**Inginerie Software**

**Documentatie proiect**

* **Aplicatie de catering** –

Giesswein Alexia

Grupa: 30231

1. **Prezentarea proiectului**

Aplicatia implementata reprezinta un sistem care simuleaza o aplicatie de catering destul de realista, utilizatorul avnd posibilitatea sa comande mancare ca in viata de zi cu zi. Aplicatia are mai multe functionalitati, precum: utilizatorul (clientul) se poate inregistra si apoi loga in aplicatie, pe baza unui username si al unei parole. Dupa ce clientul se logheaza in aplicatie, acesta are optiunea de a comanda mancare, dupa cele trei categorii: fel principal, bautura si desert, se calculeaza pretul total si poate sa trimita comanda, iar la final se poate deloga.

Adminul se poate loga cu contul de admin si se deschide o interfata diferita fata de cea a clientului. Adminul se conecteaza cu contul sau si poate vizualiza toate produsele din aplicatie, poate adauga produse noi, poate modifica un produs existent si poate sterge produse.

1. **Descriere**

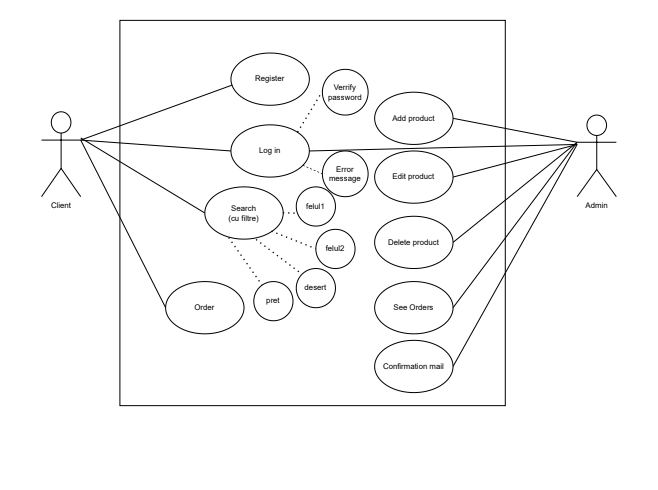
Limbajul folosit pentru implementarea proiectului este java si am folosit framework-uri si dependintele: SpringBoot, Spring Web, Jba, Spring Security, MySQL Driver.

Clasele implementate:

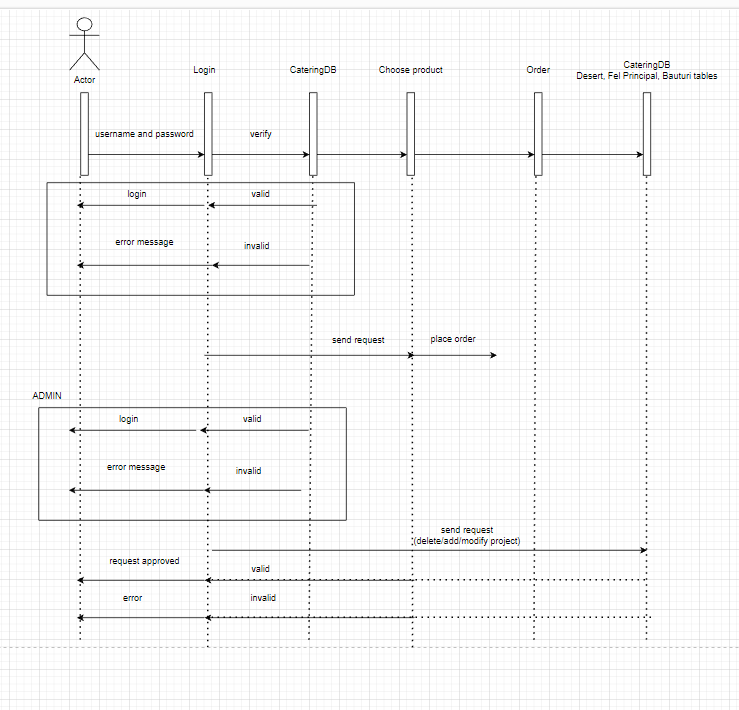
* User – clasa pentru utilizator, care contine prenumele, numele, emailul, parola si rolul (daca e client sau admin)
* Role – care reprezinta rolul utilizatorului (client sau admin)
* Meniu – clasa pentru meniul pe care poate sa l comande un utilizator
* Fel principal – clasa care contine felul principal care poate fi comandat de utilizator, avand nume, pret si descriere.
* Desert – clasa care contine desertul pe care poate sa-l comande un utilizator, avand nume, pret si descriere.
* Bauturi – clasa care contine bauturile care pot fi comandate de utilizator, avand nume, pret, descriere si daca contine alcool.
* SecurityConfiguration – care verifica datele introduse la logare si inregistrare si cripteaza parola utilizatorului
* BauturiServiceImpl – care contine metodele CRUD de la tabelul bauturi (getAllDrinks, getDrink, save, deleteDrink)
* DesertServiceImpl – care contine metodele CRUD de la tabelul desert (getAllDeserts, getDesert, save, deleteDesert)
* FelPrincipalServiceImpl – care contine metodele CRUD de la tabelul fel principal (getAllFeluriPrincipale, getOne, save, deleteFelPrincipal)
* MeniuServiceImpl – care contine metodele CRUD de la tabelul meniu
* UserServiceImpl – care contine metodele CRUD de la tabelul user (getAllUsers, getUser, update, deleteUser)
* BauturiDto – care creeaza un obiect nou de tipul bauturi pe baza datelor primite
* DesertDto – care creeaza un obiect nou de tipul desert pe baza datelor primite
* FelPrincipalDto – care creeaza un obiect nou de tipul fel principal pe baza datelor primite
* UserRegistrationDto – care creeaza un obiect nou de tipul user pe baza datelor primite
* AdminController – care contine metodele de handle pentru requesturile de la pagina adminului
* DesertController – care contine metodele de handle pentru requesturile de la desert
* DrinksController – care contine metodele de handle pentru requesturile de la bauturi
* FelPrincipalController – care contine metodele de handle pentru requesturile de la felul principal
* MainController – care contine metodele de handle pentru requesturile generale
* UserClientController – care contine metodele de handle pentru requesturile de la pagina clientului
* UserRegistrationController – care contine metodele de handle pentru requesturile de creare a unui nou user

