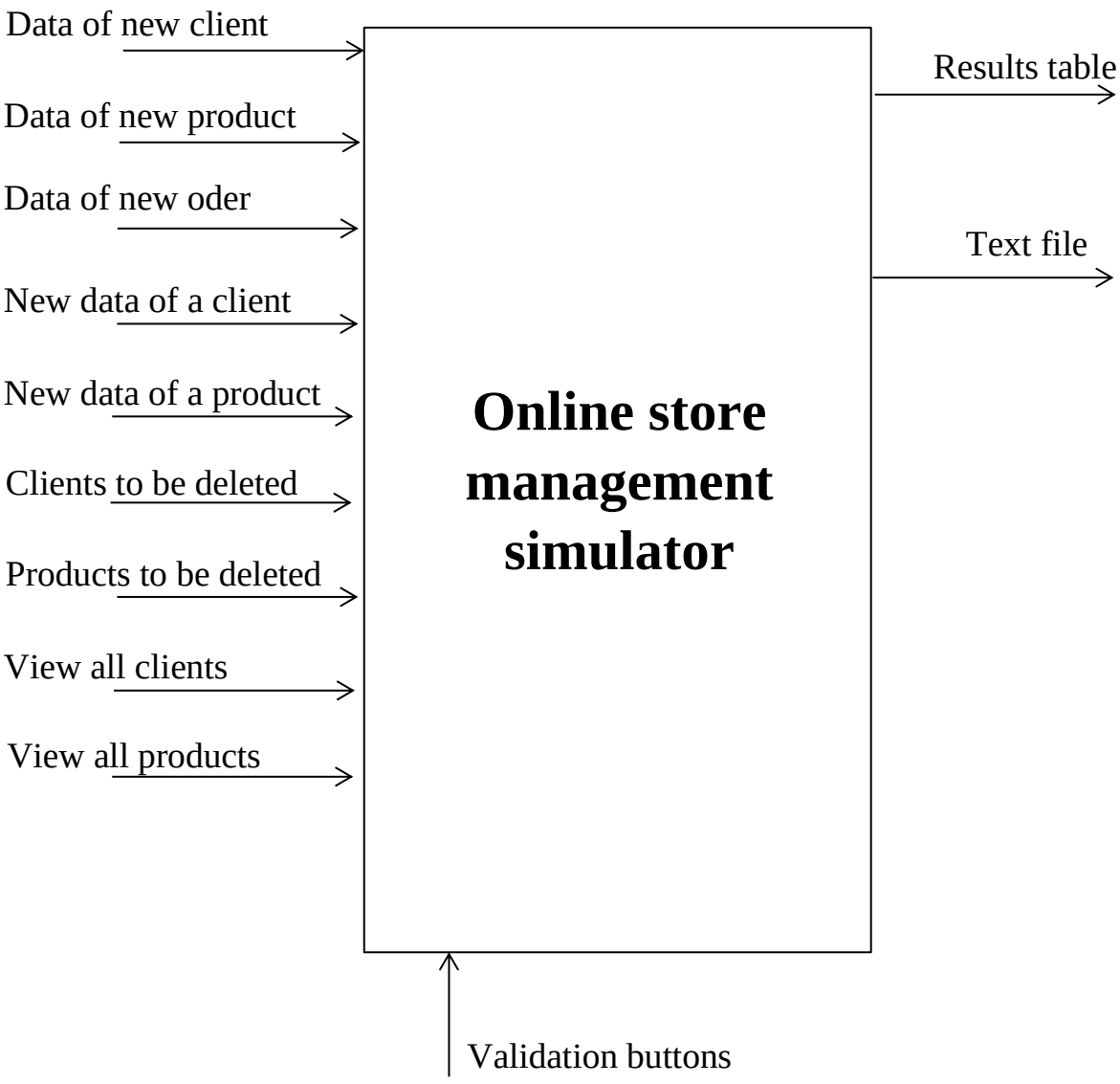
S-au stabilit cerințele funcționale și non-funcționale la care programul, simulatorul managementului unui magazin online, trebuie să răspundă:

a) Cerințe funcționale:
   a) simulatorul trebuie să permită utilizatorului să introducă datele unor noi clienți și produse care să fie introduse în baza de date
   b) simulatorul trebuie să permită utilizatorului să plaseze o comandă în numele unui client, știind pentru ce client se face comanda și ce produse dorește acesta să comande
   c) simulatorul trebuie să permită utilizatorului să modifice datele unui client/produs deja existent în baza de date
   d) simulatorul trebuie să permită utilizatorului să șteargă intrări de date deja existente în tabelele de clienți/ produse
   e) simulatorul trebuie să permită vizualizarea într-un tabel a tuturor datelor din tabelul de clienți sau de produse din baza de date
   f) simulatorul trebuie să informeze utilizatorul dacă inputul său nu este valid
   g) simulatorul trebuie să furnizeze un fișier cu rol de confirmare a comenzii și de factură/ bon fiscal pentru fiecare comandă plasată

b) Cerințe non-funcționale:
   - simulatorul trebuie să fie ușor de folosit și intuitiv
   - simulatorul trebuie să aibă o interfață atractivă, atât pentru introducerea datelor de intrare ale unui client sau produs, dar mai ales pentru editarea, ștergerea sau vizualizarea unor intrări de date deja existente, precum și pentru introducerea comenzilor

2.2. Modelare

S-a definit modelul de lucru al simulatorului de cozi de clienți, sub formă de schemă bloc:



2.3. Scenarii posibile

Scenariile în care se poate găsi simulatorul managementului magazinului online pot fi diferențiate în două mari categorii: scenarii uzuale de utilizare și scenarii limită. În cele ce urmează se va prezenta felul în care programul va răspunde la fiecare dintre aceste tipologii de scenarii:

I. Scenariu uzual:
   - utilizatorul alege tabelul din baza de date la care dorește să aibă acces în continuare (*Customer*, *Product* sau *Request*)
   - utilizatorul alege din opțiunile puse la dispoziție operația pe care dorește să o realizeze asupra datelor din tabelul selectat anterior
   - dacă este vorba de tabelul *Customer* sau *Product*, o operație realizată cu succes va fi marcată de închiderea ferestrei operației ce tocmai a fost realizată
   - în cazul introducerii în baza de date a unor noi comenzi, se va genera și fișierul de factură al comenzilor plasate în respectivul ciclu de comandare
   - programul permite executarea unui nou ciclu de utilizare, fără a părăsi și a reporni aplicația

II. Scenarii limită:
   a. Șir de caractere în loc de numere:
      - utilizatorul introduce un șir de caractere pe post de valoare întreagă pentru cantitatea unui produs
      - utilizatorul acționează butonul care ar introduce datele in baza de date
      - programul îl notifică despre faptul că datele pot fi doar numere întregi, și îi dă posibilitatea de a corecta eroarea => utilizatorul are posibilitatea să intre într-un scenariu uzual
   b. Email sau telefon invalid:
      - utilizatorul introduce un email sau un număr de telefon invalid
      - utilizatorul acționează butonul care ar introduce datele in baza de date
      - programul îl notifică despre faptul că email-ul sau telefonul este invalid => utilizatorul are posibilitatea să intre într-un scenariu uzual
   c. Număr negativ:
      - utilizatorul introduce un număr negativ în unul dintre câmpurile pentru cantitatea sau prețul unui nou produs
      - utilizatorul acționează butonul ferestrei de adăugare a unui nou produs
      - programul îl notifică despre faptul că datele nu pot fi numere negative; utilizatorul are posibilitatea să intre într-un scenariu uzual

d. Șirul vid
   - utilizatorul lasă necompletat unul dintre câmpurile pentru un client sau pentru un produs
   - utilizatorul acționează butonul ferestrei curente
   - programul îl notifică despre faptul că field-urile rămase necompletate trebuie să fie scrise, și îi dă posibilitatea să facă această adăugare

e. Număr de caractere depășit
   - utilizatorul introduce un șir de caractere a cărui lungime depășește lungimea câmpului corespunzător din baza de date MySQL
   - programul îl notifică despre faptul că șirul nu poate avea mai mult de x caractere, unde x este numărul de caractere declarat în MySQL

