PROIECT – INGINERIA SISTEMELOR SOFTWARE

AN UNIVERSITAR 2019-2020

SEMESTRUL 2

Aplicație

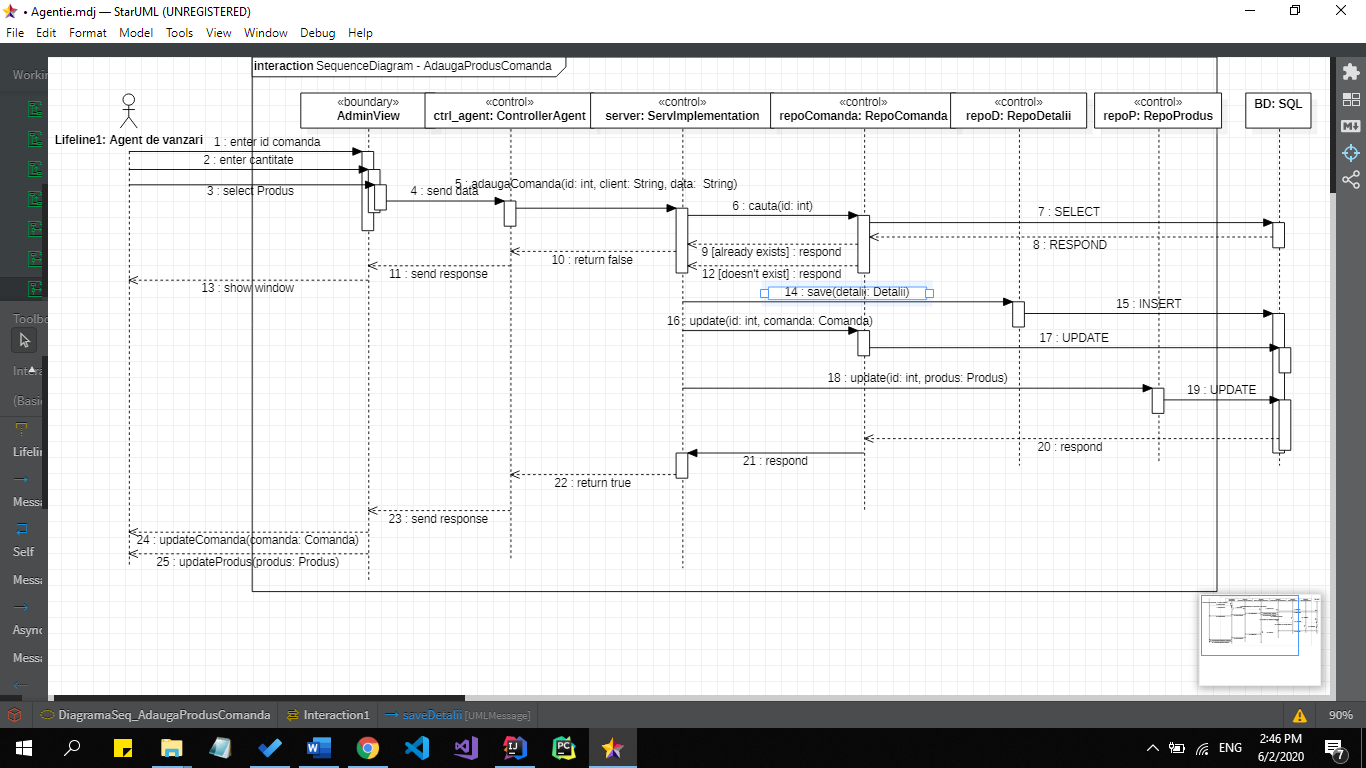
AGENȚI DE VÂNZĂRI

**Nume și prenume Student: Coordonator:**

Iloaia-Pîrvulescu Andreea Prof. Sima Ioan

224/1

Informatică



**Prezentarea cerintei**

Avand in vedere contextul social-economic ce ne-a surprins anul acesta din cauza unei catastrofe biologice neprevazute, stim cat de mult sunt afectati oamenii pe diferite planuri. Vrem sa mentionam ca suntem impreuna in asta si ca lucrurile se vor imbunatati.

Astfel, firma noastra si-a propus in timpul ordonantei de urgenta sa ajute locuitorii comunitatii noastre locale sa nu se mai ingrijoreze de nevoile care i-ar putea obliga sa iasa din casa, iar astfel sa se expuna riscurilor. Astfel am create o aplicatie pentru gestiunea comenzilor facute de diversi agenti pentru clientii nostril.

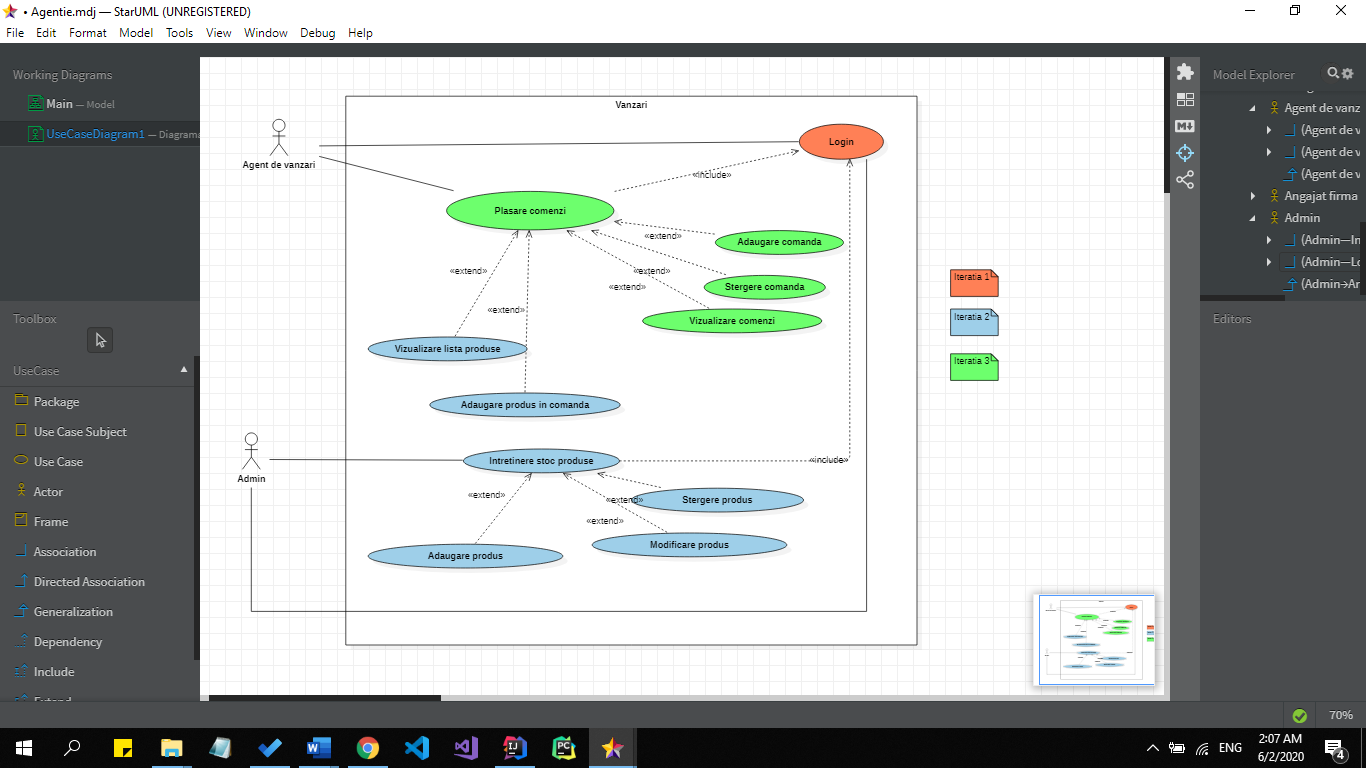
O firma are mai multi agenti de vanzari, care fac comenzi pentru clientii firmei si le distribuie acestora produsele. Pentru desfasurarea activitatilor, firma ofera o aplicatie prin care agentii pot comanda produse. Aplicatia pune la dispozitia fiecarui agent al firmei un terminal prin care:

• agentul vizualizeaza lista tuturor produselor vândute de firma, împreuna cu preturile aferente și cantitatile existente pe stoc.

• agentul poate comanda o cantitate dintr-un produs. După orice comanda valida, toti agentii logati în aplicație vor vedea lista actualizata a stocurilor (este posibil ca declansarea înregistrarii unei comenzi din partea unui agent sa conduca la un mesaj informativ "cantitate insuficienta în stoc").

2. Modelul functional

Diagrama de cazuri de utilizare:



Descrierea cazurilor de utilizare + scenarii alternative:

|  |  |
| --- | --- |
| Nume caz de utilizare | Login admin |
| Actor | Admin |
| Scenariul normal | 1. Adminul introduce datele de autentificare (username si parola) si apasa click pe butonul de login 2. Dupa verificarea credentialelor, sistemul afiseaza fereastra pentru produse |
| Scenariul alternativ | 2.a. Daca nu sunt corecte credentialele, sistemul afiseaza fereastra de eroare |

|  |  |
| --- | --- |
| Nume caz de utilizare | Adauga produs |
| Actor | Admin |
| Scenariul normal | 1. Adminul vede lista de produse existenta 2. Adauga detaliile despre noul produs (id, denumire, pret si cantitate) 3. Adminul apasa pe butonul de adauga 4. Sistemul actualizeaza lista de produse si o afiseaza |
| Scenariul alternativ | 3.a. Daca produsul exista deja, sistemul afiseaza fereastra cu notificarea |

|  |  |
| --- | --- |
| Nume caz de utilizare | Modifica produs |
| Actor | Admin |
| Scenariul normal | 1. Adminul vede lista de produse existenta 2. Adminul apasa pe un produs din lista pentru a-l selecta 3. Adminul adauga noile campuri ce vor fi actualizate 4. Adminul apasa pe butonul de actualizare 5. Daca produsul exista, sistemul actualizeaza produsul si afiseaza noua lista |
| Scenariul alternativ | 4.a. Daca nu a fost selectat niciun produs, sistemul afiseaza fereastra de notificare  4.b. Daca produsul nu exista, sistemul afiseaza fereastra cu notificarea |

|  |  |
| --- | --- |
| Nume caz de utilizare | Sterge produs |
| Actor | Admin |
| Scenariul normal | 1. Adminul vede lista de produse existenta 2. Adminul apasa pe un produs din lista pentru a-l selecta 3. Adminul apasa pe butonul de stergere 4. Daca produsul exista, sistemul sterge produsul si afiseaza noua lista |
| Scenariul alternativ | 3.a. Daca adminul nu a selectat niciun produs, sistemul va afisa o fereastra de notificare  4.a. Daca produsul nu exista, sistemul afiseaza de notificare |

|  |  |
| --- | --- |
| Nume caz de utilizare | Login agent |
| Actor | Agent |
| Scenariul normal | 1. Agentul introduce datele de autentificare (username si parola) si apasa click pe butonul de login 2. Dupa verificarea credentialelor, sistemul afiseaza fereastra principala |
| Scenariul alternativ | 2.a. Daca nu sunt corecte credentialele, sistemul afiseaza fereastra de eroare |

