**TEHNICI DE PROGRAMARE FUNDAMENTALE**

ASSIGNMENT 3

ORDER MANAGEMENT

Vilcelean Dalia

Grupa 30222

Facultatea de Automatica si Calculatoare

**CUPRINS:**

1. Obiectivul temei

2. Analiza temei si modelarea acesteia

2.1. Cazuri de utilizare

2.2. Scenarii

3. MySQL / JDBC

4. Proiectare (Impartirea claselor in pachete)

4.1. Pachetul bll.validators

a) Clasa AbstractBLL

b) Clasa ClientBLL

c) Clasa DateValidator

d) Clasa EmailValidator

e) Clasa FileOrder

f) Clasa OrdersBLL

g) Clasa PhoneValidator

h) Clasa ProdusBLL

i) Interfata Validator

4.2. Pachetul connection

a) Clasa ConnectionFactory

4.3. Pachetul dao

a) Clasa AbstractDAO

b) Clasa ClientDAO

c) Clasa OrdersDAO

d) Clasa ProduseDAO

4.4. Pachetul gui

a) Clasa Controller

b) Clasa View

4.5. Pachetul model

c) Clasa Client

e) Clasa Orders

f) Clasa Produse

4.6. Pachetul start

a) Clasa Main

5.Rezultate

6.Concluzii

7. Dezvoltari ulterioare

1. **Obiectivul Temei**

Pentru aceasta tema am avut de realizat o aplicatie OrderManagement care proceseaza comenzile unor client pe un site web.

Pentru ca aceasta aplicatie sa fie eficienta, va trebui sa reuseasca sa inregistreze comenzile unor client, s chiar sa realizeze un bon pentru acestea.

Obiectivul acestei teme este de a conecta o baza de date unde am inregistrat toti posibilii clienti, produsele pe care le ofera magazinul nostrum si posibilele comenzi pe care le vor face clientii in viitor.

Unul dintre obiectivele secunare ale acestei aplicatii este de a scoate “un bon fiscal” pentru comenzile realizate cu success prin crearea unui fisier .txt. Un al doilea obiectiv este crearea arhitecturi pe straturi a proiectului, mult mai accesbila si usor de citit.

1. **Analiza Temei si Modelarea Acesteia**

Luați în considerare o aplicație OrderManagement pentru procesarea comenzilor client pentru un depozit. Bazele de date relaționale sunt utilizate pentru a stoca produsele, clienții și comenzile.

În plus, aplicația ar trebui să fie structurată în pachete care utilizează o arhitectură stratificată și ar trebui să utilizeze (minim) următoarele clase:

Clasele de model

-reprezintă modelele de date ale aplicației

 Clasele de bussinessLogic

-conțin logica aplicației

 Clasele de prezentare

-Clasele legate de GUI

Clasele de acces la date

- clase care conțin accesul la baza de date

Aplicatia pe care o voi crea va trebui sa contina:

-Proiectare orientată spre obiecte

- Utilizați Javadoc pentru clase de documentare și generați fișierele JavaDoc corespunzătoare.

- Utilizați baze de date relaționale pentru stocarea datelor pentru aplicație

- Creați o interfață grafică cu utilizatorul care include:

o O fereastră pentru operațiunile clientului: adăugați un client nou, editați clientul, ștergeți clientul, vizualizați toți clienții dintr-un tabel (JTable)

o O fereastră pentru operațiile produsului: adăugați un produs nou, editați produsul, ștergeți produsul, vizualizați toate produsele dintr-un tabel (JTable)

o O fereastră pentru crearea comenzilor de produs - utilizatorul va putea selecta un produs existent, selecta un client existent și introduce o cantitate dorită pentru ca produsul să creeze o comandă validă. În cazul în care nu există suficiente produse, va fi afișat un mesaj sub stoc. După finalizarea comenzii, stocul produsului este diminuat.

* Utilizați tehnici de reflecție pentru a crea o metodă care primește o listă de obiecte și generează antetul tabelului prin extragerea prin reflexie a proprietăților obiectului și apoi populează tabelul cu valorile elementelor din listă.
* Creați o factură pentru fiecare comandă ca fișier text sau fișier .pdf

*CAZURI DE UTILIZARE:*

Interfata grafica va fi creata astfel incat sa fie User- Friendly pentru utilizatori, pentru a putea naviga cat mai usor posibil prin ea. Aceasta va avea o fereastra unde se vor putea realiza comenzi de produse pe care le putem oferi noi in aplicatie. User-ul va putea realiza comanda, va putea introduce mai multe obiecte in cosul de cumparaturi, va vedea pretul total si i se va elibera pana si o factura pentru comanda facuta.

Tot in aplicatie se vor putea insera, sterge, updata clientii si produsele cu ajutorul unor butoane specifice.

1. **MySQL**

MySQL este un MySQL este un sistem de gestiune a bazelor de date relaționale. Este cel mai popular SGBD open-source la ora actuală. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum ar fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc., fiecare dintre acestea folosind un tip specific API.

MySQL este mult mai ușor de învățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau, prin descărcare de pe internet, o interfață grafică: MySQL Administrator și MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, phpMyAdmin.

*MySQL Workbench:*

MySQL Workbench este mediul integrat pentru MySQL. A fost dezvoltat de MySQL AB și permite utilizatorilor să administreze grafic bazele de date MySQL și să proiecteze vizual structuri de baze de date.

MySQL Workbench este disponibil în două ediții, ediția comunitară gratuită și open source care poate fi descărcată de pe site-ul web MySQL și ediția standard proprietară care extinde și îmbunătățește setul de caracteristici ale ediției comunitare.

*MySQL Connector/j*

MySQL oferă conectivitate pentru aplicațiile client dezvoltate în limbajul de programare Java cu MySQL Connector / J. Connector / J implementează API-ul Java Database Connectivity (JDBC), precum și o serie de extensii cu valoare adăugată ale acestuia. De asemenea, acceptă noul X DevAPI.

MySQL Connector / J este un driver JDBC de tip 4, implementând specificația JDBC 4.2. Denumirea de tip 4 înseamnă că driverul este o implementare Java pură a protocolului MySQL și nu se bazează pe bibliotecile client MySQL.