1. **Diagrama de use case**

Diagrama de use case-uri reprezinta functionalitatile aplicatiei de catering, care are 2 tipuri de utilizatiori posibili: admin si clienti.

****

1. **Diagrama de secvente**



1. **Diagrama bazei de date cu tabele generata in MySQL**

**Diagram

Description automatically generated**

1. **Design pattern**

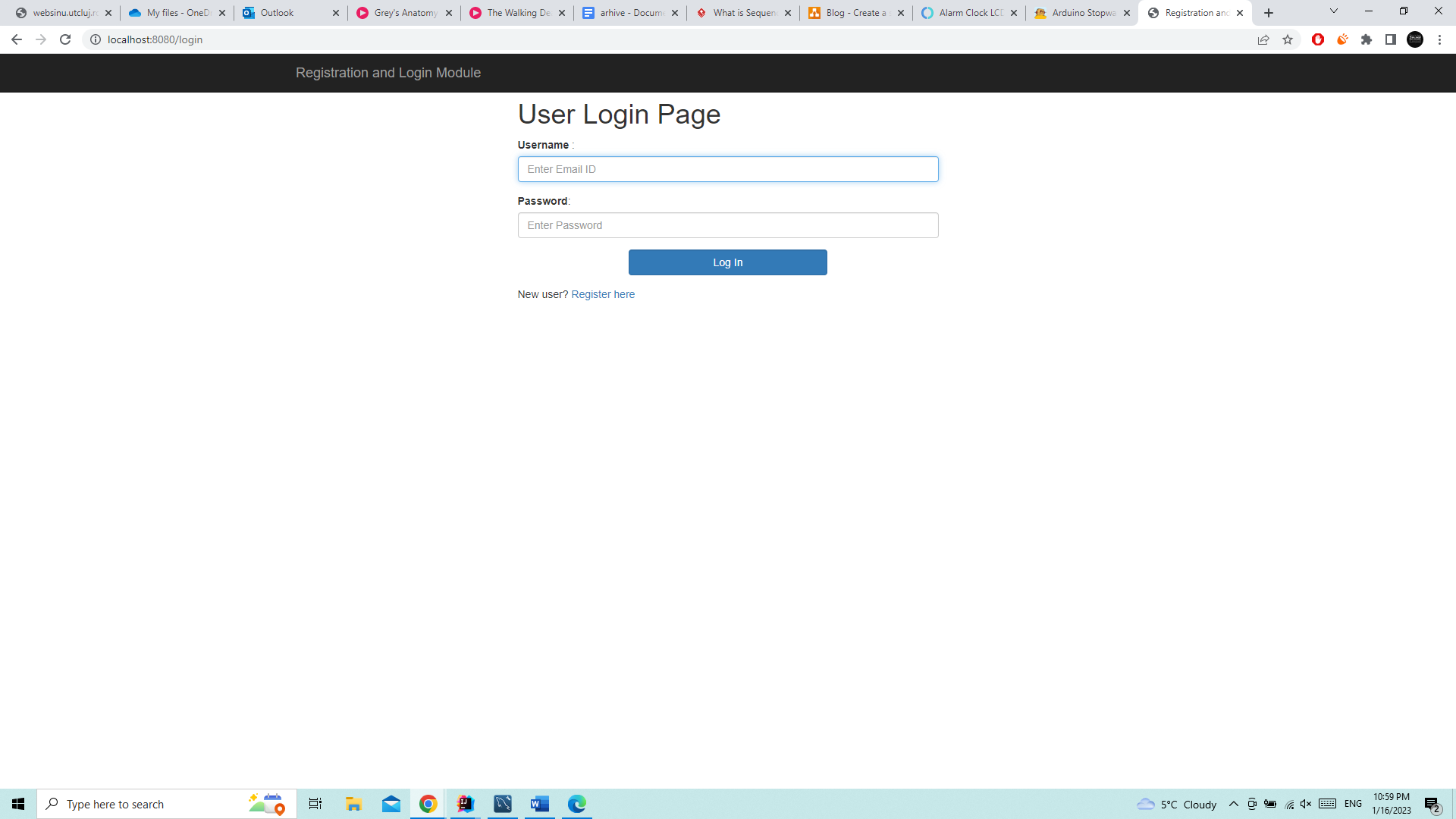
Design patternul ales de mine este Observer, pe care l am implementat atunci cand clientul plaseaza o comanda, primeste un mail de confirmare care poate fi vizualizat pe pagina din contul lui. Am vrut sa implementez si design patternul Composite pentru produse, un produs sa fie compus din mai multe produse, dar in final am decis sa fac produsele ca o singura componenta. In final, am decis sa implementez design patternul Factory method, care poate fi vizualizat la clasa meniu care are mai multe subclase, precum fel principal, desert si bauturi. De asemenea, toate clasele ServiceImpl implementeaza design patternul Factory method.