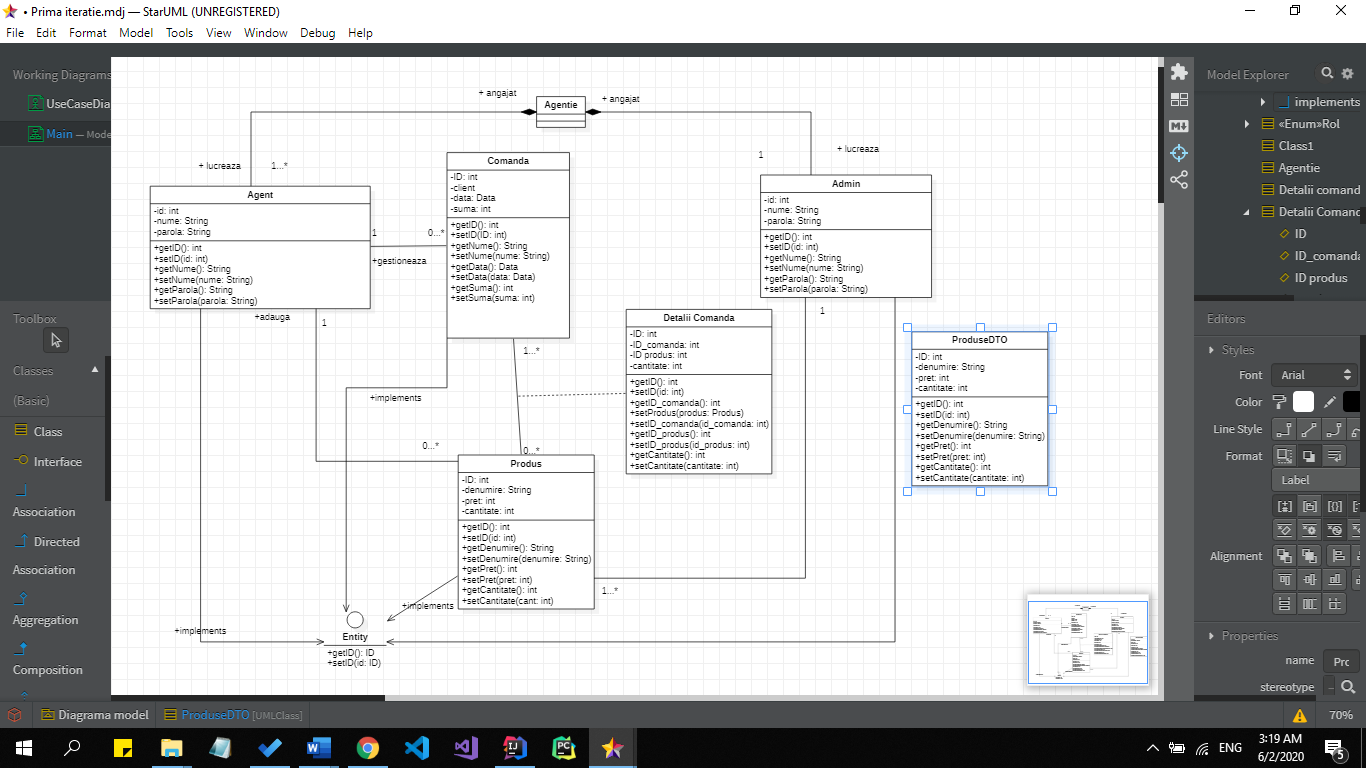
|  |  |
| --- | --- |
| Nume caz de utilizare | Vizualizare lista produse |
| Actor | Agent |
| Scenariul normal | 1. Agentul apasa pe butonul de afisare si ii afiseaza lista produselor disponibile in tabelul din stanga |
| Scenariul alternativ | 2.a. Daca nu exista niciun produs, lista va fi goala |

|  |  |
| --- | --- |
| Nume caz de utilizare | Adauga comanda |
| Actor | Agent |
| Scenariul normal | 1. Agentul adauga o comanda noua introducand ID-ul (numarul comenzii), data si numele clientului 2. Daca nu exista comanda, sistemul afiseaza noua lista de comenzi |
| Scenariul alternativ | 2.a. Daca deja exista aceasta comanda, sistemul afiseaza o fereastra de eroare |
| Nume caz de utilizare | Stergere comanda |
| Actor | Agent |
| Scenariul normal | 1. Agentul selecteaza o comanda 2. Apasa pe butonul de stergere 3. Daca comanda exista, sistemul afiseaza noua lista de comenzi |
| Scenariul alternativ | 2.a. Daca nu exista aceasta comanda, sistemul afiseaza o fereastra de eroare |

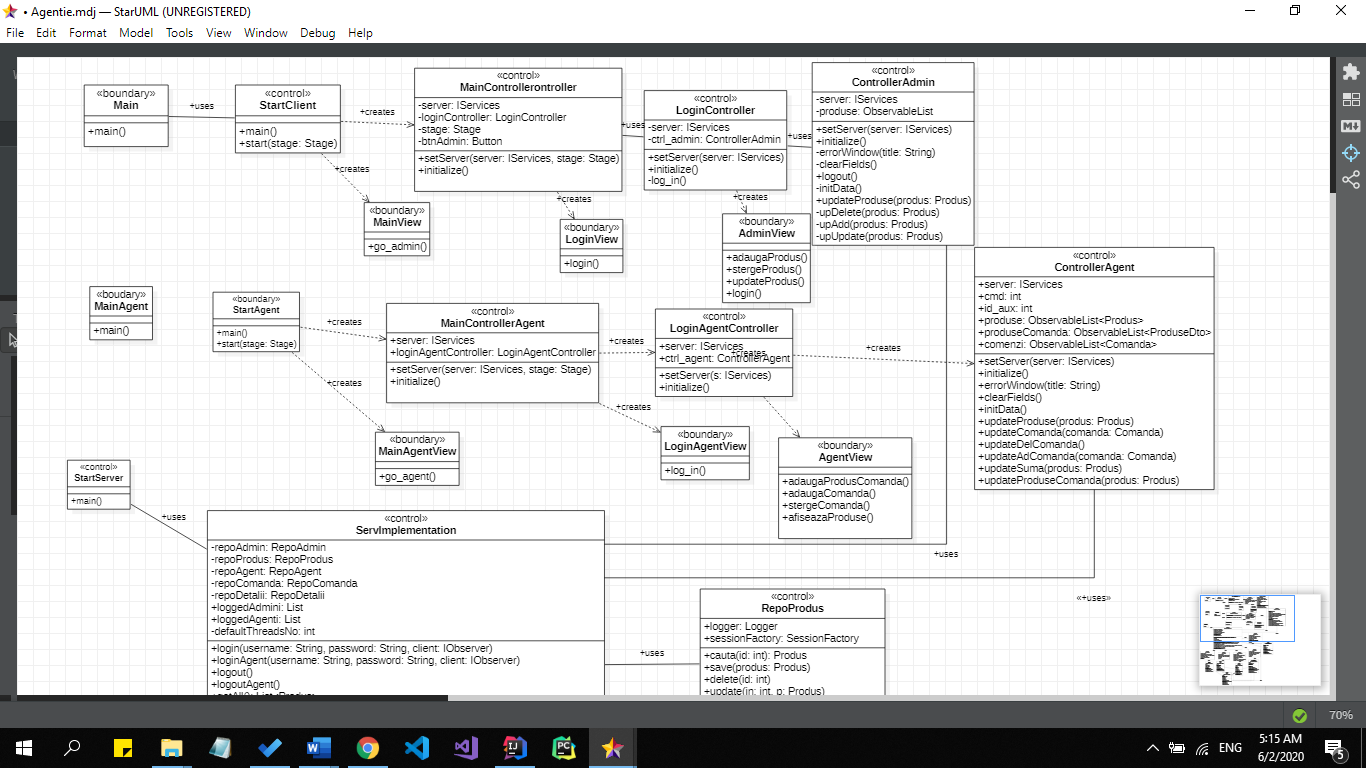
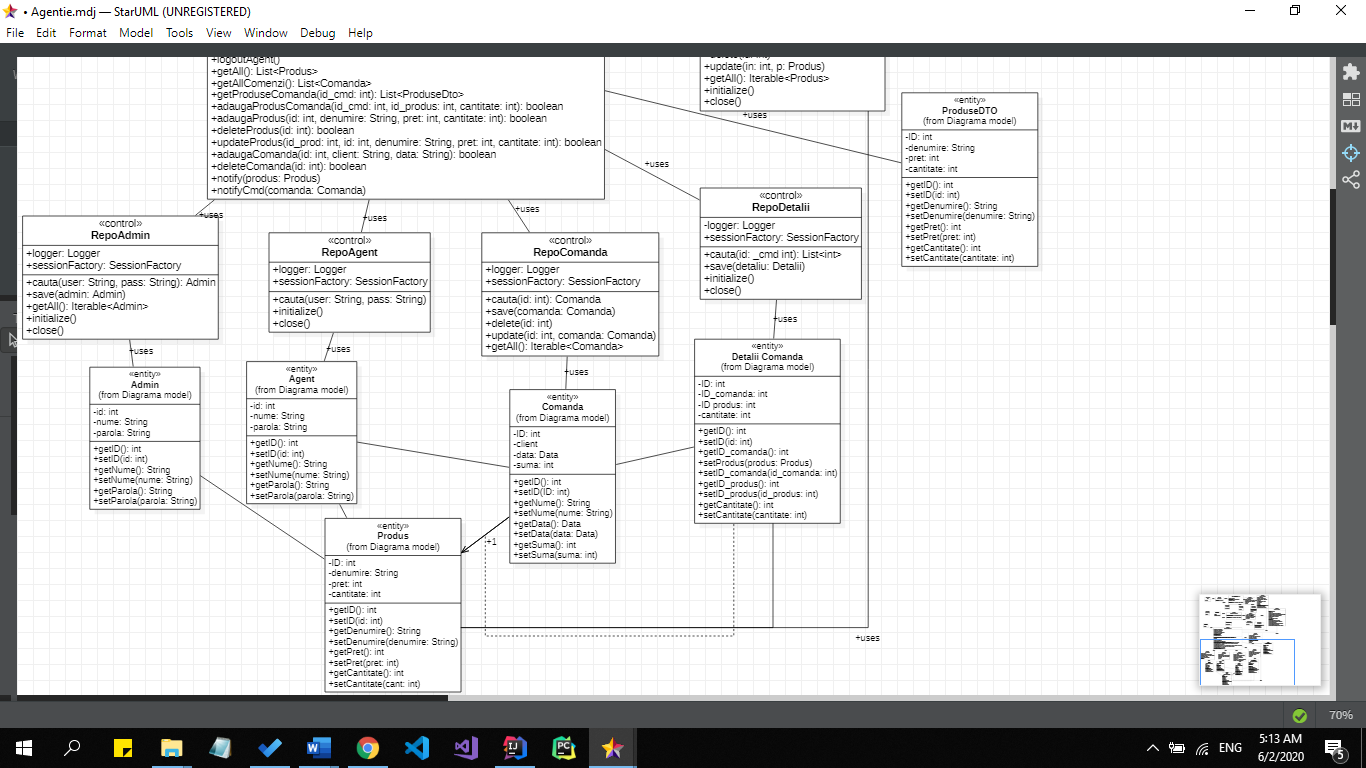
|  |  |
| --- | --- |
| Nume caz de utilizare | Vizualizare comanda |
| Actor | Agent |
| Scenariul normal | 1. Agentul poate vedea produsele dintr-o comanda, prin simpla apasare a butonului de “view produse” |