- Este important de menționat că în cazul editării datelor unor clienți/produse deja existente, notificarea asupra inconsistenței datelor de intrare va fi semnalată imediat după completarea câmpului, și nu doar la apăsarea butonului. Astfel utilizatorul este mai rapid notificat asupra erorii

2.4. Cazuri de utilizare

Descrierea utilizării:

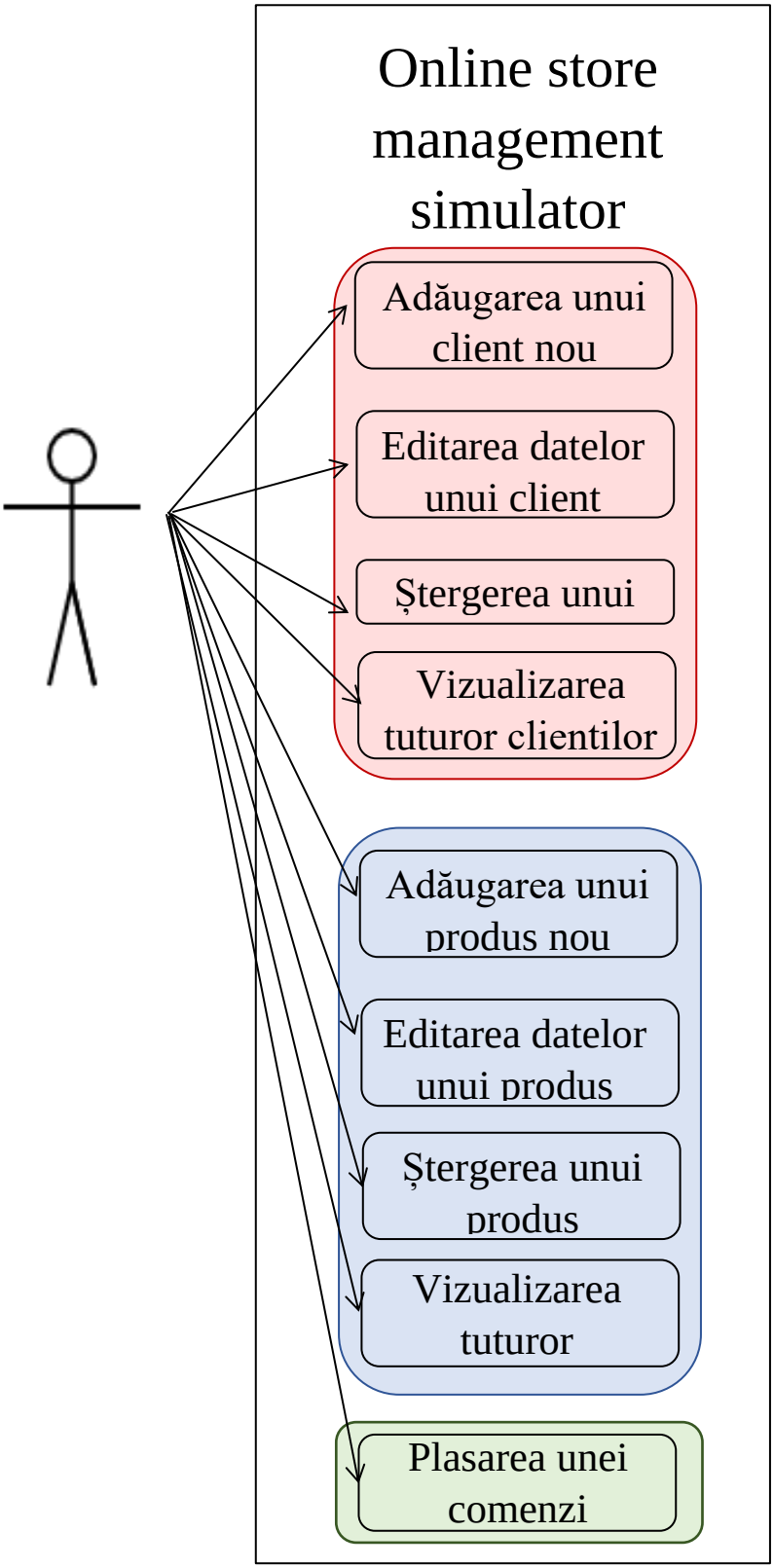
Folosirea simulatorului magazinului online este facilă și intuitivă. Datele de intrare cerute nu conțin niciun fel de termeni de specialitate care să deruteze utilizatorul. Mai mult decât atât, comunicarea cu user-ul în caz de eroare la introducerea datelor este una extrem de complexă și explicită.

De exemplu, pentru adăugarea unui nou client sau produs în baza de date, utilizatorul introduce datele de intrare în câmpurile specifice folosind tastatura propriului computer. La apăsarea butonului *Add*, dacă există date incorecte se va afișa pe ecran o casetă de dialog ce va semnala eroarea. Dacă toate datele sunt valide, se închide fereastra de adăugare și rezultatul se poate observa prin deschiderea ferestrei de vizualizare a tuturor clienților/ produselor. Obiectul ce tocmai a fost adăugat se va găsi la finalul listei.

Pentru ștergerea unui nou client se va afișa în fereastra corespunzătoare un tabel cu toți clienții. Se vor selecta rândurile care se dorește a fi șterse din baza de date. Se pot selecta oricâte rânduri. La apăsarea butonului *Delete* înregistrările selectate se vor șterge din sistem. Idem pentru ștergerea unui produs.

Pentru editarea datelor unui client se va deschide în fereastra corespunzătoare un tabel în care apar toți clienții. Se modifică datele dorite. Se selectează rândurile ce au fost modificate și se apasă butonul *Edit*. Același mecanism se aplică și pentru ștergerea unui produs.

Pentru vizualizarea tuturor clienților/ produselor trebuie doar să fie apăsat butonul ce indică acest lucru din fereastra principală a operațiilor pe tabelul *Customer*/*Product* din baza de date. Se va deschide o nouă fereastră cu un tabel conținând toate înregistrările dorite.



Pentru plasarea unei comenzi se va apăsa butonul corespunzător din fereastra inițială a programului (cea de bun-venit). Se va deschide o fereastră ce conține două tabele: unul cu datele clienților, celălalt cu datele produselor. Se vor selecta din tabelul din stânga clienții care doresc să comande aceleași produse și în exact aceleași cantități. Se vor selecta din tabelul din dreapta produsele ce se comandă. Se va completa în coloana pentru cantitate a fiecărui produs care este cantitatea în care se comandă respectivul produs. La apăsarea butonului *Order* se înregistrează comenzile în baza de date și se generează confirmarea comenzii/bonul pentru fiecare client în parte.