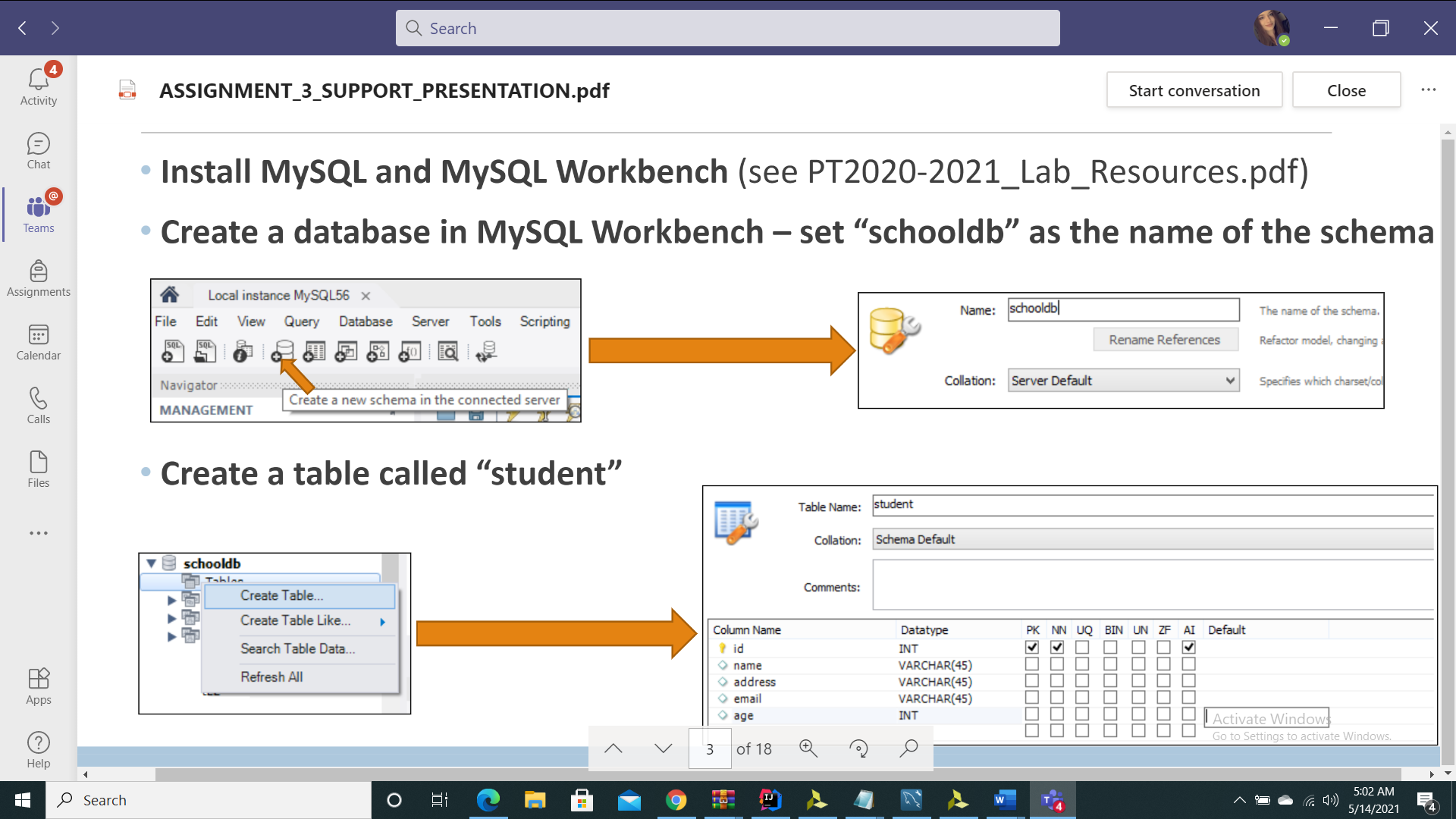
Connector / J 8.0 oferă caracteristici ușoare de dezvoltare, inclusiv înregistrarea automată cu Driver Manager, verificări de validitate standardizate, excepții SQLE clasificate, suport pentru actualizări mari, suport pentru variante de dată și oră locale și offset din pachetul java.time, suport pentru JDBC -4.x Procesare XML, suport pentru informațiile clientului pe conexiune și suport pentru tipurile de date NCHAR, NVARCHAR și NCLOB.

*BAZELE JDBC:*

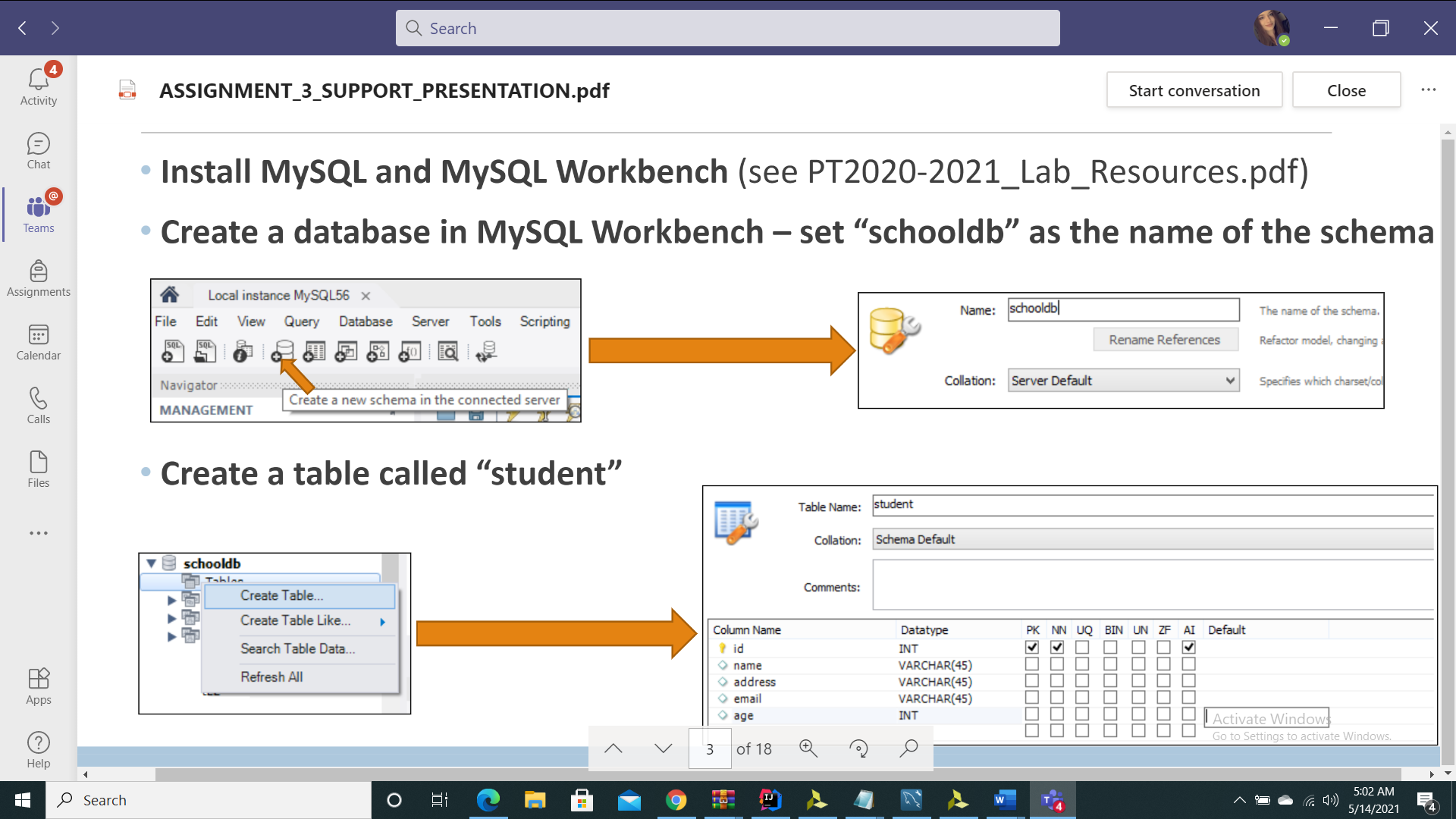
Pentru a putea lucre cu JDBC, va trebui sa:

-Instalam MySQL si MySQL Workbench

-Cream o baza de date noua in MySQL Workbench



-Cream un table in baza de date deja existenta



Pași:

• Stabiliți o conexiune cu sursa de date

• Creați o declarație

• Executați interogarea

• Procesați ResultSetobject

• Închideți conexiunea

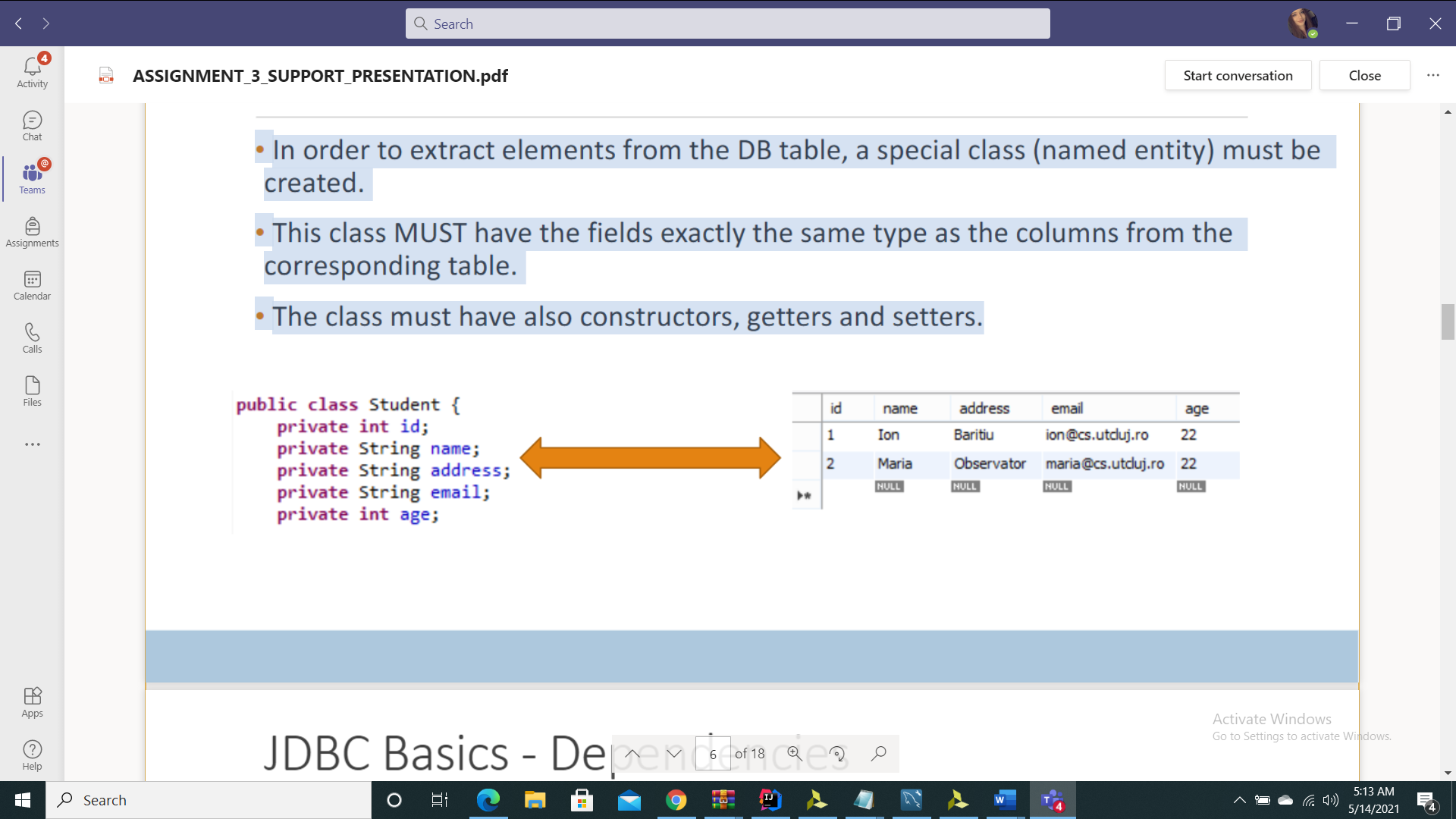
Conexiunea se va realiza cu ajutorul clasei ConnectionFactory (vom vorbi despre ea in detaliat in capitolele urmatoare).

*Asocierea tabelelor:*

• Pentru a extrage elemente din tabelul DB, trebuie creată o clasă specială (denumită entitate).

• Această clasă TREBUIE să aibă câmpurile exact de același tip cu coloanele din tabelul corespunzător.

• Clasa trebuie să aibă și constructori , getters și setters.



*Dependențe*:

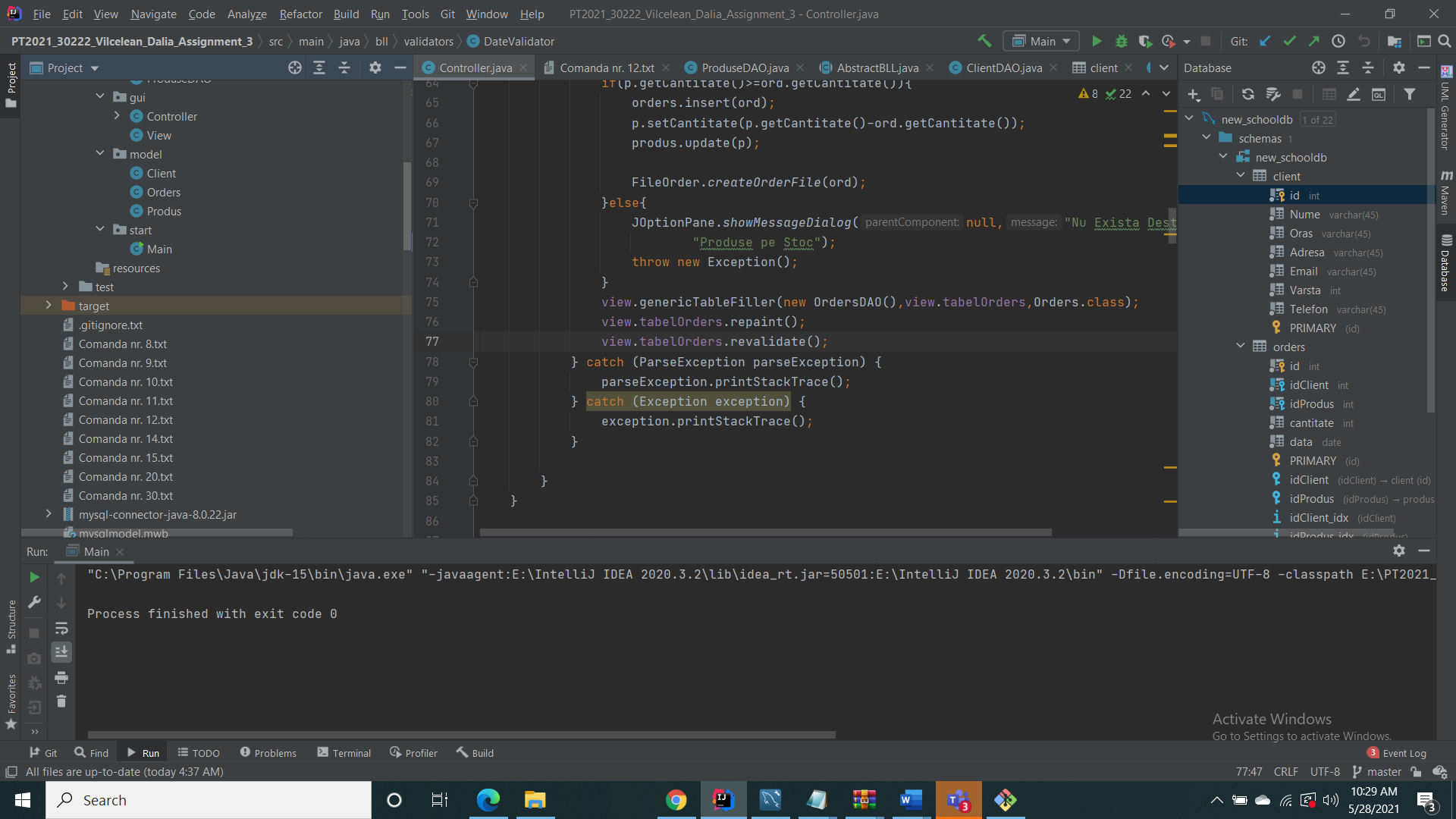
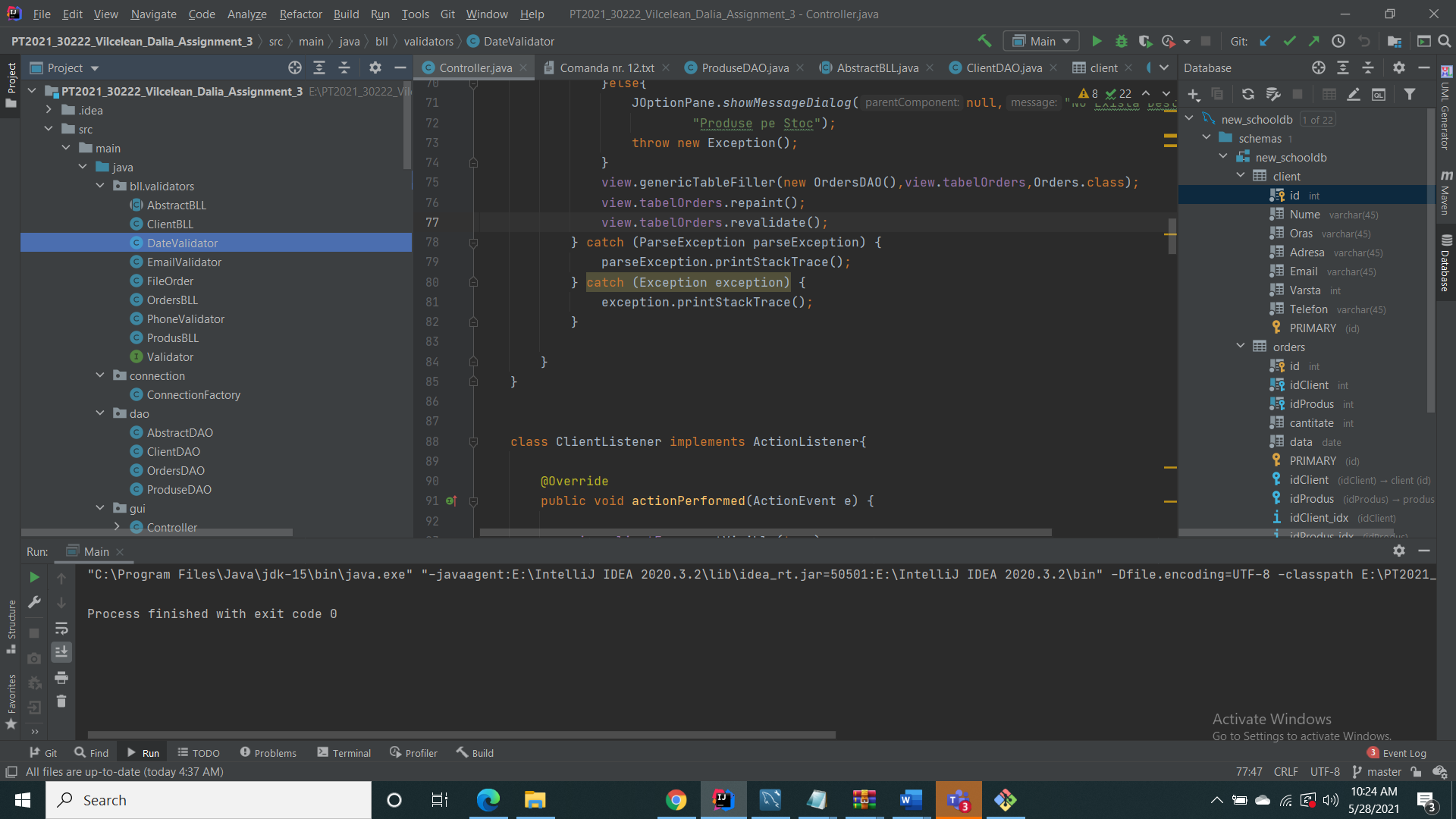
• Pentru ca aplicația Java să interacționeze cu DB, trebuie adăugată o aplicație .jar library specială la aplicație