|  |  |
| --- | --- |
| Nume caz de utilizare | Adaugare produs in comanda |
| Actor | Agent |
| Scenariul normal | 1. Agentul scrie id-ul comenzii in care doreste sa adauge produs 2. Agentul introduce cantitatea dorita din produs 3. Agentul selecteaza un produs pe care doreste sa il plaseze intr-o comanda 4. Agentul apasa click pe Adaugare produs 5. Sistemul notifica agentul printr-o fereastra si updateaza lista produselor din comanda respectiva |
| Scenariul alternativ | 2.a. Daca agentul nu selecteaza nicio comanda, sistemul afiseaza o fereastra de eroare  3.a. Daca agentul nu selecteaza niciun produs, sistemul afiseaza o fereastra de eroare |

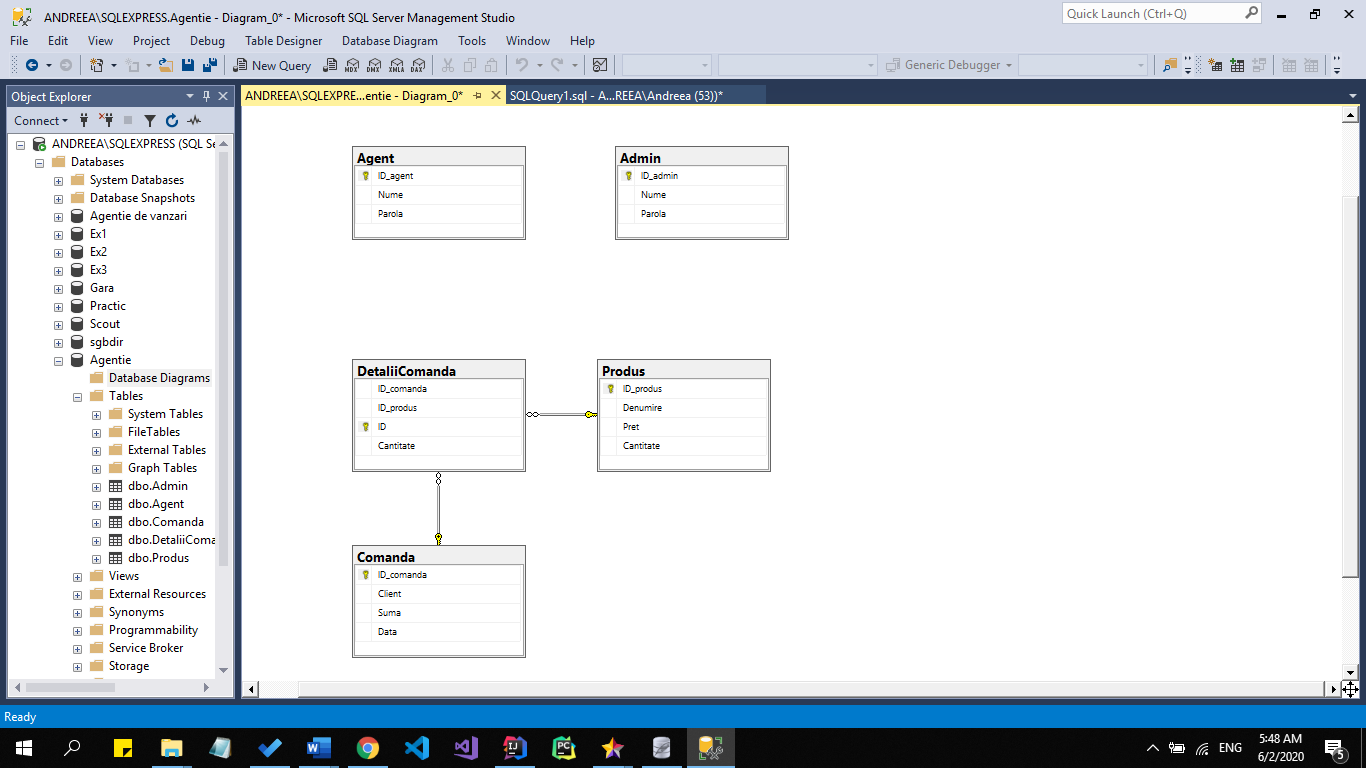
1. Modelul conceptual
2. Diagrama de clase a modelului

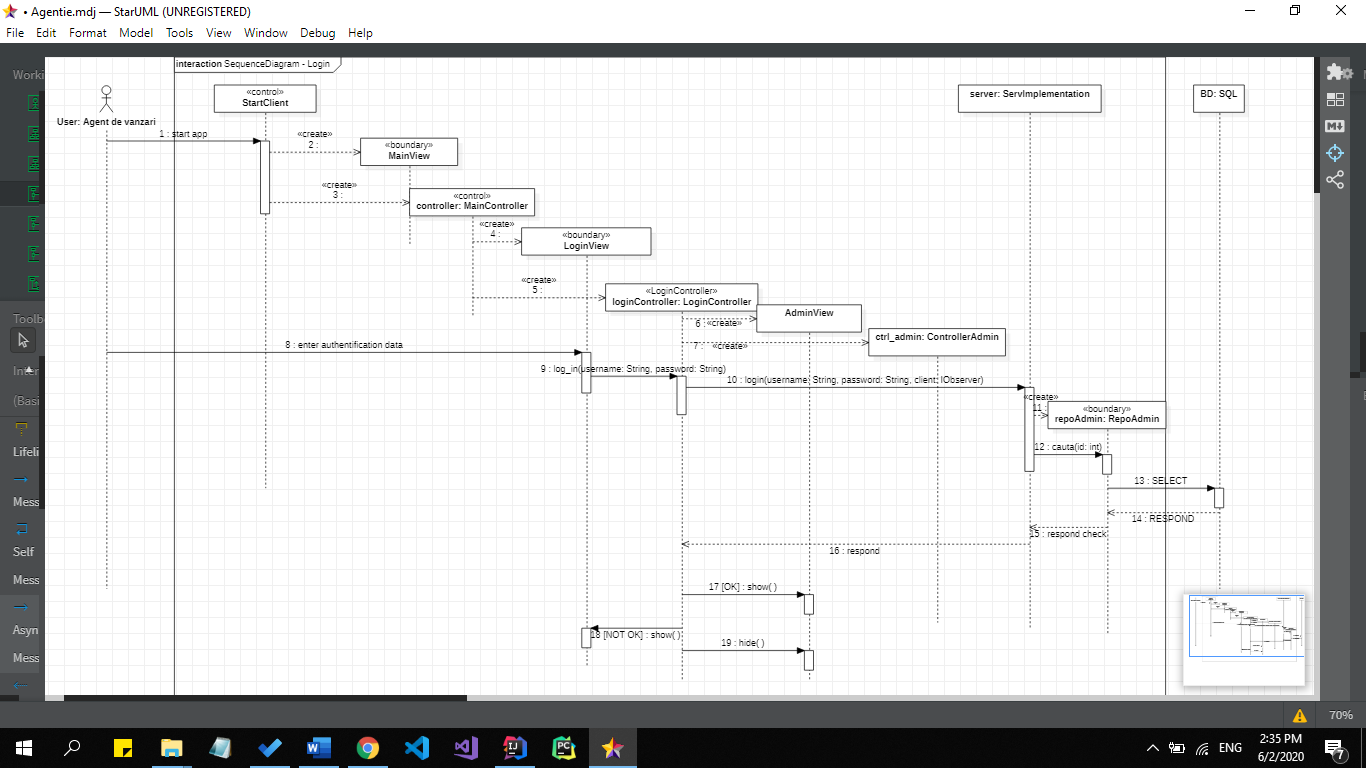
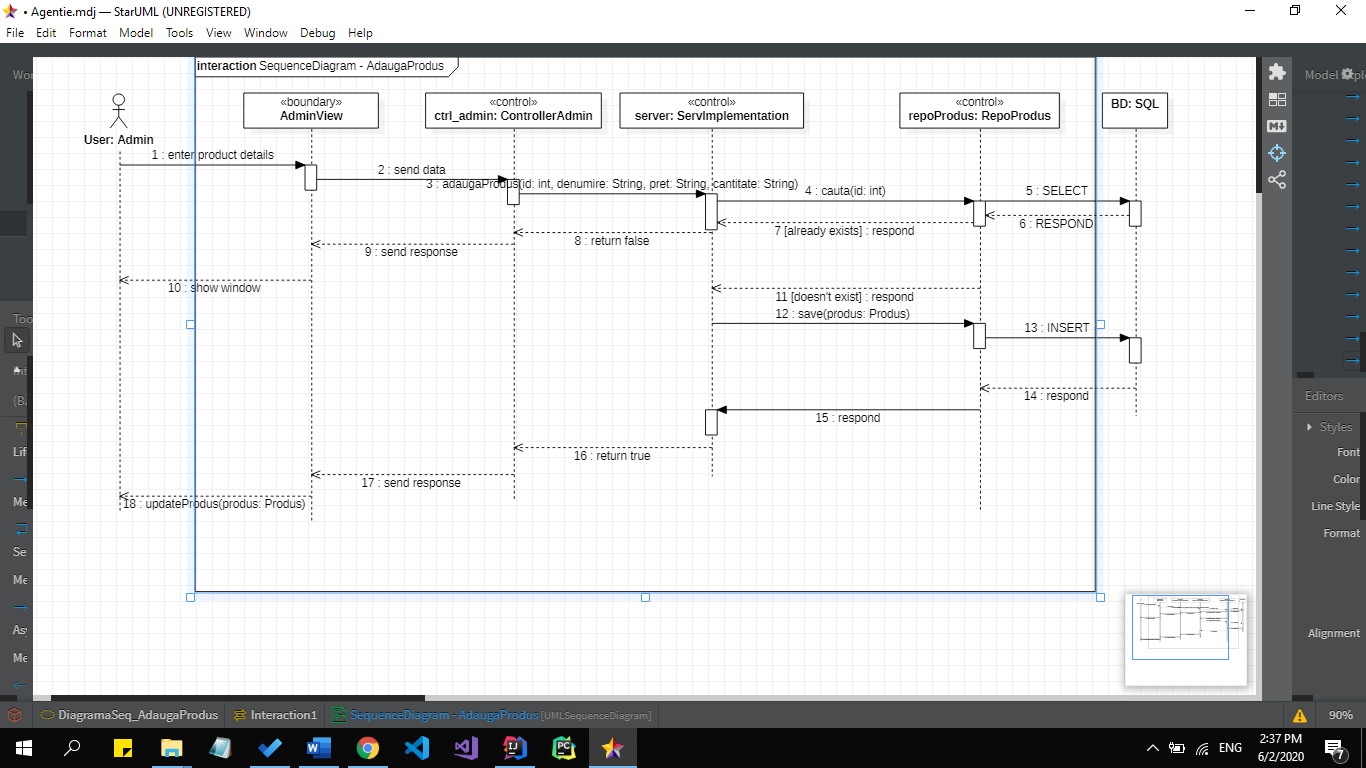