Rațiunile pentru care utilizatorul poate dori să folosească acest simulator al managementului unui magazin online:

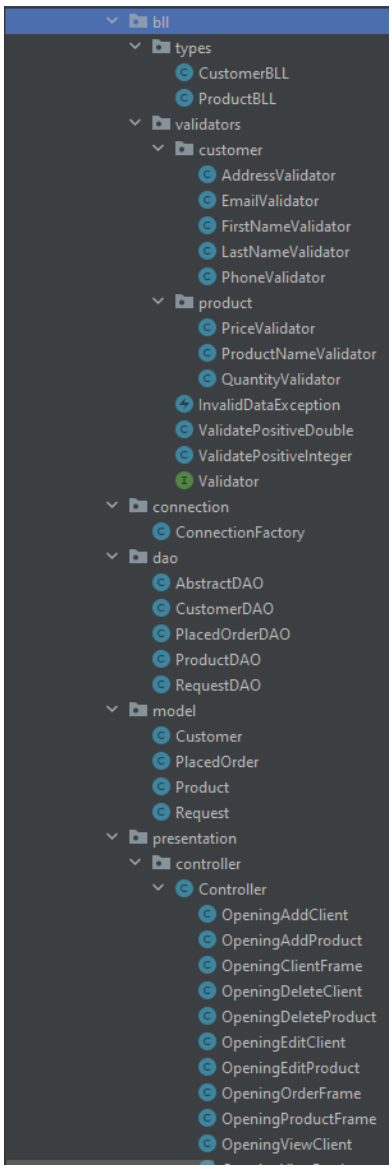
3. Proiectare

Pentru a respecta conceptele programării orientate pe obiect, fiecare clasă implementată este cât mai specifică, având cât mai puține funcționalități (ex: clasă separată pentru validarea fiecărei variabile instanță).

3.1. Pachete

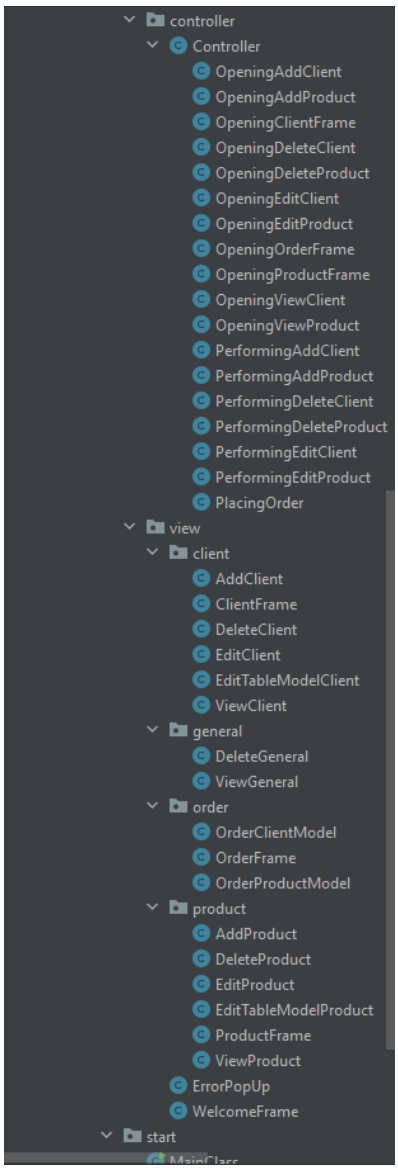
S-a folosit Layered Architecture pentru structurarea proiectului, ceea ce înseamnă că există câte un pachet care să răspundă următoarelor necesități: validarea datelor (Busssiness Layer - bll), accesul la baza de date (Data Access Layer - dao), modelarea în Java a informațiilor din MySQL (model) și interacțiunea cu utilizatorul (presentation).

Proiectul este structurat pe unități funcționale cât mai specifice, clasele care se ocupă de o anumită ramură a programului fiind grupate în pachete și sub-pachete, după cum se poate vedea alături în lista claselor așa cum apare ea în IDE-ul IntelliJ, dar și mai jos în diagrama de pachete:



Structura proiectului.

Organizarea pe pachete



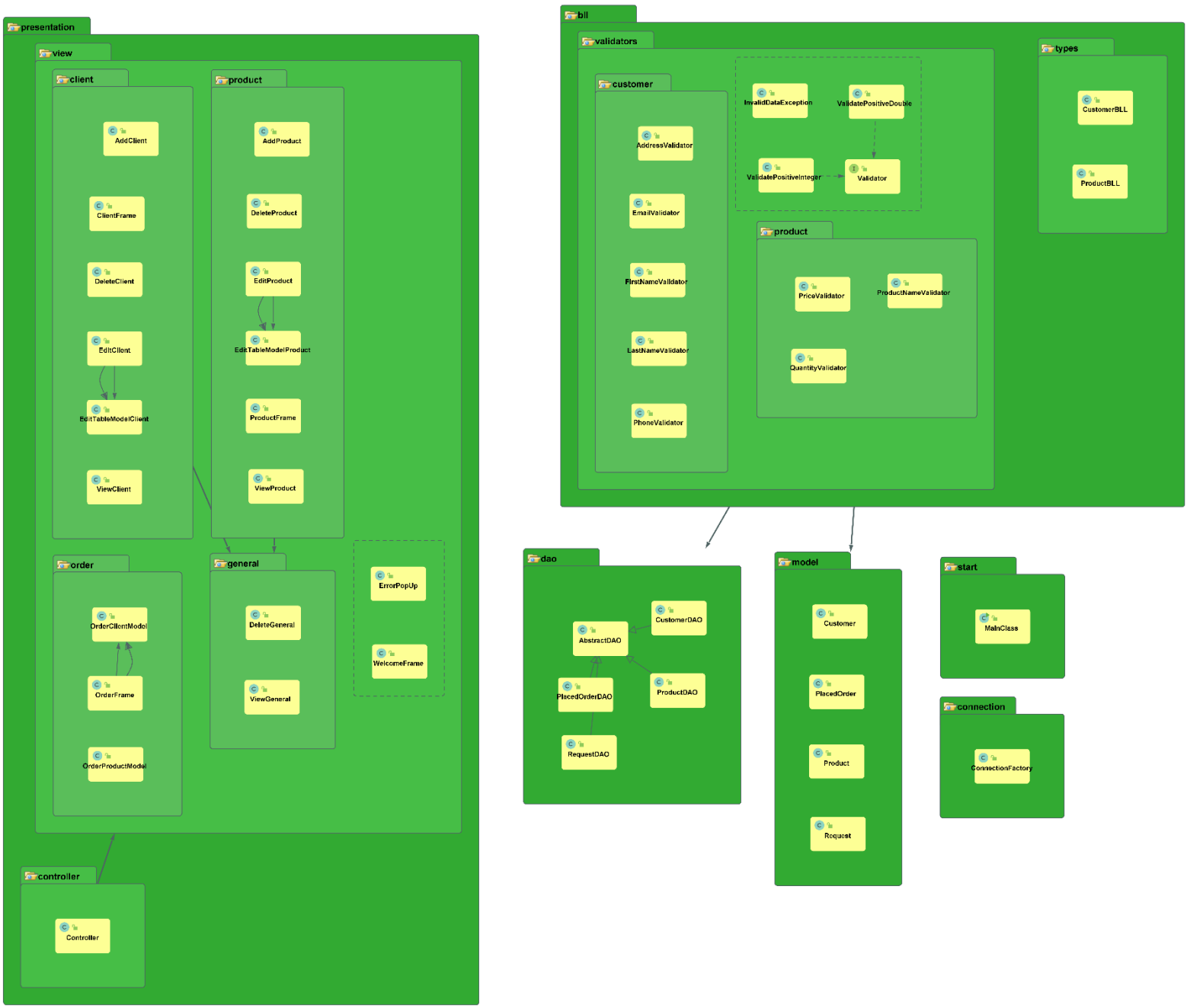


Diagrama de pachete

3.2. Clase

Diagrama de clase a întregului proiect arată relațiile dintre cele 45 de clase și interfețe definite, și este atașată pe următoarea pagină. Clasele vor fi descrise detaliat în secțiunea 4.Implementare.