• Poate fi adăugată fie ca dependență de fișier jar extern, fie ca dependență maven, în cazul unui proiect Maven

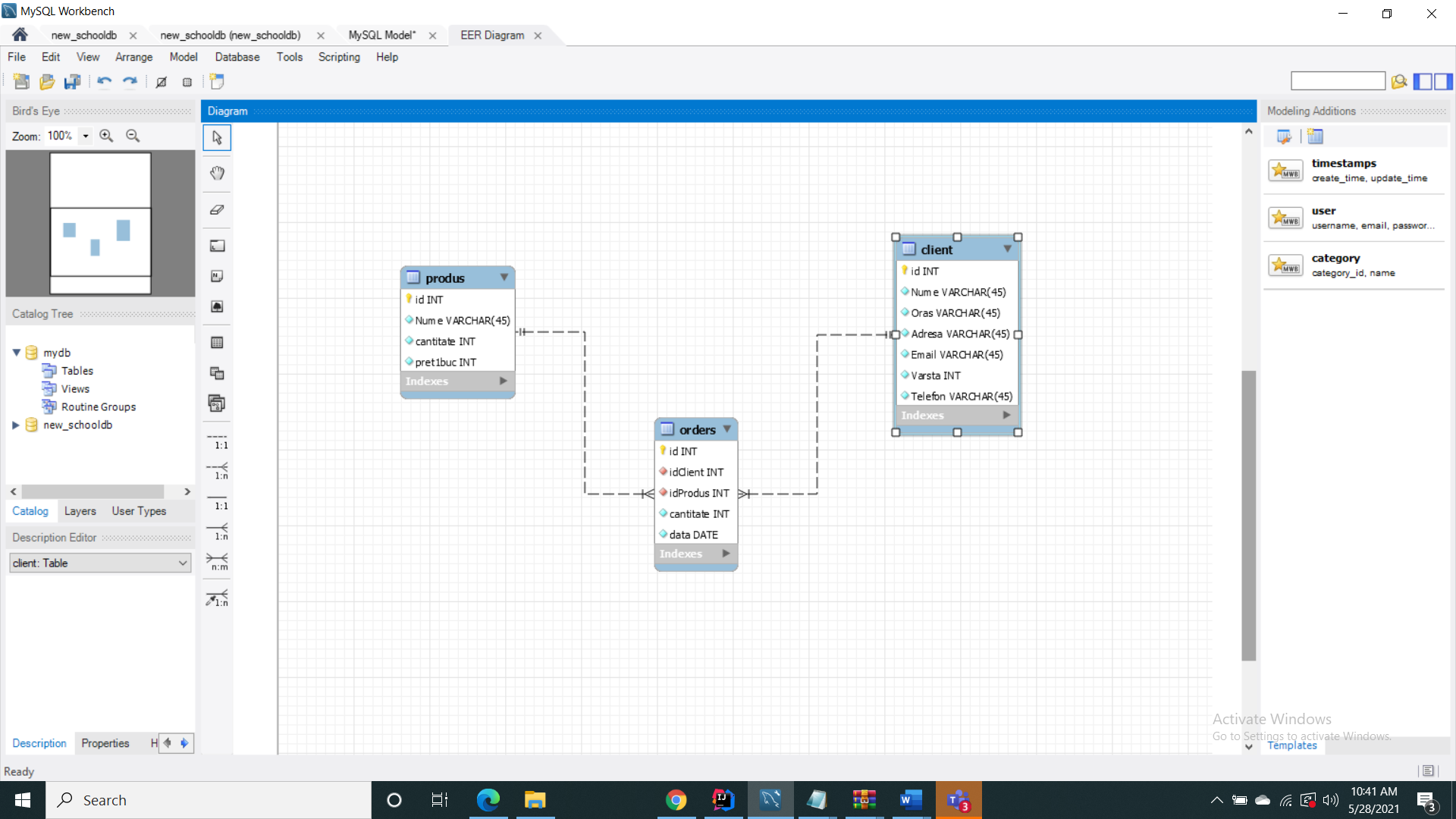
• Aplicația Java folosește această bibliotecă externă pentru a comunica cu serverul MySQL

• Trimite interogări către server folosind Declarații și primește rezultatele interogărilor ca ResultSet

1. **PROIECTARE**



In cadrul programului MySQL amm realizat o schema de tabele, pe care le-am legat intre ele cu ajutorul cheilor primare si cheilor straine.