1. Diagrama de clase finisate

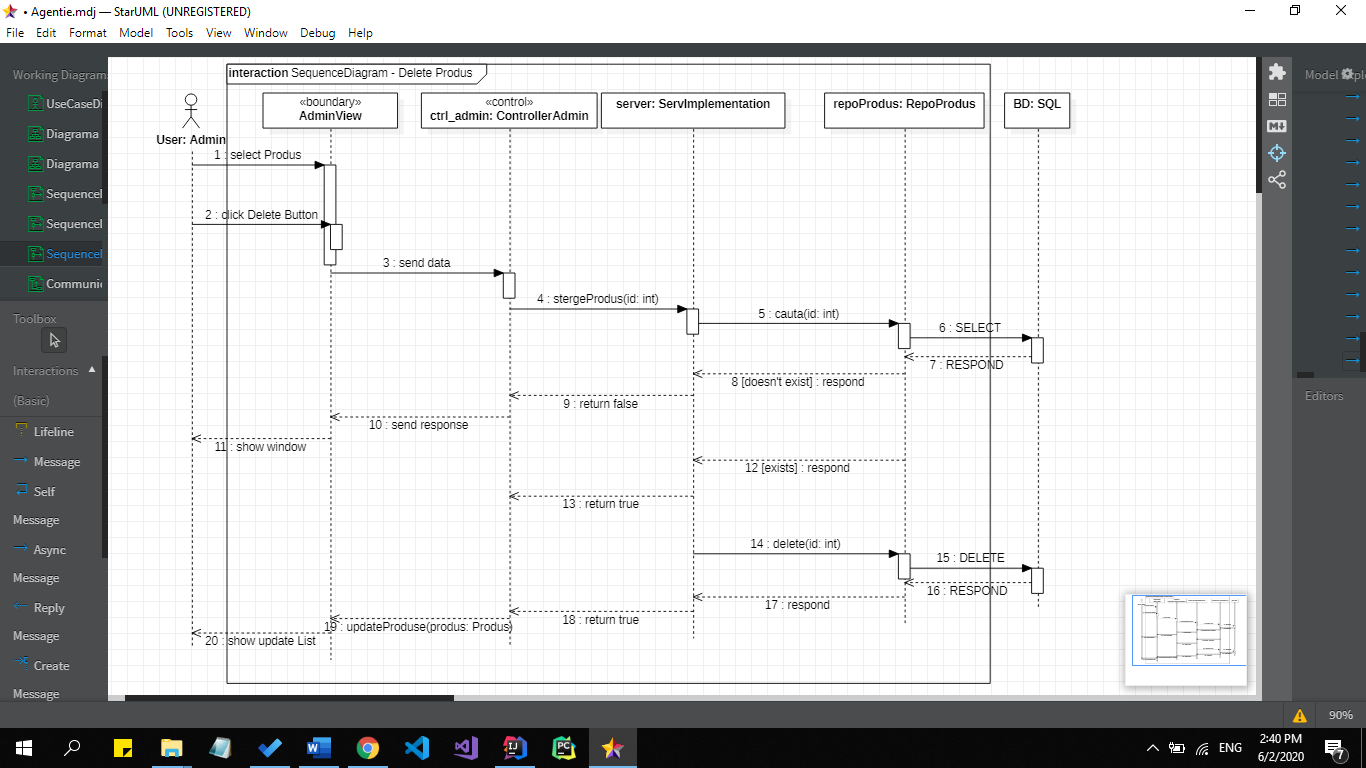


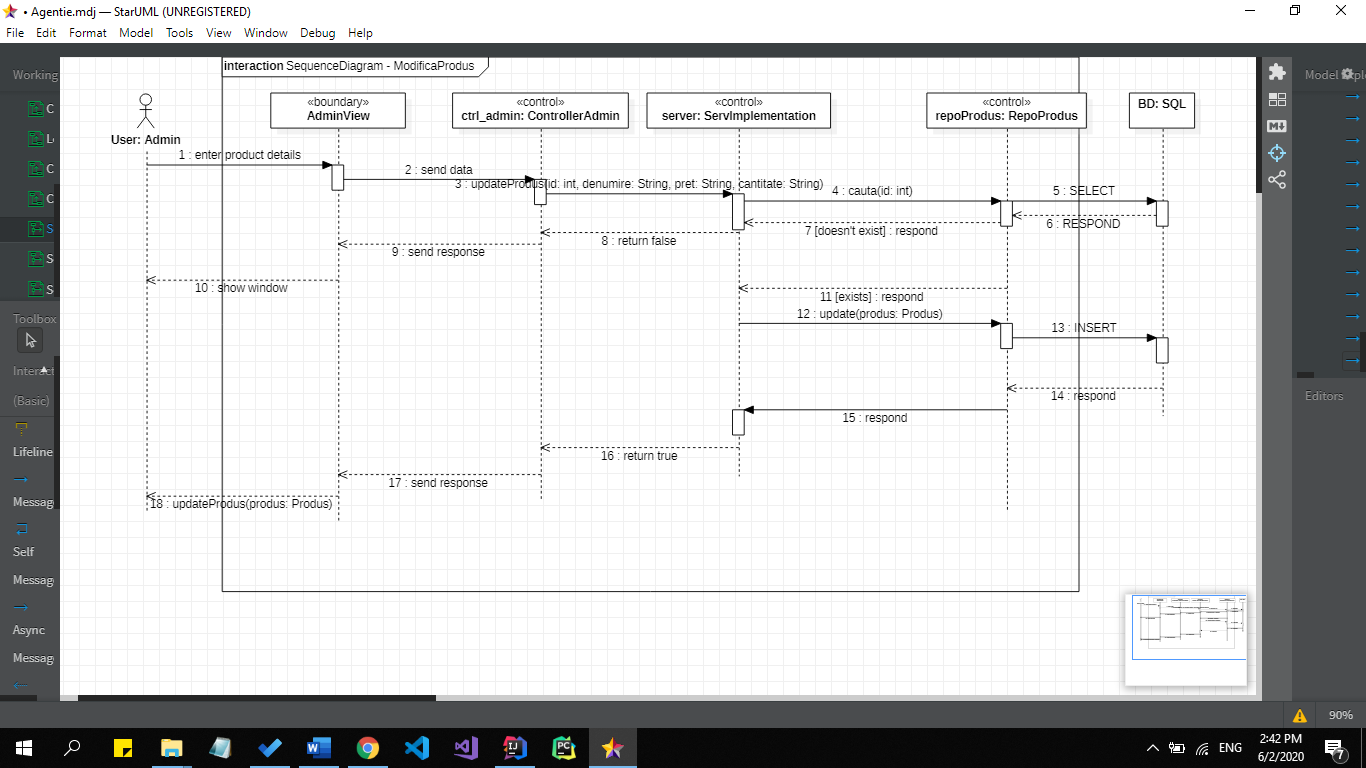