3.3.Interfețe definite

Singura interfață definită în proiect este *Validator<T>,* folosită pentru validarea datelor ce se dorește a fi introduse în baza de date.

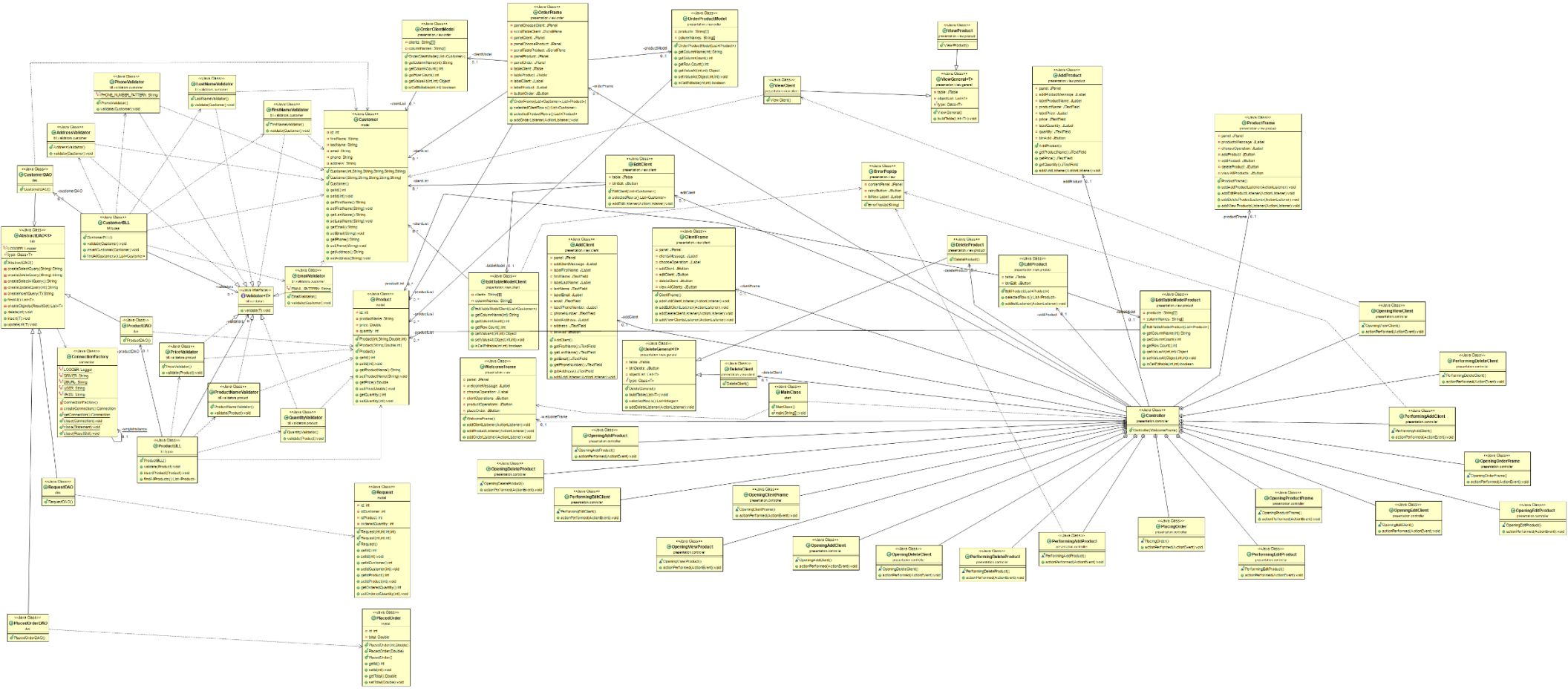
Fiecare clasă oferă o implementare particulară a metodei *validate* din interfață, în concordanță cu operația denumită de clasă (ex: EmailValidator, PhoneValidator, AddressValidator etc.).

3.4.Structuri de date folosite

Ca structuri de date s-au folosit *List* și *ArrayList* pentru a defini liste de clienți, produse, obiecte, String-uri etc.

Clasele *Customer, Product, Reques* și *PlacedOrder* pot fi considerate și ele structuri de date, întrucât acționează ca un tot-unitar în cadrul programului de față.

Diagrama de clase



3.5. Algoritmi folosiți

Cu titulatura de algoritmi se pot considera metodele de construire a statement-urilor de interogare a bazei de date, precum și cele de executare a statment-ului, cele care execută efectiv interogarea asupra bazei de date prin conexiunea cu aceasta, conexiune definită în prealabil.

Fiecare statement se construiește folosind un StringBuilder. În locul valorii câmpurilor din tabel se inserează semne de întrebare, câte unul pentru fiecare valoare. Rezultatul returnat de o astfel de metoda este un String de genul “DELETE * FROM Customer WHERE id=?”.

Metoda de executare a statement-ului înlocuiește semnul de întrebare cu valoarea efectivă a parametrului, în cazul prezentat mai sus, cu valoarea id-ului clientului căutat.

4. Implementare

S-a folosit conceptul încapsulării din paradigma programării orientate pe obiect, atributele claselor fiind declarate *private* și accesibile claselor exterioare prin metode *getter* și *setter*.

Explicarea detaliată a metodelor din fiecare dintre cele 45 de clase se poate regăsi în documentația JavaDoc (atașată proiectului), ca urmare se vor aminti în continuare doar clasele fiecărui pachet și funcționalitățile acestora

4.1. Clasele pachetului *bll.types*

4.1.1. *CustomerBLL:*

Clasa pentru validarea tuturor campurilor clasei Customer si pentru apelarea metodelor de interactiune cu baza de date prin intermediul CustomerDAO.

4.1.2. *ProductBLL*

Clasa pentru validarea tuturor variabilelor instanta ale unui obiect de tip Product, si pentru realizarea unor operatii in baza de date prin intermediu ProductDAO.

4.2. Clasele pachetului *bll.validators*

Acesta este pachetul în care s-a definit și interfața Validator<T>, explicată anterior, în secțiunea *3.3. Interfețe definite*

4.2.1. *ValidatePositiveDouble*

Clasa pentru validarea introducerii unui număr pozitiv de tip Double.

4.2.2. *ValidatePositiveInteger*

Clasa pentru validarea introducerii unui număr de tip Integer.

4.2.3. *InvalidDataException*

Clasa de excepție folosita la semnalarea unei inconsistente in datele ce se doresc a fi introduse in baza de date.