Dupa cum se poate observa din schema realizata, tabelul client are o cheie primara “id” care este legata de cheia secundara “idClient” din tabelul orders, astfel realizandu-se o legatura intre ele.

La fel ca la client, si tabelul produs are o cheie primara “id” care este legata de cheia secundara “idProdus” a tabelului orders, in acest fel am facut ca tabelul orders sa fie dependent si de produse, si de clienti.

Dupa ce am realizat cu succes schema de tabele, am inceput sa populez tabelul cu diferite valori( nume, varsta, adresa, telefon, nume produs etc) pentru a le prelucra mai departe.

Pentru acest proiect a trebuit sa lucrez cu o arhitectura organizata pe pachete, fiecare pachet indeplinindu-si propria lui functie. Va urma sa descriu fiecar pachet in parte si functia pe care acesta o indeplineste:

**Pachetul connection**

Acest pachet contine o sngura clasa sireprezinta conexiunea dintre Java si MySQL Workbench, unde am creat o schema de tabele, pentru a le putea prelucra in continuare.

In aceasta clasa am facut mai multe metode, de care ma voi folosi atunci cand am nevoie de valori din baza de date, cum ar fi:

createConnection(), getConnection(), close().

Pentru a putea aplicatia java sa interactioneze cu baza de date, a trebuit sa includ in proiect un fisier .jar care realizeaza conexiunea propriu-zisa cu baza de date.

**Pachetul model**

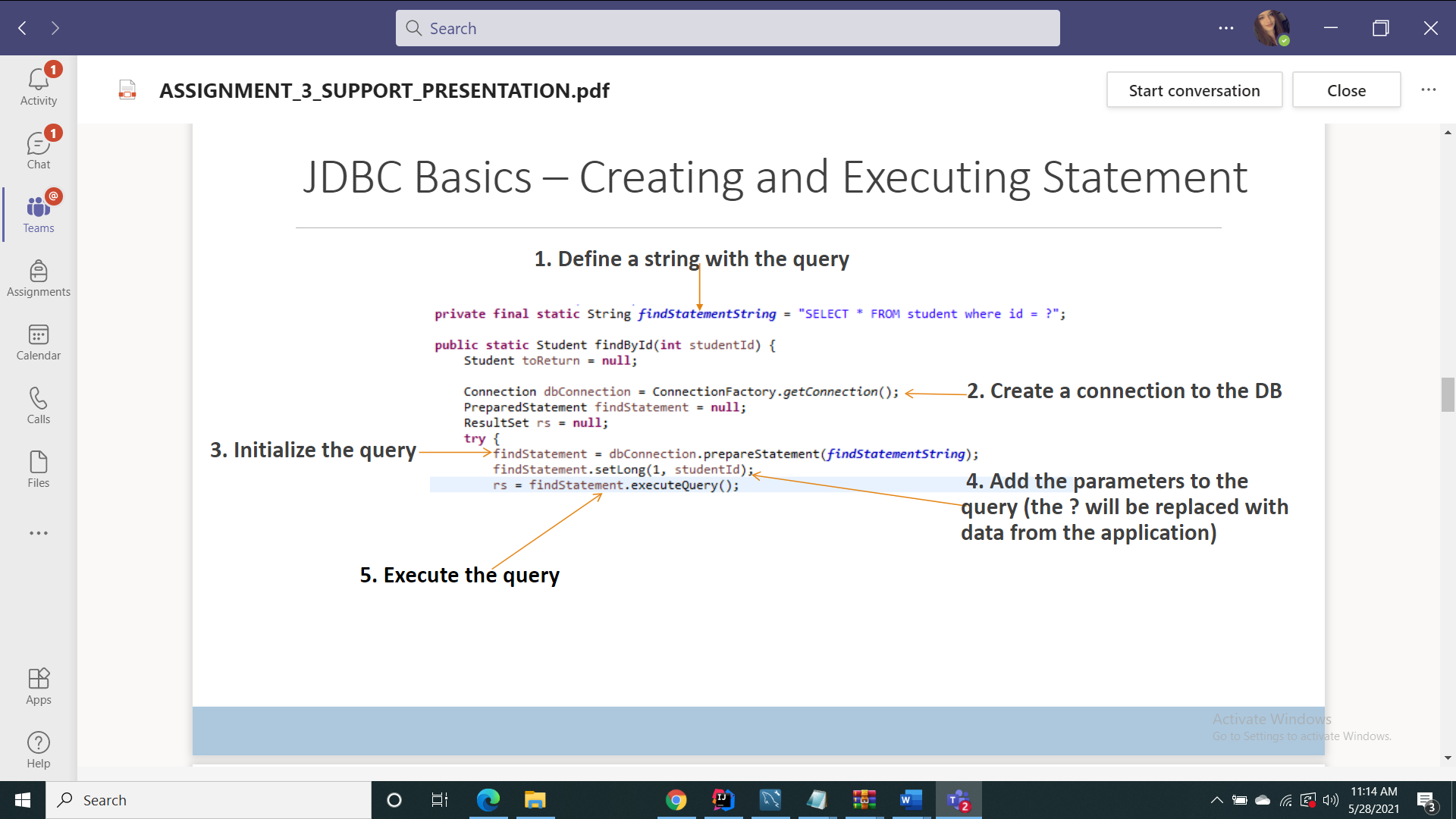
Luandu-le dupa ordinea nivelelor create in aceasta arhitectura, acesta este urmatorul pachet. Acesta contine clasele: Client, Produs, Orders.

Pentru a putea extrage elemente din baza de date, aceste clase au trebuit sa fie create. Variablele trebuie sa fie exact aceleasi ca si in tabelul corespunzator clasei, cu constructor, gettere si settere.

**Pachetul dao**

Acest pachet contine clase care acceseaza datele din tabel si le prelucreaza.

AbstractDAO este clasa “parinte” a acestui pachet, deoarece aceasta contine toate metodele necesare aplicatiei, si le trimite prin mostenire tuturor celorlalte clase.



Metodele clasei sunt:

- private String createSelectQuery(String field)

- private String createSelectAllQ()

- private String createInsertQuery(T t)

- private String createUpdateQuery(String field)

- private String createDeleteQuery(String field)

- public List<T> findAll()

- private List<T> createObjects(ResultSet resultSet)

- public T findById(int id)

- public int insert(T t)

- public T update(T t)

- public int delete(int id)

Toate aceste clase prelucreaza datele din tabele, cu ajutorul StringBuilder.

Am realizat toate cele 3 metode obligatorii, update(), insert() si delete(), cu ajutorul createInsertQuery(), createUpdateQuery() si createDeleteQuery(), unde am folosit fnctiile MySQL “INSERT INTO ….”, “UPDATE…….”, “DELETE FROM ………. WHERE………”.

Metoda findById() este foarte importanta deoarece aceasta poate gasi un rand din tabel in functie de id-ul acestuia, iar findAll() va stoca toate datele existente in tabel ca si obiecte.

Celelalte clase din acest pachet, ClientDAO, OrderDAO si ProduDAO nu contin nicio metoda, deoarece acestea le mostenesc de la clasa principala abstracta AbstractDAO, si le transmite parametrul T prin care se va mentiona clasa aleasa.

**Pachetul bll.validators:**

Acest pachet este, practic,cel mai important nivel al arhitecturii. Acesta primeste datele din toate pachetele pe care le-am descris pana acum si le prelucreaza, ca un pas final, pentru a fi, in sfarsit, transmise catre interfata grafica.

Pachetul contine, in primul rand, o interfata Validator, unde se valideaza textul anumitor coloane. Eu am ales sa fac o clasa DateValidator, unde am facut Match de Pattern pentru ca textul sa aiba forma unei date “yyyy-mm-dd”, clasa EmailValidator, care verifica textul sa aiba forma unui e-mail, adica sa exite un “@” in textul respectiv, si clasa PhoneValidator, care verifica textul sa aiba forma unui numar de telefon, adica un sir de 10 cifre.