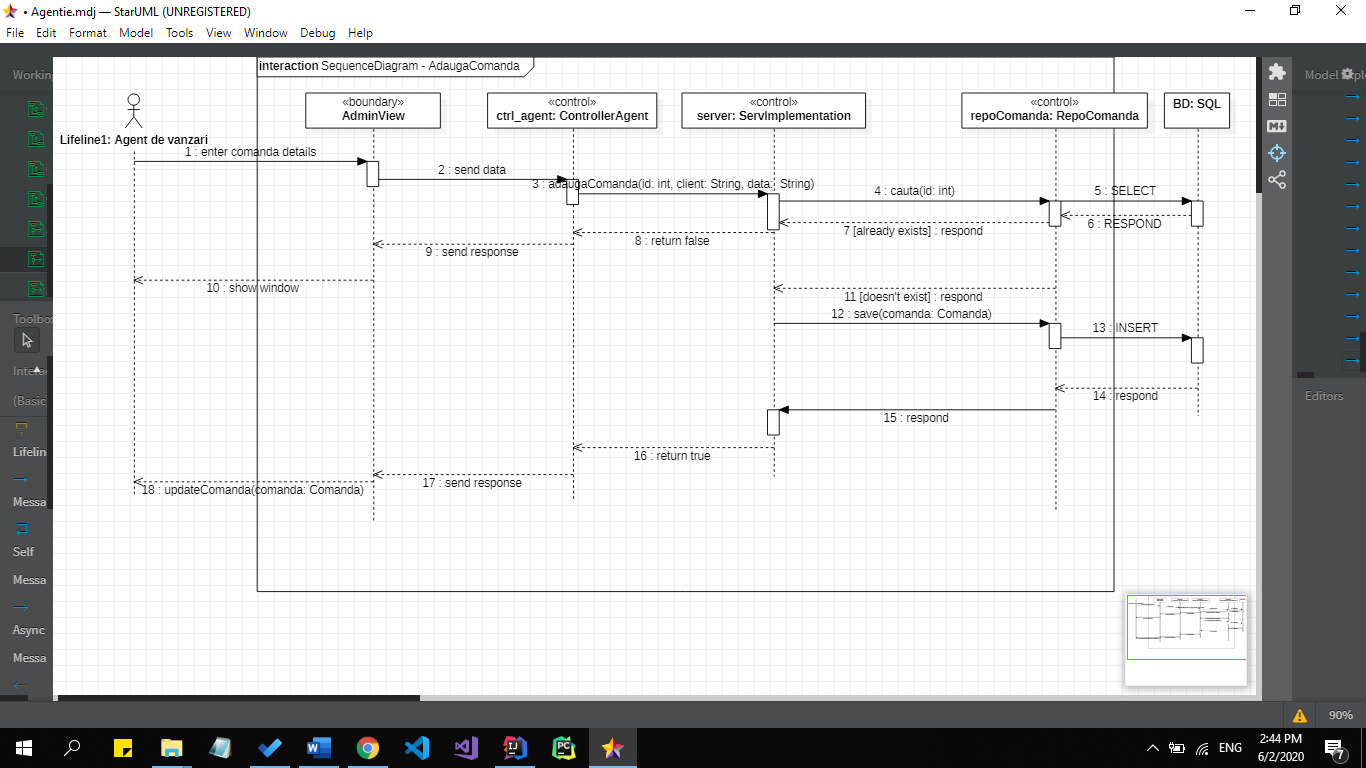
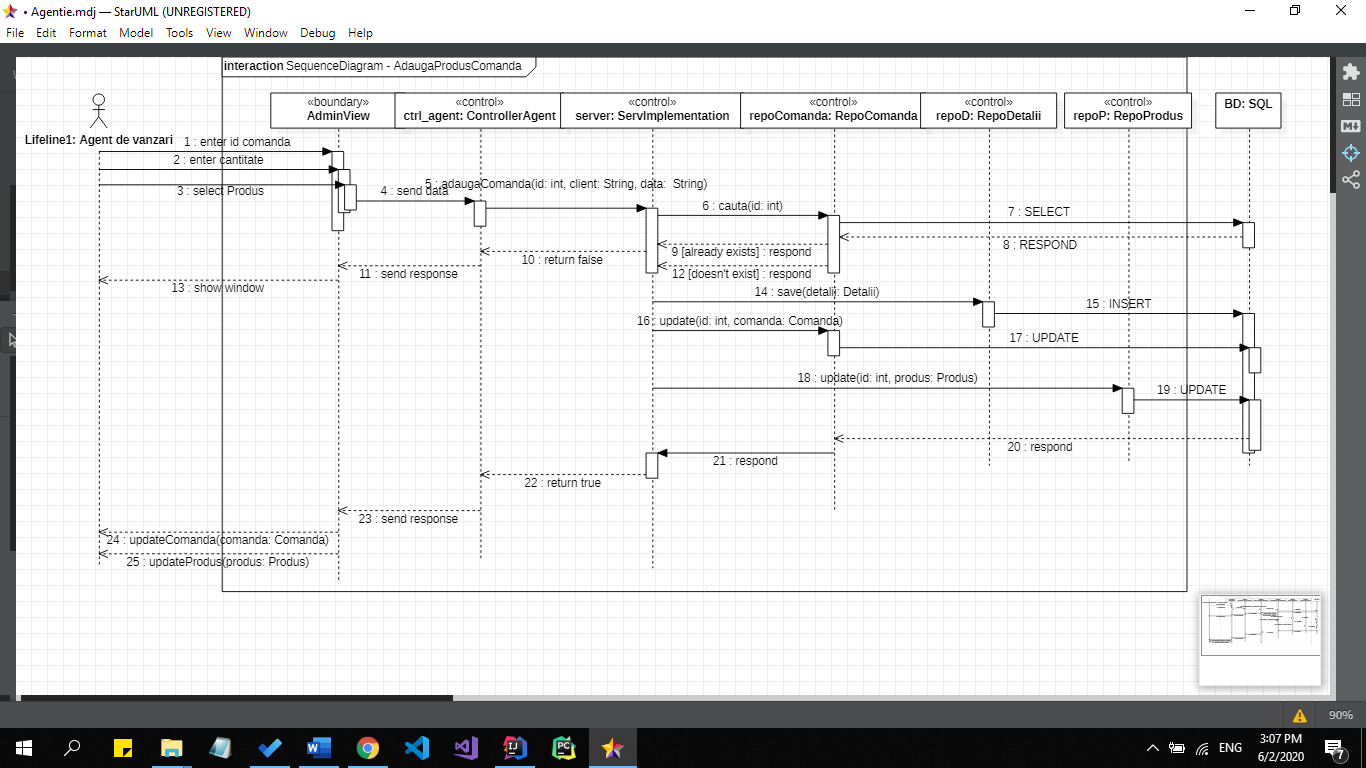
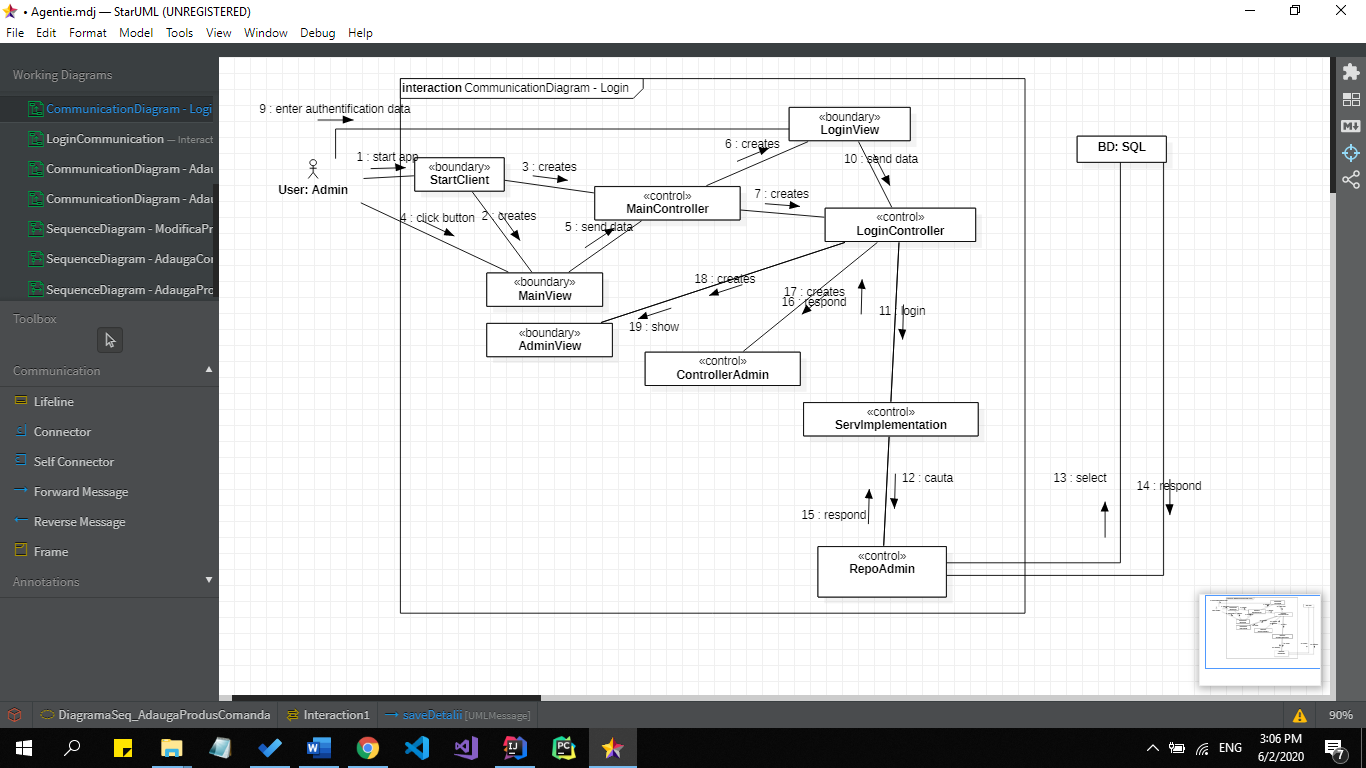
1. Diagrama BD