4.3. Clasele pachetului *bll.validators.customer*

4.3.1. *AddressValidator*

Clasa pentru validarea câmpului Address al unui client.

4.3.2. *EmailValidator*

Clasa pentru validarea câmpului Email al unui client.

4.3.3. *FirstNameValidator*

Clasa pentru validarea câmpului FistName al unui client.

4.3.4. *LastNameValidator*

Clasa pentru validarea câmpului LastName al unui client.

4.3.5. *PhoneValidator*

Clasa pentru validarea câmpului Phone al unui client.

4.4. Clasele pachetului *bll.validators.product*

4.4.1. *PriceValidator*

Clasa pentru validarea campului Price al unui produs.

4.4.2. *ProductNameValidator*

Clasa pentru validarea campului ProductName al unui produs.

4.4.3. *QuantityValidator*

Clasa pentru validarea campului Quantity al unui produs.

4.5. Clasele pachetului *connection*

4.5.1. *ConnectionFactory*

Clasa pentru realizarea conexiunii dintre cod si baza de date.

4.6. Clasele pachetului *dao*

4.6.1. *AbstractDAO<T>*

Clasa pentru construirea interogărilor asupra bazei de date

4.6.2. *CustomerDAO*

Clasa pentru realizarea interacțiunii cu baza de date asupra tabelului Customer.

4.6.6. *PlacedOrderDAO*

Clasa pentru realizarea interacțiunii cu baza de date asupra tabelului PlacedOrder.

4.6.7. *ProductDAO*

Clasa pentru realizarea interacțiunii cu baza de date asupra tabelului Product

4.6.8. *RequestDAO*

Clasa pentru realizarea interacțiunii cu baza de date asupra tabelului Request

4.7. Clasele pachetului *model*

4.7.1. *Customer*

Clasa pentru "traducerea" in Java din baza de date a tuplelor din tabelul Customer

4.7.2. *PlacedOrder*

Clasa pentru "traducerea" in Java din baza de date a tuplelor din tabelul PlacedOrder

4.7.3. *Product*

Clasa pentru "traducerea" in Java din baza de date a tuplelor din tabelul Product

4.7.4. *Request*

Clasa pentru "traducerea" in Java din baza de date a tuplelor din tabelul Request

4.8. Clasele pachetului *presentation.controller*

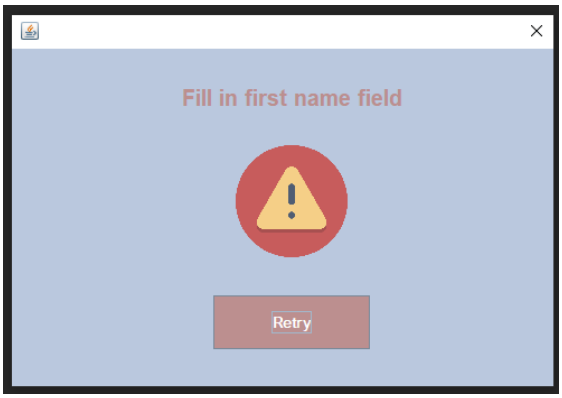
4.8.1. *Controller*

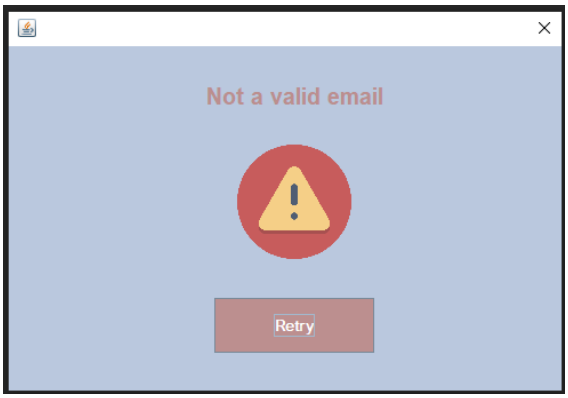
Clasa pentru realizarea conexiunii dintre Model si View

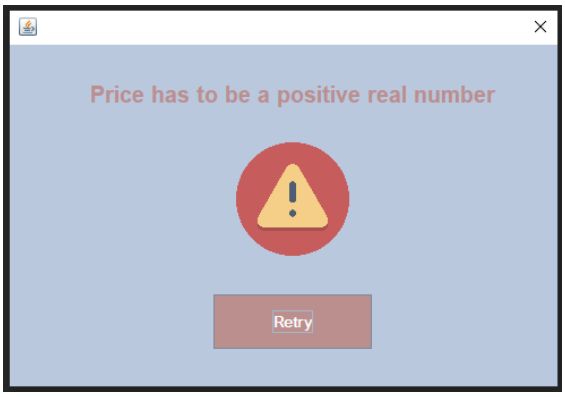
4.9. Clasele pachetului *presentation.view*

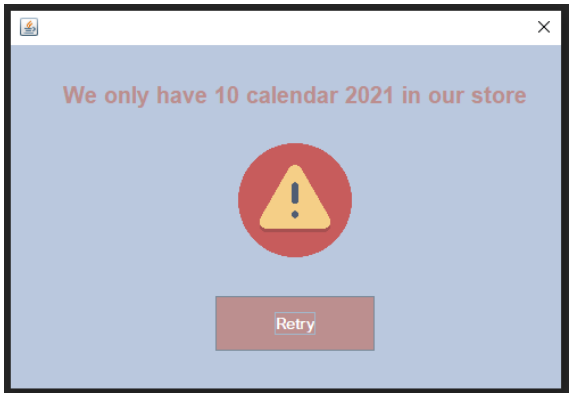
4.9.1. *ErrorPopUp*

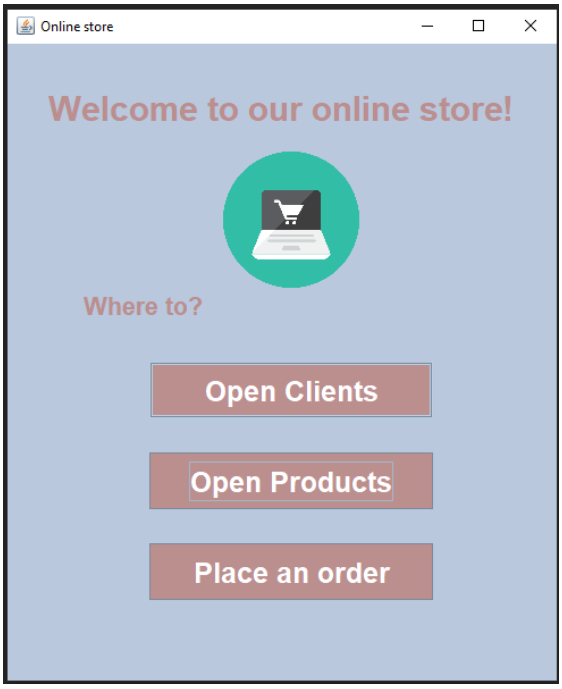
Clasa pentru construirea casetei de dialog ce avertizează user-ul ca a introdus un input invalid











4.9.2. *WelcomeFrame*

Clasa pentru crearea primei ferestre vizualizate de user la deschiderea aplicatiei, fereastra din care se poate alege daca urmatoarea operatie va fi asupra tabelului de clienti, produse sau comenzi.