Am introdus si in acest pachet o clasa “parinte” AbstractBLL, pentru a putea mosteni metodele comune ale celorlalte clase.

AbstractBLL contine metodele:

- public boolean validateAll(T t)

- public int insert(T t)

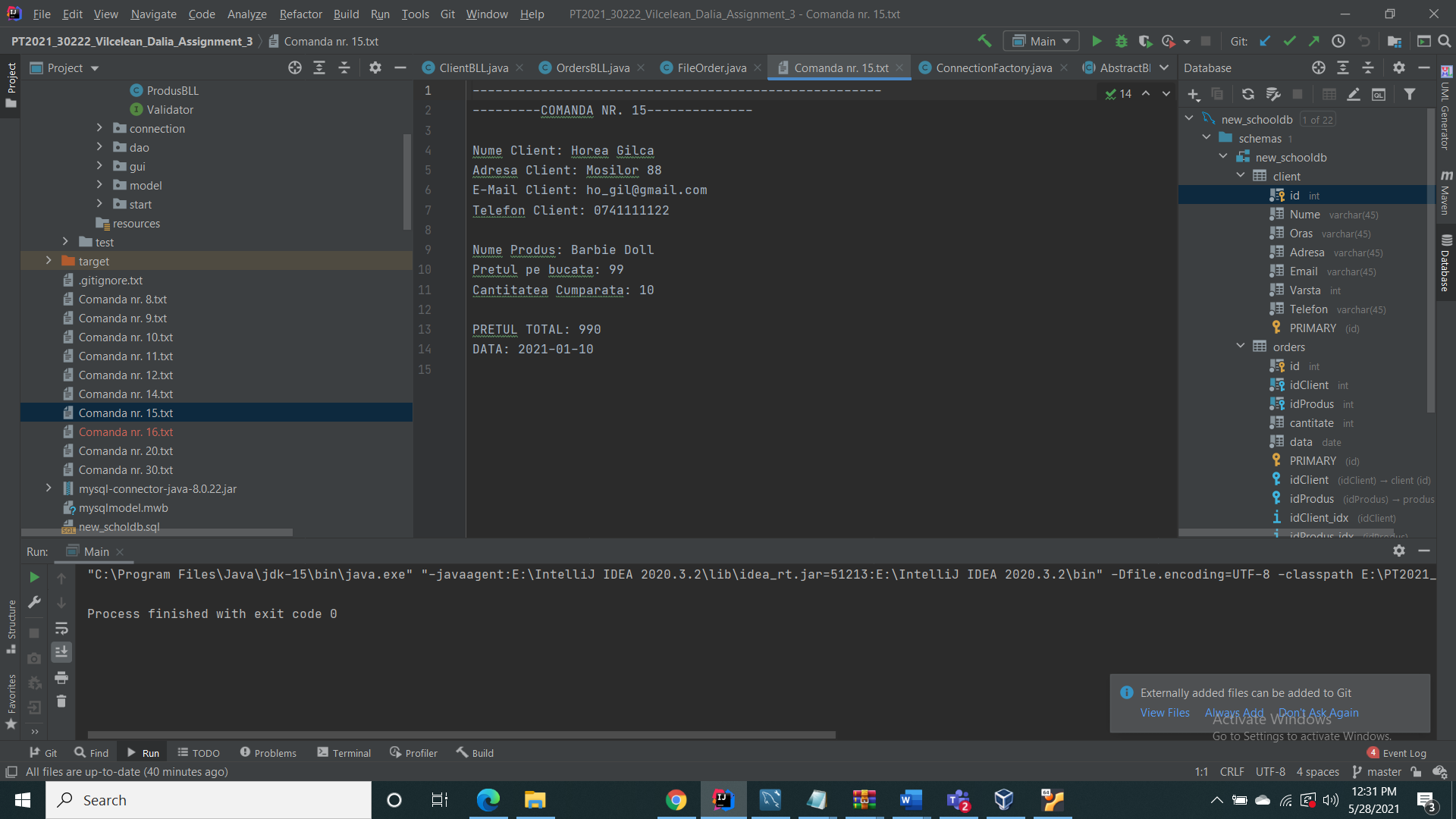
- public int delete(int id)

Am creat metoda validateAll pentru a putea folosi interfata Validator, imreuna cu clasele ei.

Clasele care motenesc AbsractBLL, adica ProdusBLL, ClientBLL si OrdersBLL au implementate si ele metode particulare si, pentru fiecare, validatorul de text respectiv.

Acest pachet mai contine inca o clasa, FileOrder, care a fost facuta special pentru eliberarea bonului fiscal, printr-un fisier.txt.

In aceasta clasa am declarat un PrintWriter, iar apoi am scris in fisier toate detaliile importante ale comenzii.



**Pachetul gui**

Acesta este pachetul care se afla la primul nivel din arhitectura, reprezentand front-end-ul aplicatiei.

Clasele existente in acest pachet sunt: View si Controller.

Clasa View: In aceasta clasa am modelat aspectul interfetei grafice, am plasat butoanele la locurile lor potrivite, am creat frame-uri etc.

Pentru o viziune de ansmblu mai buna, am creat o pasina principala care contine 3 butoane: Clienti, Produse si Comenzi.

Prin apasarea oricarui buton, se va ajunge la tabelul respectiv si instructiunile proprii.

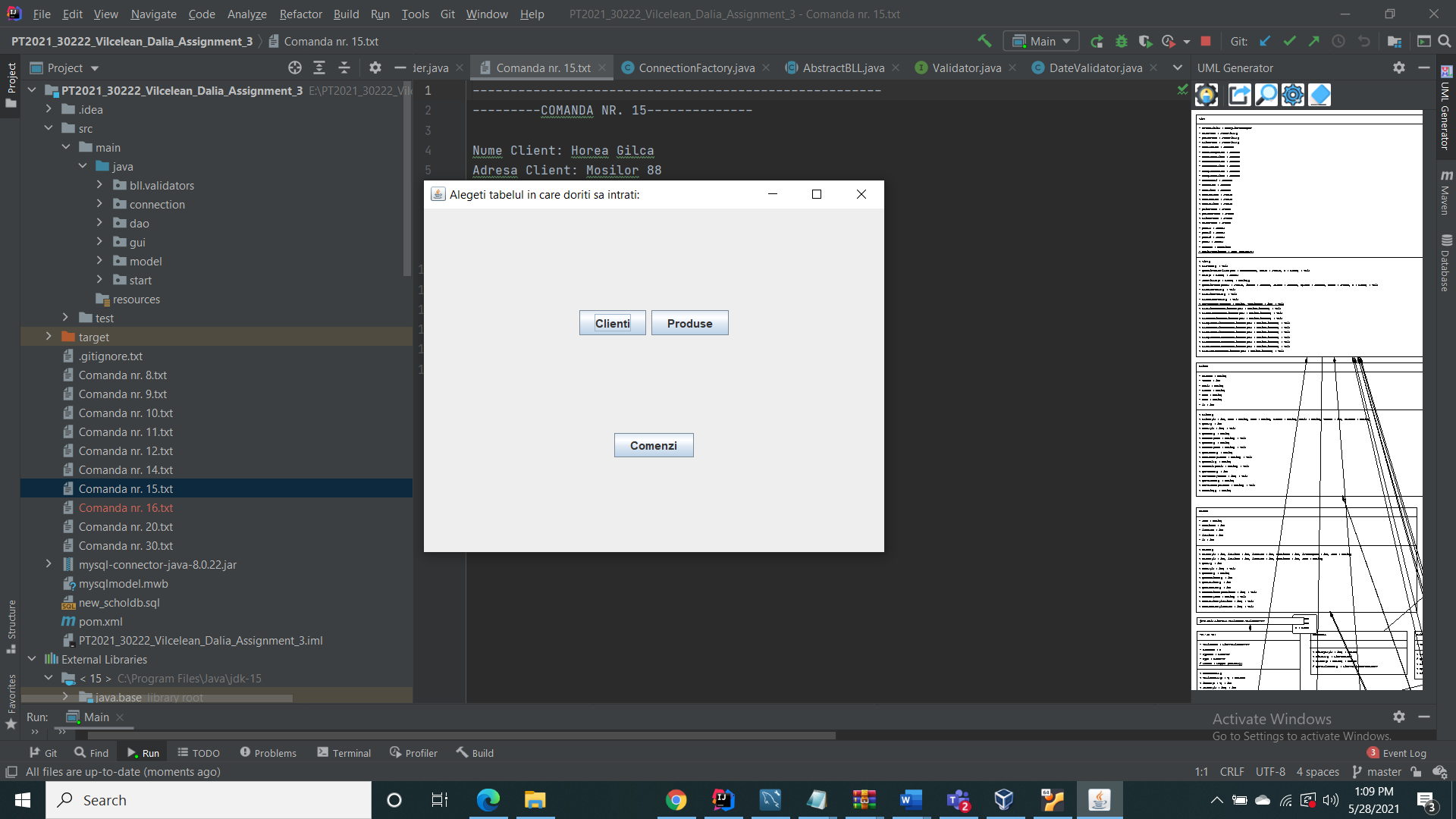
Clasa Controller contine alte clase interne pentru fiecare buton pe care il vom avea in aplicatie, toate acestea continand un ActionListener.