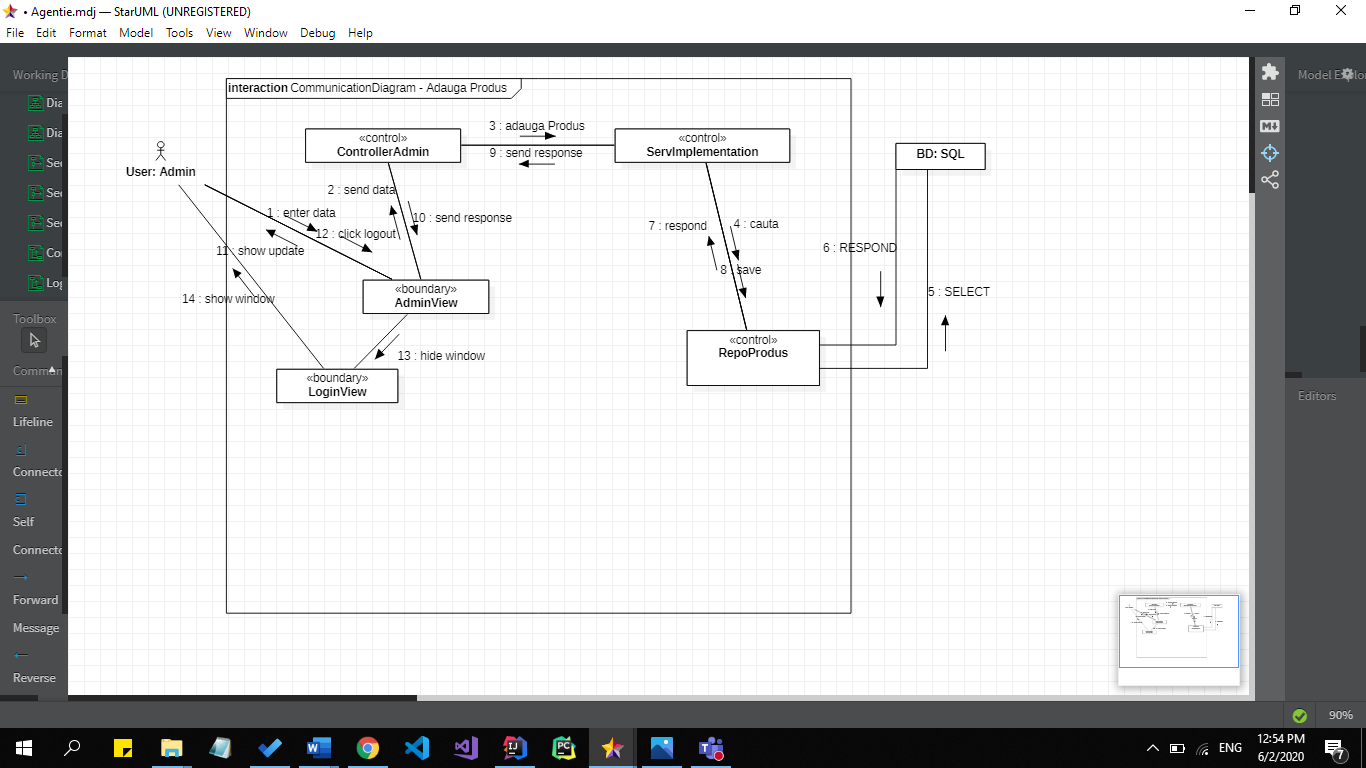


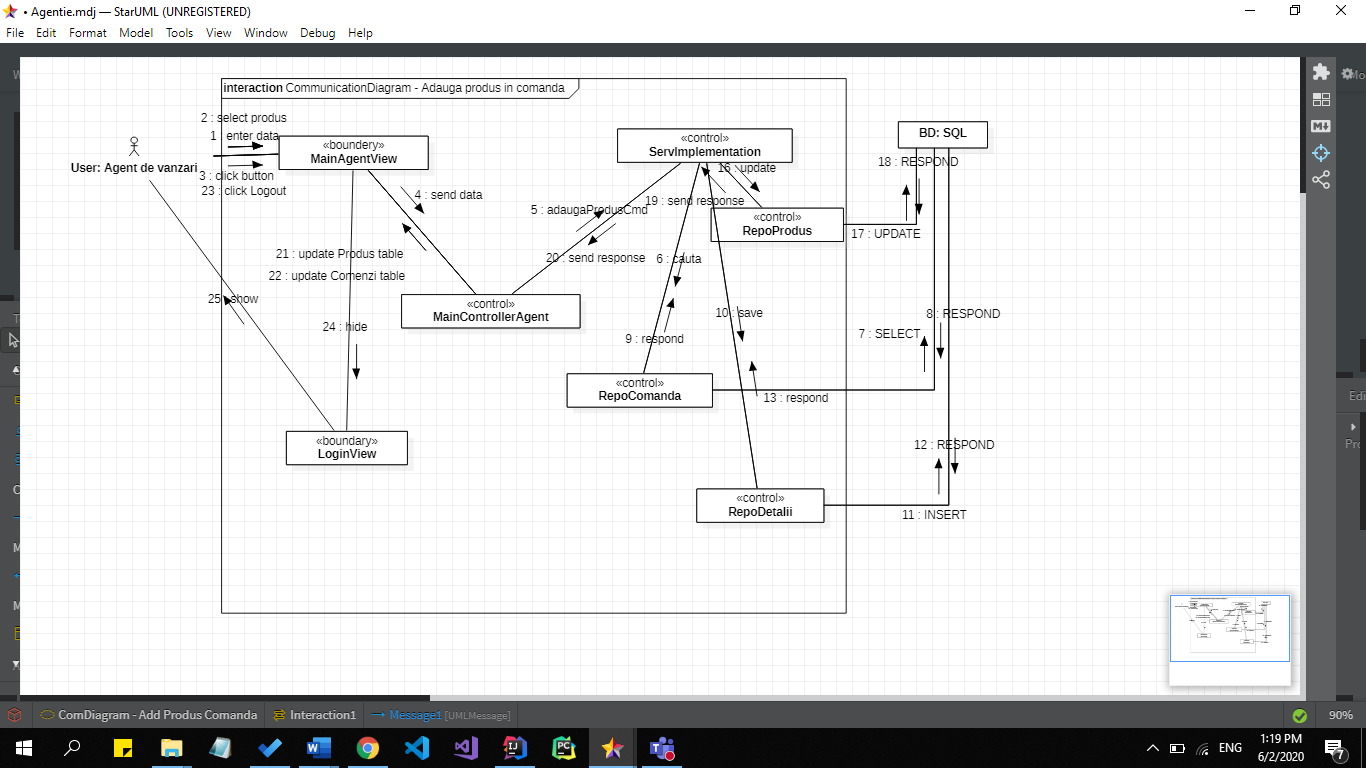
1. Model dinamic:
2. Diagrama de secventa – Login (Admin) – asemanator cu Agent – alta entitate
3. Diagrama de secventa – Adauga Produs

1. Diagrama de secventa: Sterge Produs



1. Diagrama de secventa: Modifica produs
2. Diagrama de secventa: Adauga Comanda
3. Diagrama de secventa: Adauga Produs in Comanda
4. Diagrama de comunicare: Login
5. Diagrama de comunicare: Adaugare Produs



1. Diagrama de comunicare: Adauga Produs in Comanda
2. Documentatie:

* Functionalitati:

Admin:

* Gestioneaza stocul de produse, putand sa adauge, sa stearga si sa apdateaza un produs.
* Acesta poate intra in aplicatie introducand credentialele, username si parola.

Agent:

* Pentru a intra in aplicatie se logheaza – introduce credentiaele: username si parola.
* Acesta creeaza o noua comanda, utilizand un numar de facturare nefolosit, numele clientului si data – in format “yyyy-mm-dd”, comanda care la inceput nu continue niciun produs si are suma 0. Acesta poate sterge o comanda selectand-o din tabel si apasand pe butonul de stergere.
* Pentru a adauga produse in comanda, merge pe partea de produse si selecteaza produsele, introducand si numarul comenzii dorite si cantitatea de produs pe care o doreste. Aceste produse se pot observa apasand pe fiecare comanda si apoi pe butonul “Vezi produse”.
* Etapele proiectarii:

**Analizare cerintelor:**

* + Am indentificat cerintele impuse, functionalitatiile (autentificarea, un agent de vanzari sa adauge sau sa stearga o comanda) si constrângerile (adminul trebuie să fie autentificat pentru a adaugare/sterge/modifica un produs, la fel si agentul trebuie sa fie autentificat pentru a putea gestiona comenzile).

**Analiza:**

* + Cerințe funcționale: Aplicatia trebuie sa-i permita agentului sa adauge o comanda si sa o stearga. Administratorul poate sa adauge/modifice/sterge un produs.
  + Cerințe non-funcționale: Aplicația trebuie să fie scrisă în Java, să fie ușor de utilizat de oricine – e natural de urmarit procesul.

**Proiectarea sistemului:**

* + Am folosit StarUML pentru a realiza diagrame pentru a planifica dezvoltarea aplicației. Tipurile de diagrame pe care le-am folosit pentru a-mi facilita munca sunt: diagramă de utilizare, diagramă de secvență, diagramă de comunicare, diagram de clase.
  + Pentru diagrama bazei de date am folosit SQL Server.

**Implementarea:**

* + Am inceput prin a-mi crea entitatile, apoi am facut partea de networking. Dupa care a inceput crearea interfetelor grafice si implementarea functionalitatilor. Pe parcurs am mai modificat entitatile initiale, adaugand cate un camp sau creand entitati noi sau DTO-uri.

**Testare:**

* + Am testat personal codul – prin exemple si am rugat si un utilizator necunoscator al aplicatiei sa incerce.
* Tehnologii folosite:

**Baza de date**: Am folosit SQLite ca baza de date, deoarece are o interfata user-friendly si e usor de gestionat.

**Mediul de dezvoltare**: Am ales sa folosim IntelliJ IDEA Ultimate – exista cateva avantaje vis-à-vis de versiunea Community. Usurinta de a scrie intr-un editor cunoscut isi spune cuvantul cand scrii o aplicatie.

**Limbaj de programare**: Limbajul de programare ales este Java pentru cateva motive simple. Are o comunitate mare de utilizatori care sprijina dezvoltarea lui, un mare ajutor la nelamurirele noastre pe care le-am intampinat pe parcursul proiectului. Este un limbaj nativ, sugerandusi usor gestiunea codului. Functioneaza pe platformele de operare. In plus, este un limbaj potrivit pentru dezvoltarea de software. Partea cea mai buna este garbage collector si usurinta de a nu trebui sa te ocupi ca programator de pointeri si gestiunea memoriei.

**ORM**: Object / Relational Mapping (ORM) este o tehnica de programare ce face posibila accesarea și manipularea obiectelor fara ca programatorii sa fie interesati de sursa de date de unde provin aceste obiecte. Aceasta tehnica a aparut din nevoia de a depasi diferentele de paradigma dintre modelul orientat pe obiecte si modelul relational. Am folosit Hibernate, pentru usurinta sa, fara a implementa manual baza de date.

**Proiectarea diagramelor**: Am folosit StarUML la proiectarea diagramelor pentru usurinta si multitudinea de avantaje ale programului.

**GUI**: Am folosit JavaFX pentru usurinta de folosire si combinarea cu CSS-ul pentru un design mai placut (pe care o voi implementa in viitor). Scene Builder e foarte comod de implementat interfete, acestea aratand foarte bine.

**Structura proiectului:** Proiectul este organizat pe niveluri: Model, Persistence, Services, Server, Client.

* Model: contine declaratii de entitati pentru controlul bazei de date.
* Persistence: folosim repository care permit interogarea bazei de date folosind Session Factory de la Hibernate.
* Services: contine toate interfetele folosite in applicate si clasa de exceptii.
* Server: contine toate implementarile specificate pana acum.
* Client: contine toate interfetele vizibile si controller-urile asociate, care comunica informatii la server.