4.10. Clasele pachetului *presentation.view.client*

4.10.1. *AddClient*

Clasa pentru crearea interfetei grafice de adaugare a unui nou client in baza de date

4.10.2. *ClientFrame*

Clasa pentru crearea interfetei grafice de selectare a unei operatii pe tabelul Customer din baza de date

4.10.3. *DeleteClient*

Clasa pentru crearea interfetei grafice de stergere client din baza de date Particulatizeaza fereastra generala de stergere din baza de date

4.10.4. *EditClient*

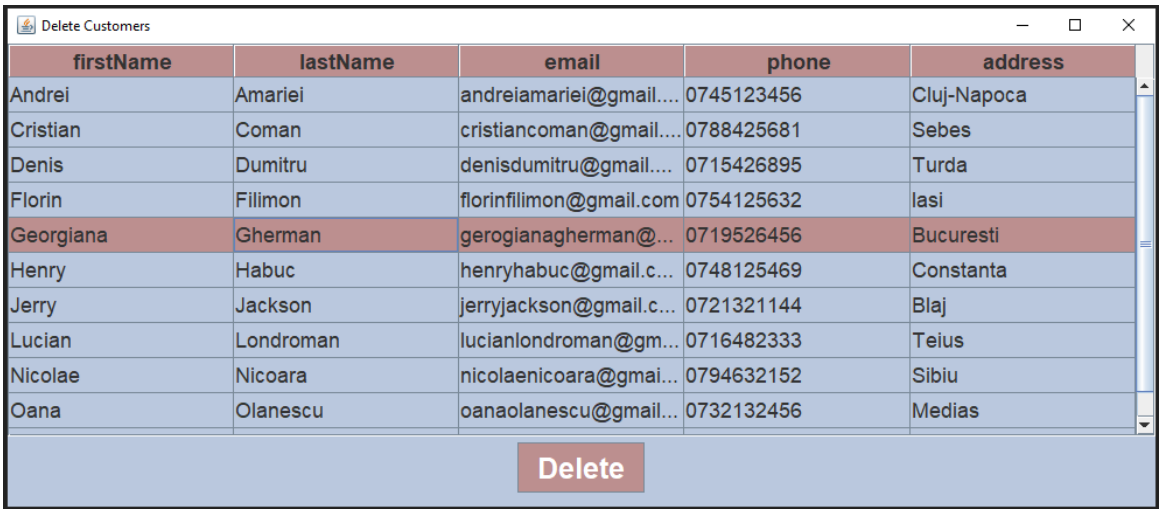
Clasa pentru crearea interfetei grafice de editare client in baza de date

4.10.5. *EditTableModelClient*

Clasa pentru crearea modelului abstract de tabel al JTable-ului din interfata grafica de editare client in baza de date

4.10.6. *ViewClient*

Clasa pentru crearea interfetei grafice de vizualizare a tuturor clientilor din baza de date Particulatizeaza fereastra generala de vizualizare a tuturor intrarilor unui tabel din baza de date


Delete Customers
firstName lastName email phone address
Andrei Amariei andreiamariei@gmail.... 0745123456 Cluj-Napoca
Cristian Coman cristiancoman@gmail.... 0788425681 Sebes
Denis Dumitru denisdumitru@gmail.... 0715426895 Turda
Florin Filimon florinfilimon@gmail.com 0754125632 Iasi
Georgiana Gherman gerogianagherman@... 0719526456 Bucuresti
Henry Habuc henryhabuc@gmail.c... 0748125469 Constanta
Jerry Jackson jerryjackson@gmail.c... 0721321144 Blaj
Lucian Londroman lucianlondroman@gm... 0716482333 Teius
Nicolae Nicoara nicolaenicoara@gmai... 0794632152 Sibiu
Oana Olanescu oanaolanescu@gmail... 0732132456 Medias
Delete


Edit clients
firstName lastName email phone address
Andrei Amariei andreiamariei@gm... 0745123456 Cluj-Napoca
Cristian Coman cristiancoman@gm... 0788425681 Sebes
Denis Dumitru denisdumitru@gma... 0715426895 Turda
Florin Filimon florinfilimon@gmail.... 0754125632 Iasi
Georgiana Gherman gerogianagherman... 0719526456 Bucuresti
Henry Habuc henryhabuc@gmail... 0748125469 Constanta
Jerry Jackson jerryjackson@gmai... 0721321144 Blaj
Lucian Londroman lucianlondroman@... 0716482333 Teius
Nicolae Nicoara nicolaenicoara@g... 0794632152 Sibiu
Oana Olanescu oanaolanescu@gm... 0732132456 Medias
Rodica Rus rodicarus@gmail.c... 0745125456 Lancram
Edit


All Customers
firstName lastName email phone address
Andrei Amariei andreiamariei@gm... 0745123456 Cluj-Napoca
Cristian Coman cristiancoman@gm... 0788425681 Sebes
Denis Dumitru denisdumitru@gma... 0715426895 Turda
Florin Filimon florinfilimon@gmail.... 0754125632 Iasi
Georgiana Gherman gerogianagherman... 0719526456 Bucuresti
Henry Habuc henryhabuc@gmail... 0748125469 Constanta
Jerry Jackson jerryjackson@gmai... 0721321144 Blaj
Lucian Londroman lucianlondroman@... 0716482333 Teius
Nicolae Nicoara nicolaenicoara@g... 0794632152 Sibiu
Oana Olanescu oanaolanescu@gm... 0732132456 Medias

4.11. Clasele pachetului *presentation.view.general*

4.11.1. *DeleteGeneral<T>*

Clasa pentru definirea modelului general de fereastra de ștergere dintr-un tabel din baza de date

4.11.2. *ViewGeneral<T>*

Clasa pentru definirea modelului general de fereastra de vizualizare a tuturor datelor dintr-un tabel din baza de date

4.12. Clasele pachetului *presentation.view.order*

4.12.1. *OrderClientModel*

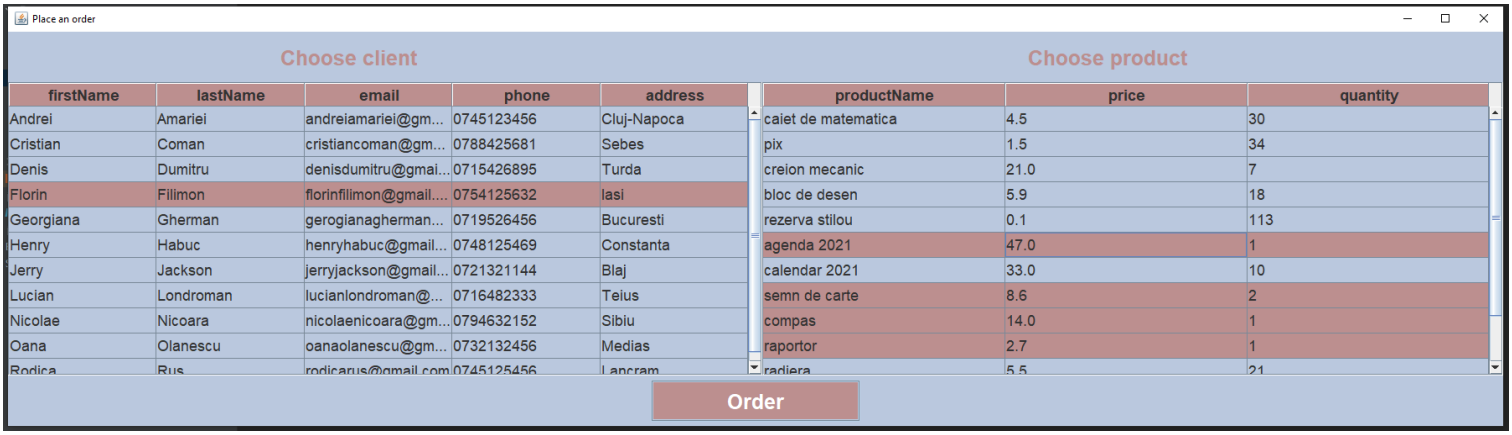
Clasa pentru crearea modelului abstract de tabel al JTable-ului de clienti din interfata grafica de introducere a unei comenzi in baza de date