**Pachetul start:**

In acest pachet se afla clasa Main, care contine metoda public static void main(String[] args), de unde incepe functionarea intregului program.

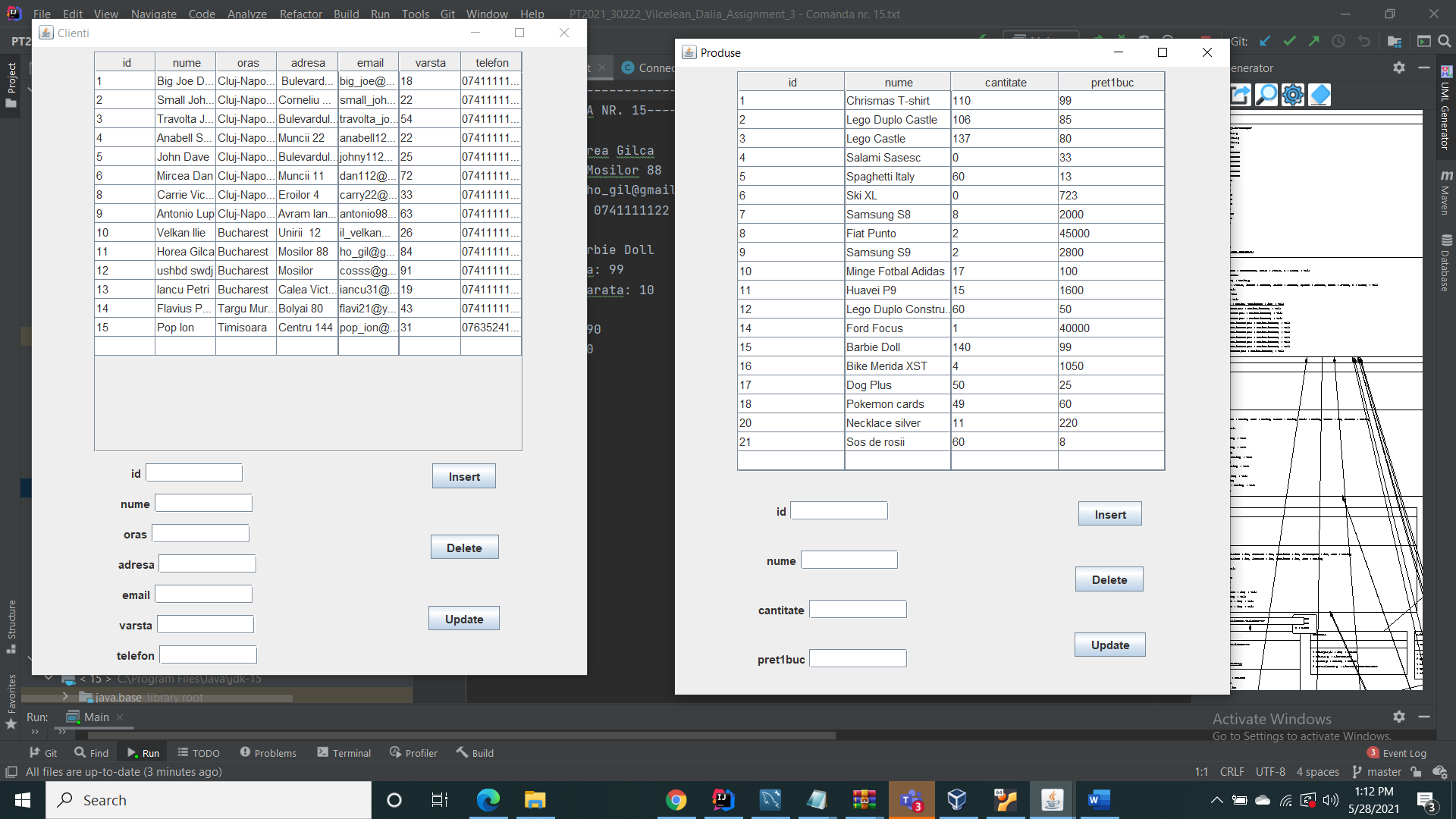
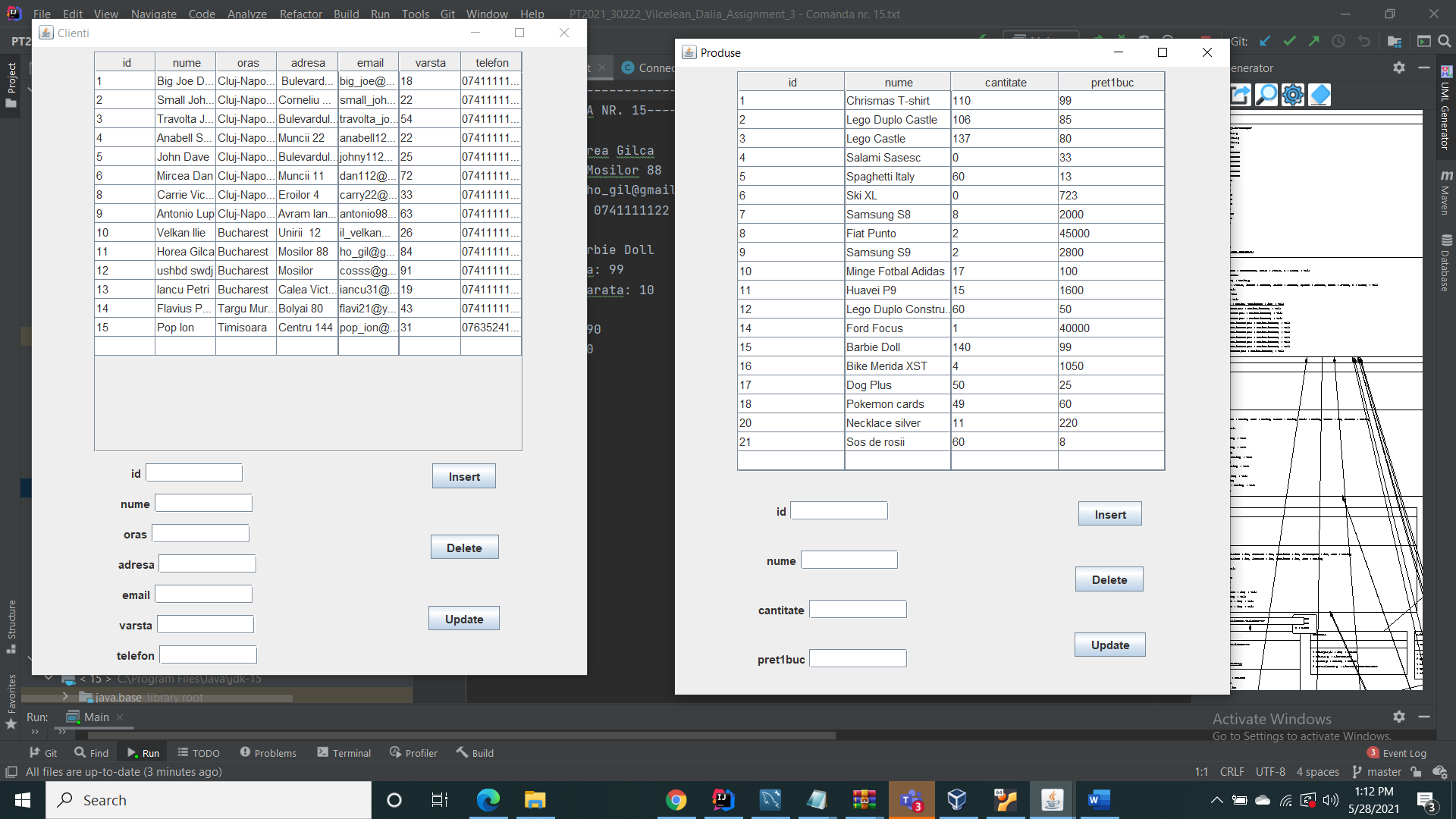
In interiorul acesteia am declarat clasa View ca si un obiect, la care i-am dat ca variabila clasa Controller, care se ocupa cu butoanele s ActionListener-urile.