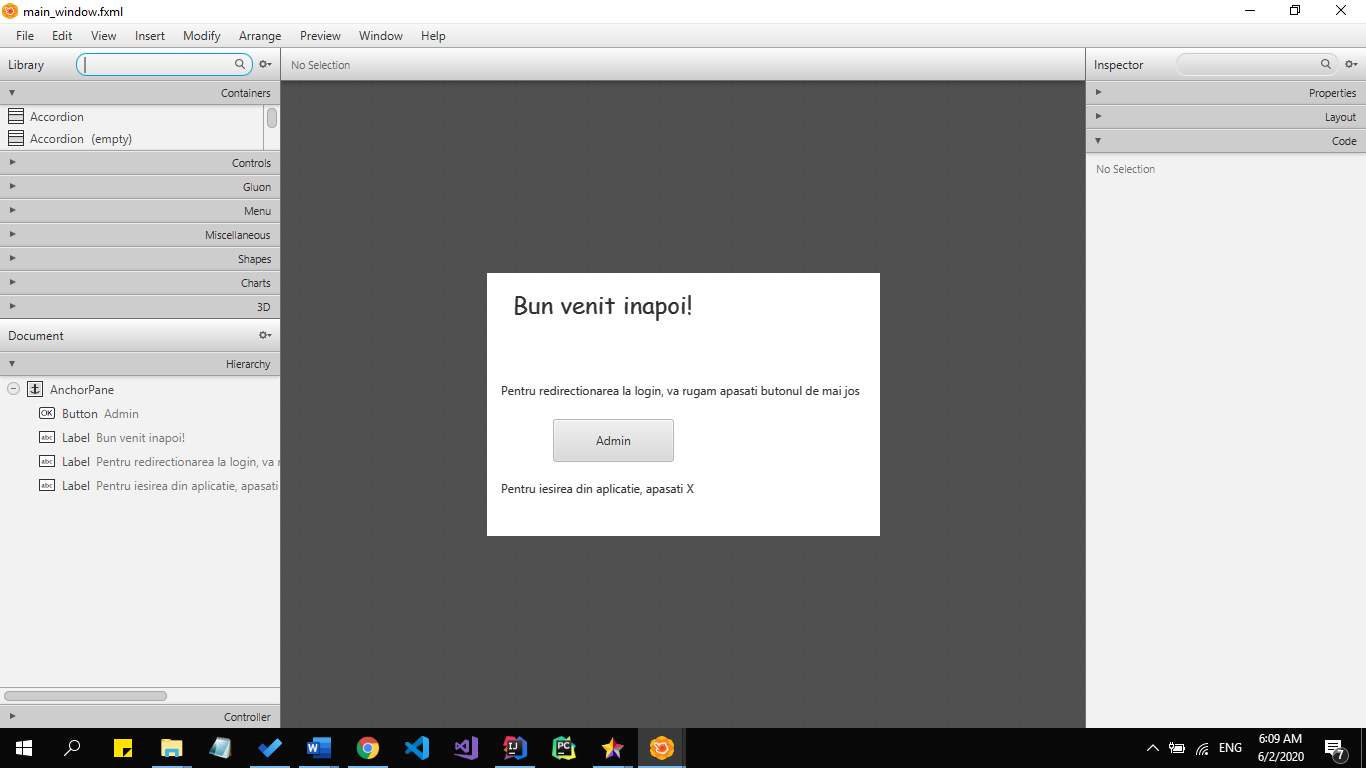
1. Help

Figure 1. Fereastra Admin

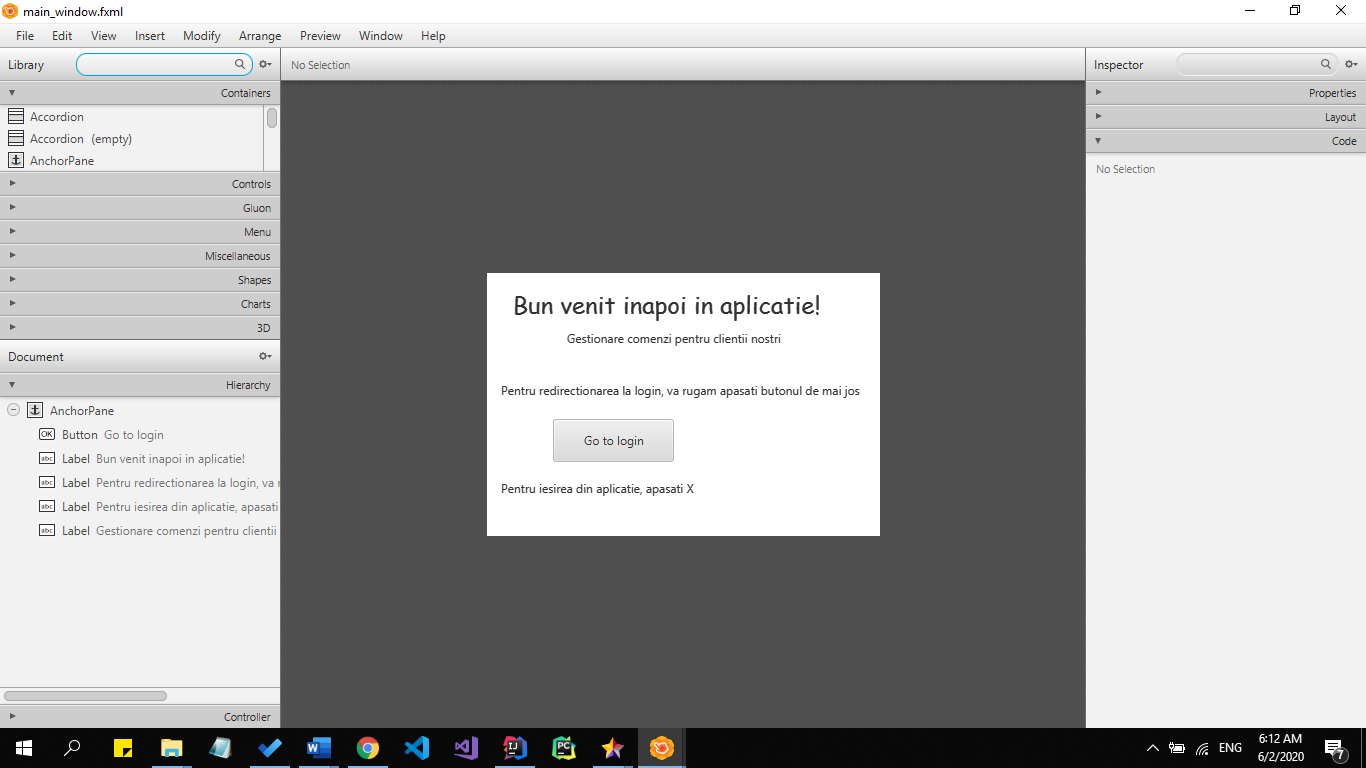


Figure 2. Fereastra agent

In functie de rolul dumneavoastra,

va rugam sa alegeti una dintre cele doua ferestre:

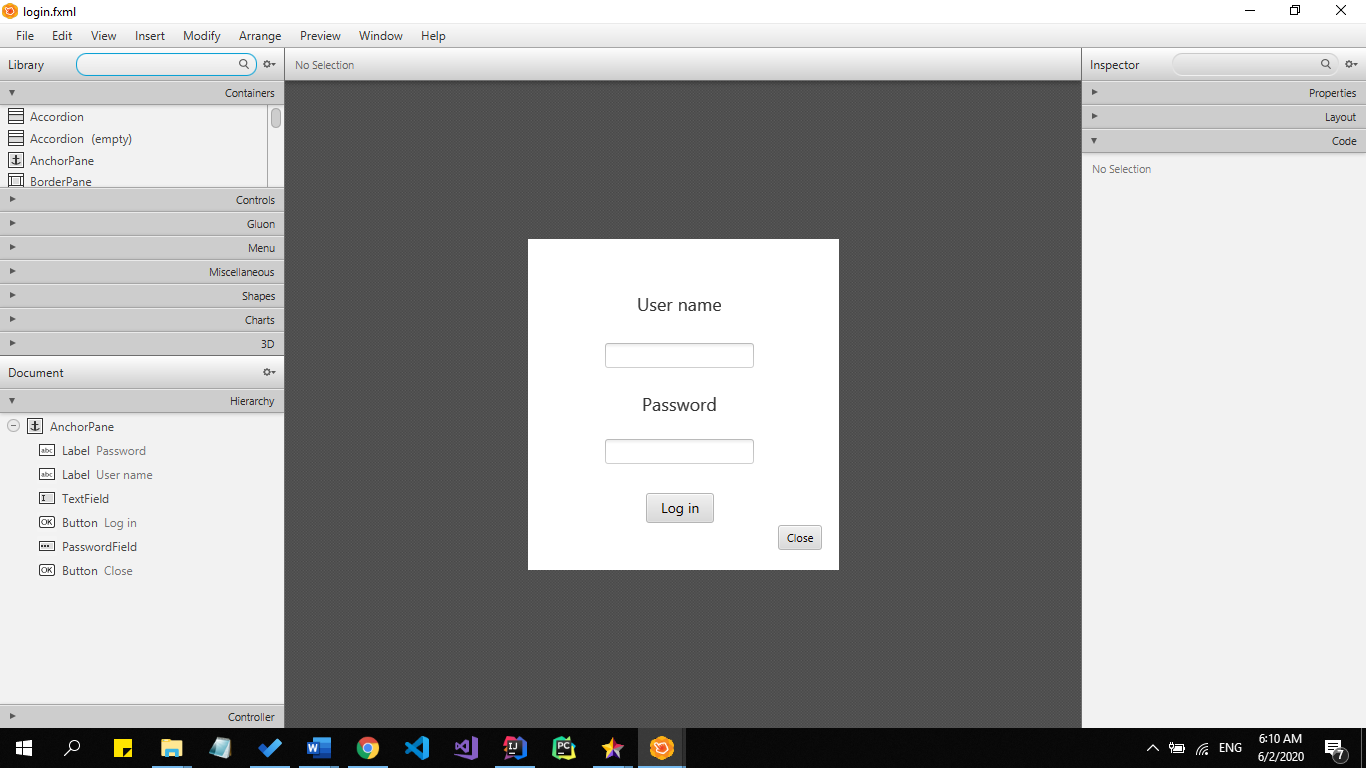
Daca doriti sa parasiti aplicatia in acest punct, este suficient sa apasati pe X.

Daca doriti sa continuati, apasati pe butonul indicat.

Acesta va va redirectiona spre o noua fereastra unde trebuie sa va introduceti credentialele.