4.12.2. *OrderFrame*

Clasa pentru crearea interfetei grafice de introducere a unei comenzi in baza de date

4.12.3. *OrderProductModel*

Clasa pentru crearea modelului abstract de tabel al JTable-ului de produse din interfata grafica de introducere a unei comenzi in baza de date



firstName	lastName	email	phone	address
Andrei	Amariei	andreiamariei@gm...	0745123456	Cluj-Napoca
Cristian	Coman	cristiancoman@gm...	0788425681	Sebes
Denis	Dumitru	denisdumitru@gmai...	0715426895	Turda
Florin	Filimon	florinfilimon@gmail....	0754125632	Iasi
Georgiana	Gherman	gerogianagherman...	0719526456	Bucuresti
Henry	Habuc	henryhabuc@gmail...	0748125469	Constanta
Jerry	Jackson	jerryjackson@gmail...	0721321144	Blaj
Lucian	Londroman	lucianlondroman@...	0716482333	Teius
Nicolae	Nicoara	nicolaenicoara@gm...	0794632152	Sibiu
Oana	Olanescu	oanaolanescu@gm...	0732132456	Medias
Rodica	Rus	rodicarus@gmail.com	0745125456	Lancram

productName	price	quantity
caiet de matematica	4.5	30
pix	1.5	34
creion mecanic	21.0	7
bloc de desen	5.9	18
rezerva stilou	0.1	113
agenda 2021	47.0	1
calendar 2021	33.0	10
semn de carte	8.6	2
compas	14.0	1
raportor	2.7	1
radiera	5.5	21

4.13. Clasele pachetului *presentation.view.product*

4.13.1. *AddProduct*

Clasa pentru crearea interfetei grafice de adaugare a unui nou produs in baza de date

4.13.2. *DeleteProduct*

Clasa pentru crearea interfetei grafice de stergere produs din baza de date Particulatizeaza fereastra generala de stergere din baza de date

4.13.3. *EditProduct*

Clasa pentru crearea interfetei grafice de editare produs in baza de date

4.13.4. *EditTableModelProduct*

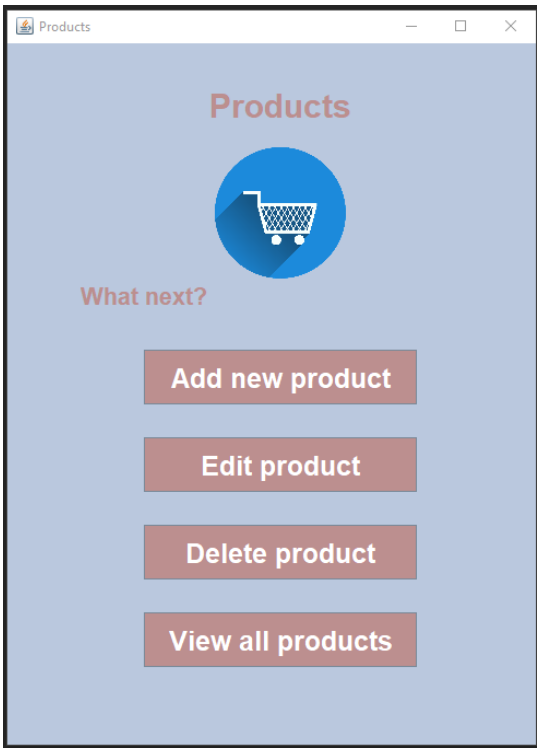
Clasa pentru crearea modelului abstract de tabel al JTable-ului din interfata grafica de editare produs in baza de date

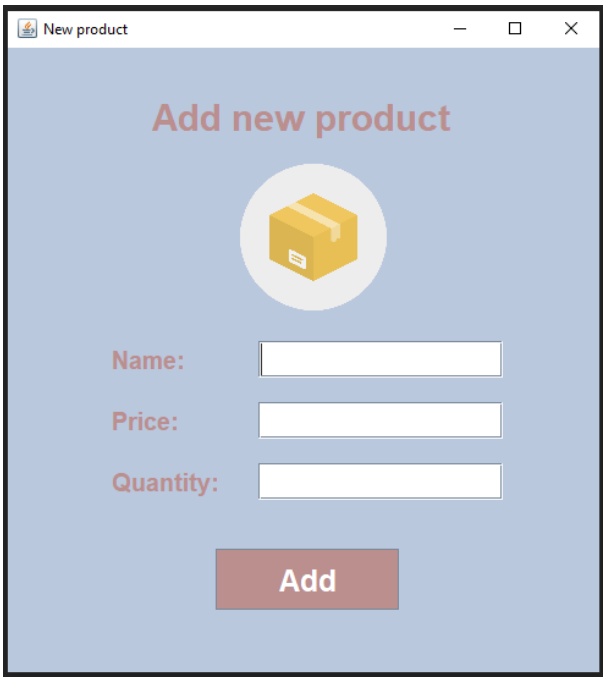
4.13.5. *ProductFrame*

Clasa pentru crearea interfetei grafice de selectare a unei operatii pe tabelul Product din baza de date