1. **REZULTATE**



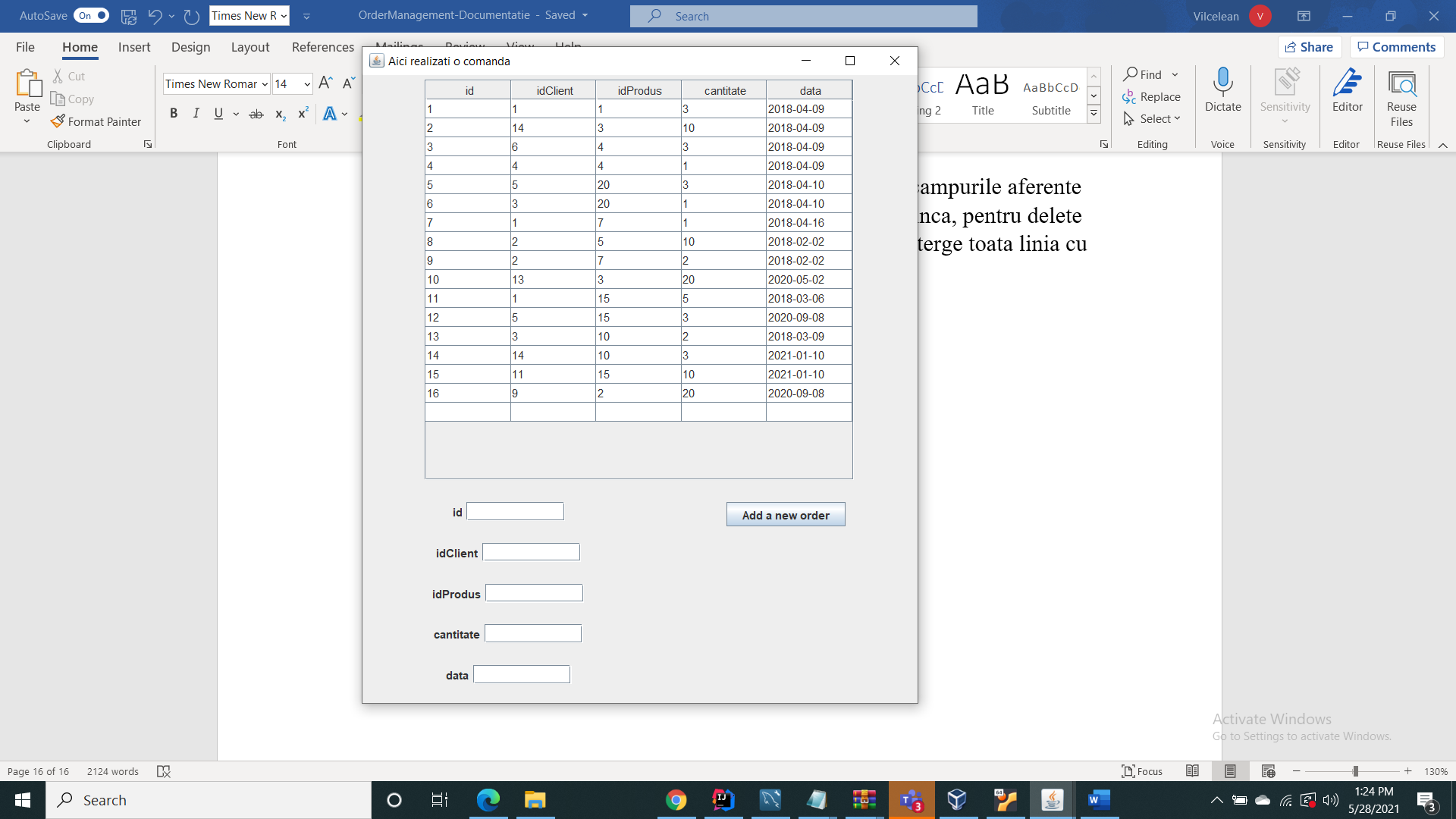
Aceasta este fereastra principala, cate contine 3 butoane.

La apasarea oricarui dintre aceste butoane, se va deschide alt panel, dedicat tabelului propriu.

Acestea sunt panel-urile dedicate clientilor si produselor. Dupa cum se poate observa, amadoua contin butoane de insert, delete si update. Prin aceste butoane, asa cum li se spune numele, pot insera, sterge sau updata campurile din tabel.

Pentru insert si update va trebui sa completam toate campurile aferente tabelului, altfel va aparea un popUp cu mesaj de eroare, inca, pentru delete trebuie sa completam doar campul cu id-ul, si asa se va sterge toata linia cu id-ul respectiv.



Panel-ul pentru comenzi este la fel ca si cel al produselor si clientilor, doar ca acesta contine doar butonul de “Add a new order”.

In campurile libere va trebui sa scriem numarul comenzii, id-ul clientului, id-ul produsului pe care doreste sa-l comande, cantitatea dorita si data.

Odata ce am apaat butonul care adauga comanda, aceasta se va insera in tabel, se va crea un bon printr-un fisier.txt, iar cantitatea pe stoc din tabelul Produse va scadea.

1. **CONCLUZII:**

Cu ajutorul acestui proiect am invatat sa lucrez in Java cu o baza de date, sa parsez informatiile din tabel si sa le trimit interfetei grafice si sa desenez tabelele intr-un panel.

Nu este foarte usor sa detii un site de comenzi on-line, deoarece trebuie sa tii cont de fiecare comanda, de numarul ei si de continutul ei, iar greselile sunt rareori acceptate, din moment ce rezulta niste clienti nesatisfacuti.

Nu este usor nici sa lucrez cu tabele, cu foarte multe date care trebuie stocate, legate de alte tabele etc.

1. **DEZVOLTARI ULTERIOARE**

Nicio aplicatie nu este perfecta din prima, mereu este loc de o imbunatatire, la fel este si in cazul aplicatiei mele.

Ca si dezvoltare ulterioara principala as putea nota posibilitatea clientilor de a comanda mai mult produse, de tipuri dferite, in aceeasi comanda. In momentul de fata se poate adauga doar cate un id pe rand.

O alta dezvoltare ulterioara ar putea fi update-ul tabelului produse dup plasarea unei comenzi imediata, in mometul de fata trebuie sa dau re-run pentru a functiona.