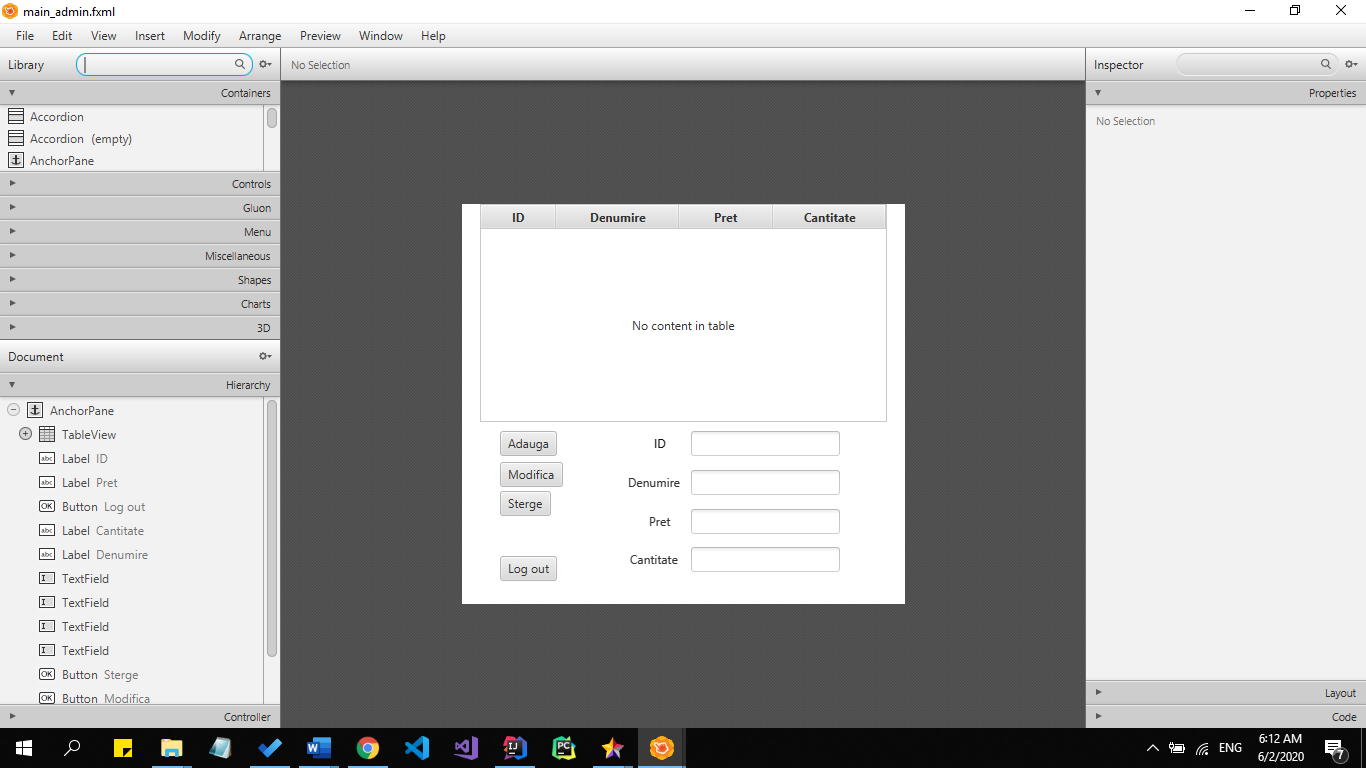
In cazul in care exista o eroare de typing, va aparea o fereastra care sa indice aceasta eroare pe care va trebui sa o inchideti si sa mai incercati o data.

In cazul in care v-ati razgandit si nu mai doriti sa utilizati aplicatia, doar apasati pe butonul de Close, care va va redirectiona la fereastra anterioara.



Daca sunteti admin in aceasta aplicatie, va fi afisata urmatoarea fereastra, unde veti putea gestiona stocul de produse.

Tabelul unde se vor afisa produsele existente



Butonul de adauga

Butonul de modifica

Butonul de sterge

Text box-urile unde se introduc detaliile legate de produs

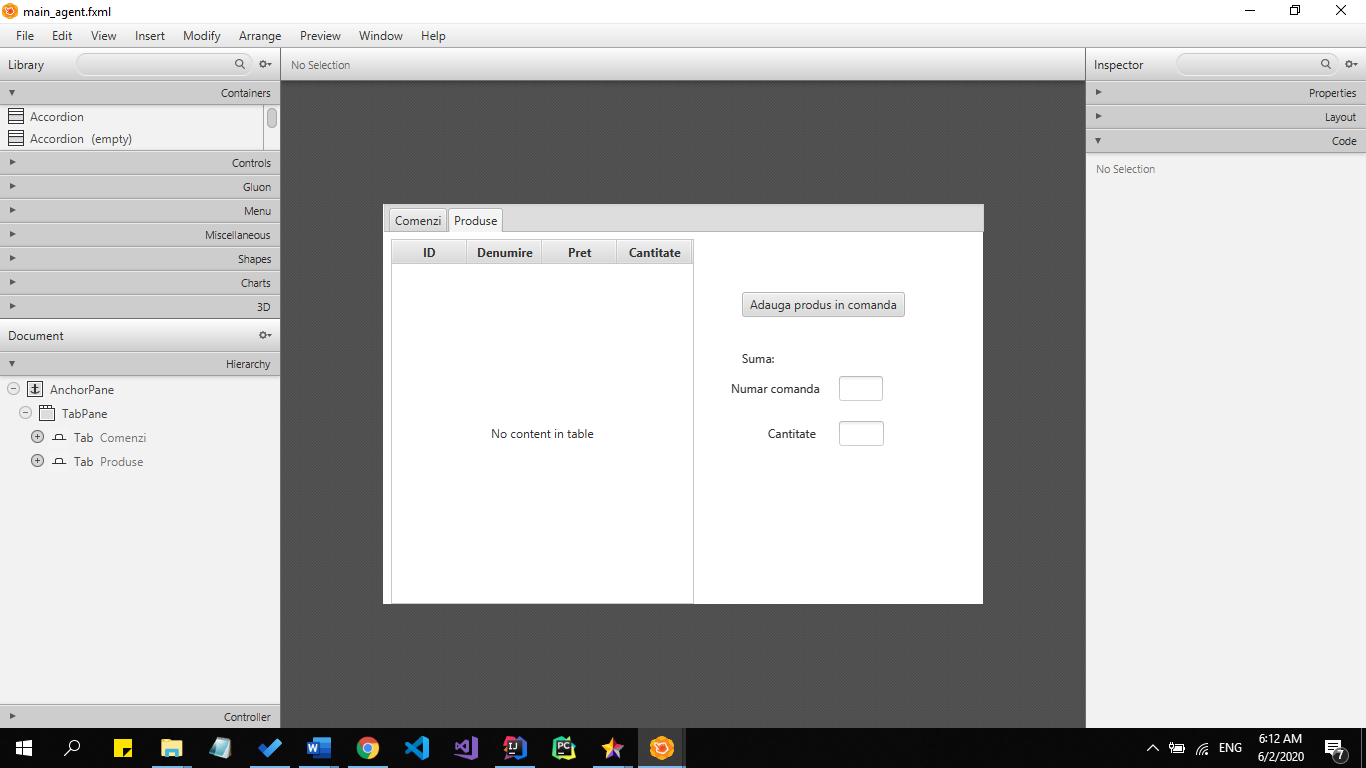
Butonul de logout

In cazul in care sunteti un agent de vanzare, se va afisa urmatoarea fereastra, care este impartita in 2 parti: una pentru comanda, iar cea de-a doua pentru produse.

Vizionare comenzi

Vizionare produse

Butonul de adaugare produs in comanda



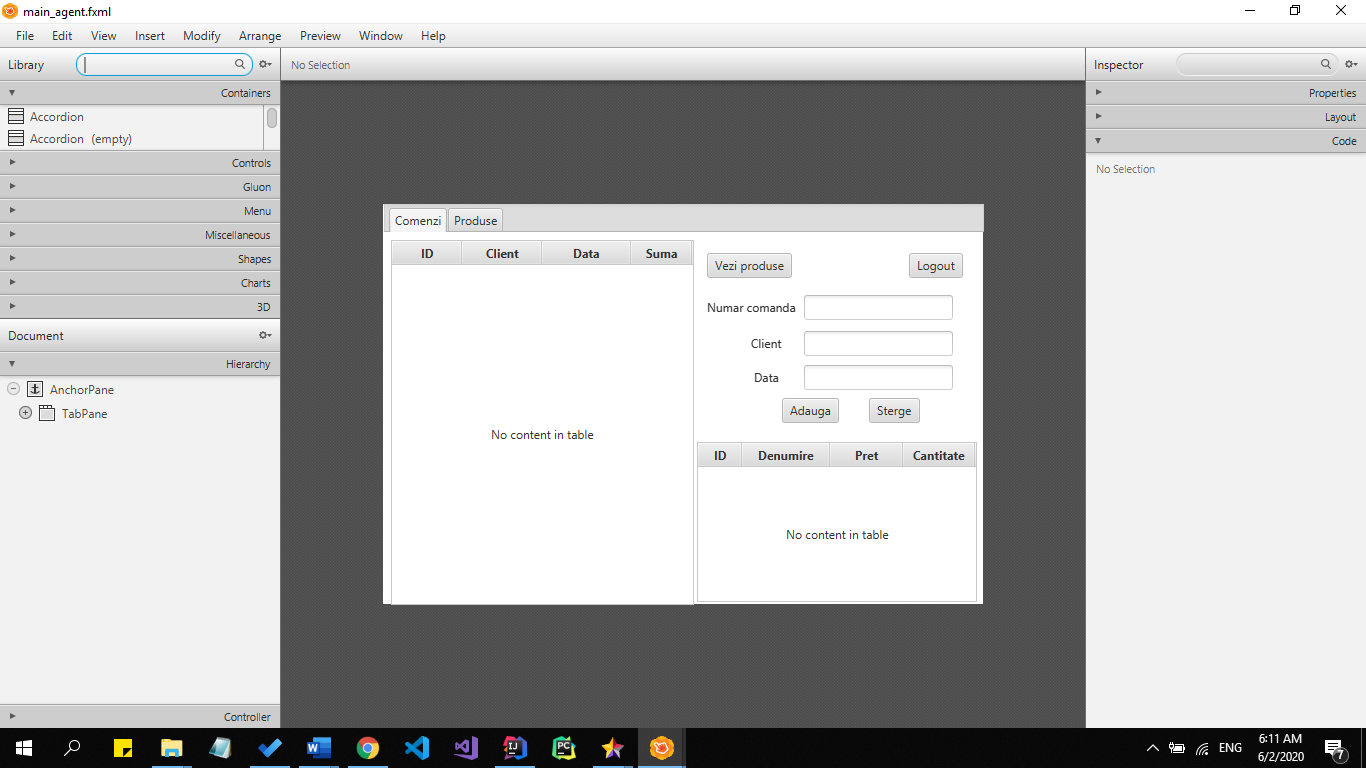
Text box pentru comanda in care se doreste adaugarea produsului

Tabela Produs

Text box-ul pentru cantitatea de produs dorita

Atentie! Pentru selectarea unui produs, respective comanda – pentru adaugare/ stergere – se selecteaza pur si simplu randul droit.

Buton pentru afisarea produselor dintr-o comanda



Buton pentru logout

Tabela comenzi

Text box-uri pentru informatiile unei comenzi pe care dorim sa o cream

Tabela produse dintr-o comanda selectata

Buton pentru stergerea unei comenzi

Buton pentru adaugarea unei comenzi

Biblografie si referinte:

1. <https://simaioan.wordpress.com/>
2. <http://www.cs.ubbcluj.ro/~vladi/Teaching/Didactic/ISS%202019-2020/info%20romana/seminar/>
3. <http://www.cs.ubbcluj.ro/~grigo/mpp/Curs/>
4. <http://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/ecb/ecb.htm>
5. <https://www.youtube.com/watch?v=UjmcusHEvIA>
6. <https://www.youtube.com/watch?v=pCK6prSq8aw>
7. <https://www.tutorialspoint.com/hibernate/hibernate_examples.htm>