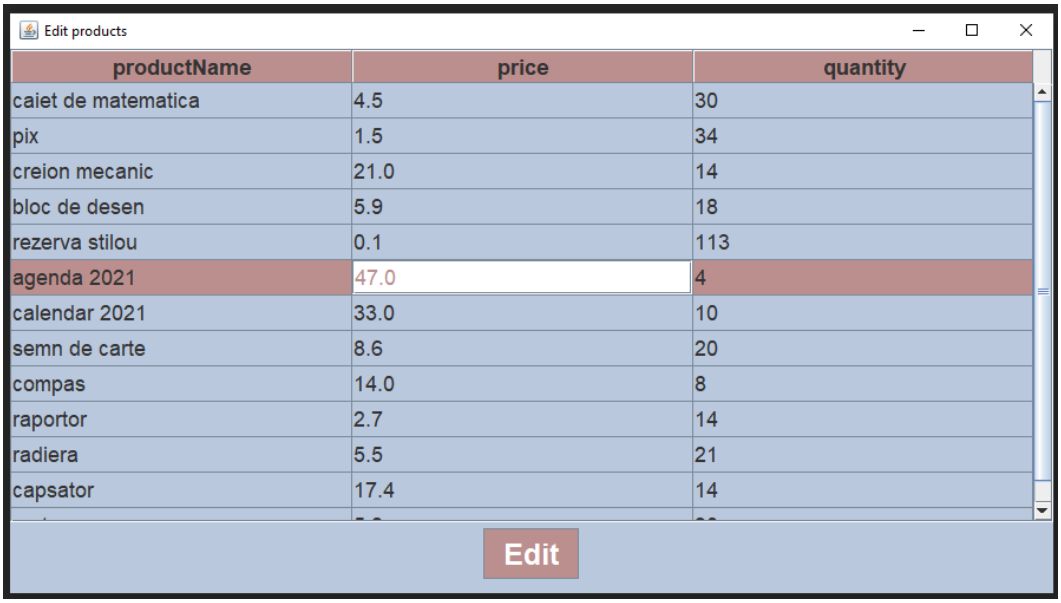
4.13.6. *ViewProduct*

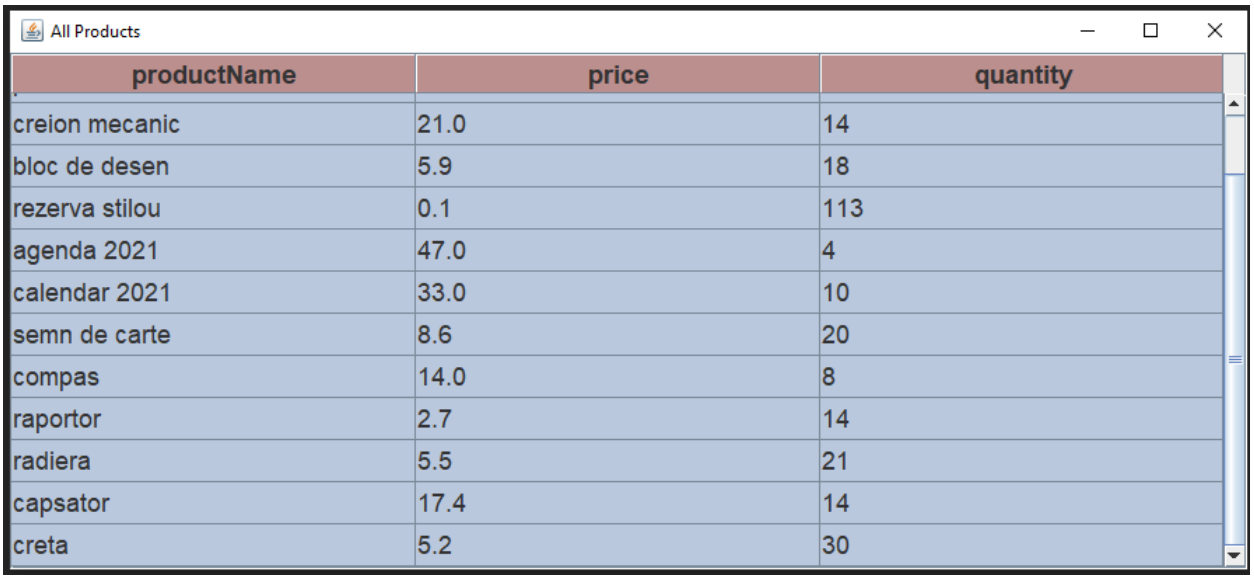
Clasa pentru crearea interfetei grafice de vizualizare a tuturor produselor din baza de date Particulatizeaza fereastra generala de vizualizare a tuturor intrarilor unui tabel din baza de date






Delete Products
productName
price
quantity
caiet de matematica
4.5
30
pix
1.5
34
creion mecanic
21.0
14
bloc de desen
5.9
18
rezerva stilou
0.1
113
agenda 2021
47.0
4
calendar 2021
33.0
10
semn de carte
8.6
20
compas
14.0
8
raportor
2.7
14
Delete


Edit products
productName
price
quantity
caiet de matematica
4.5
30
pix
1.5
34
creion mecanic
21.0
14
bloc de desen
5.9
18
rezerva stilou
0.1
113
agenda 2021
47.0
4
calendar 2021
33.0
10
semn de carte
8.6
20
compas
14.0
8
raportor
2.7
14
radiera
5.5
21
capsator
17.4
14
Edit


All Products
productName
price
quantity
creion mecanic
21.0
14
bloc de desen
5.9
18
rezerva stilou
0.1
113
agenda 2021
47.0
4
calendar 2021
33.0
10
semn de carte
8.6
20
compas
14.0
8
raportor
2.7
14
radiera
5.5
21
capsator
17.4
14
creta
5.2
30

4.14. Clasele pachetului *start*

4.14.1. *MainClass*

Clasa principala a programului, singura clasa runnable. Aceasta este metoda care se va rula pentru punerea în funcțiune a programului.

5. Rezultate

În urma definitivării proiectului, s-au realizat o mulțime de teste pentru verificare corectitudini și a consistenței aplicației Java cu baza de date MySQL. Mai mult decât atât, în cazul plasării unei comenzi se generează și un fișier .txt confirmare a comenzii, cu rol de factură/bon. Numele fișierului indică a câta comandă plasată în baza de date este cea curentă, considerând, în acest caz, comenzile cu oricâte produse.

Exemplu de fișier de ieșire .txt cu rol de confirmare a comezii/factură:

Order 9

Florin, thank you for your order!

We will provide further details by mail, to florinfilimon@gmail.com.

Your products will be delivered to Iasi in no time.

On the day of the delivery, we will send a text message to 0754125632 to let you know your products are near.

Here is a quick overview of your order:

Product 1: agenda 2021 47.0 RON x 1

Product 2: semn de carte 8.6 RON x 2

Product 3: compas 14.0 RON x 1

Product 4: raportor 2.7 RON x 1

TOTAL: 80.9 RON

6. Concluzii

Tema de față a prezentat elemente de noutate în primul rând prin folosirea unor clase specifice și a unui model clar și ordonat al interacțiunii Java-MySQL. Folosirea JTable a fost, de asemenea, un lucru nou învățat.

Elementele de grafică au fost, de asemenea, o provocare în realizarea proiectului, în special prin realizarea tabelelor și stabilirea zonelor editabile din acestea.

Alte direcții de dezvoltare ulterioară a proiectului ar putea include:

- îmbunătățirea modului de editare a datelor unui client/produs; să nu mai fie nevoie de selectarea rândurilor ce au fost în prealabil editate, ci la apăsarea butonului Edit să se proceseze toate înregistrările la care s-au efectuat modificări
- îmbunătățire modului de introducere a unei comenzi noi; să nu mai fie nevoie de selectarea rândurilor ce marchează produsele dorite, ci să se ia în calcul automat cele ale căror cantități au fost modificate
- introducerea în interfața grafică, în fereastra de plasare a unei comenzi, în tabelul de produse, a unui element *Spinner* pentru incrementarea sau decrementarea cantității comandate
- adăugarea unor efecte 3D butoanelor ca upgrade al interfeței cu utilizatorul

7. Bibliografie

- TP2020-2021_Descriere_Laborator.pdf
- ASSIGNMENT_3_SUPPORT_PRESENTATION.pdf
- Tema3.pdf
- https://gitlab.com/utcn_dsrl/pt-layered-architecture
- https://www.baeldung.com/javadoc
- https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html
- https://www.tutorialspoint.com/java/java_documentation.htm
- https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html
- https://stackoverflow.com/questions/22762144/regular-expression-for-double-values-using-qregexp
- https://stackoverflow.com/questions/290513/regex-pattern-for-numeric-values
- https://regexr.com/3c53v
- https://docs.oracle.com/javase/tutorial/uiswing/components/table.html
- https://www.codejava.net/java-se/swing/editable-jtable-example
- https://stackoverflow.com/questions/29345792/java-jtable-getting-the-data-of-the-selected-row