1. **Manual de utilizare**

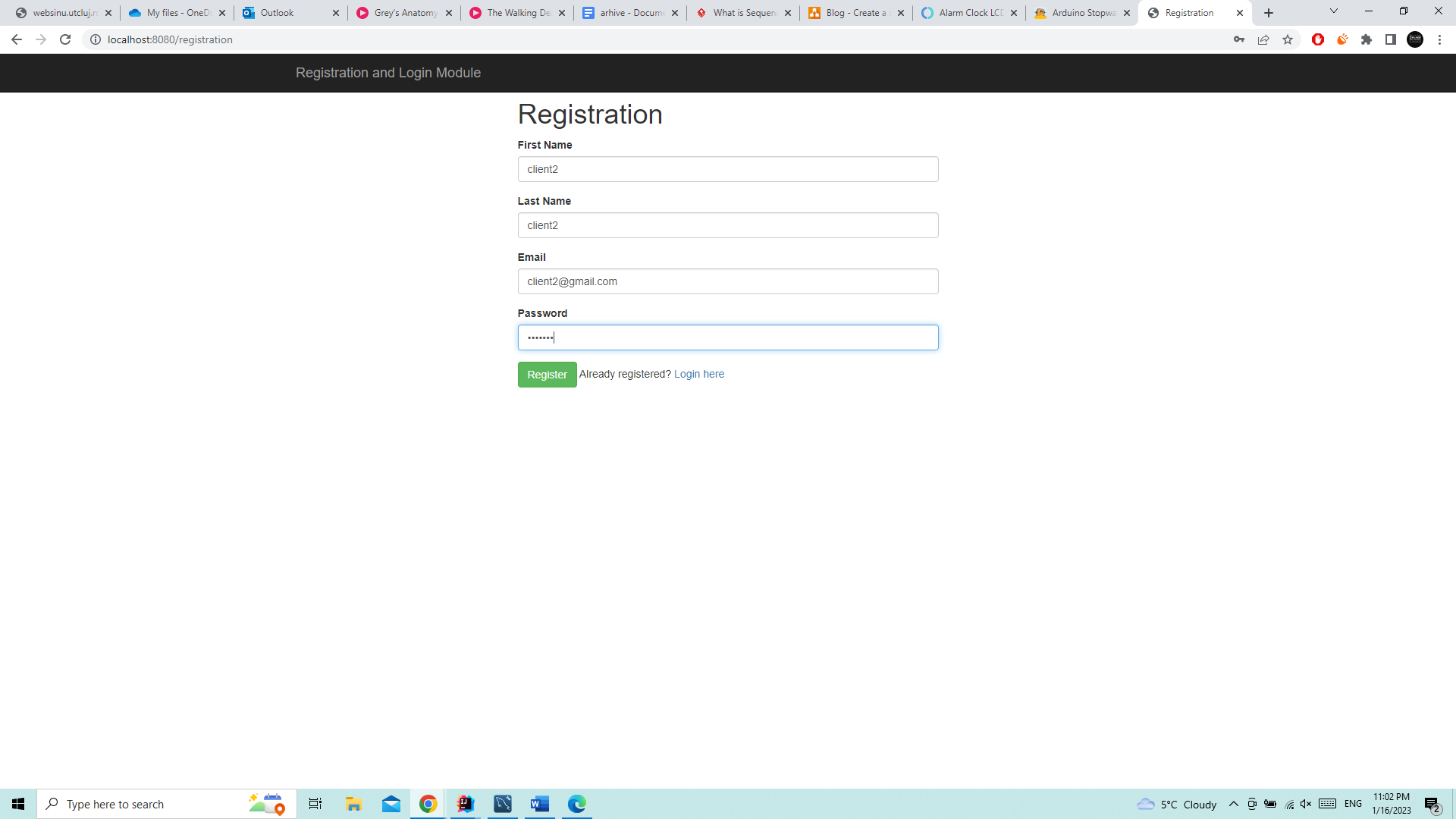
Pentru a putea folosi aplicatie sunt necesare: pentru programatorul care a proiectat interfata trebuie sa aiba acces la Intellij, programul unde s-a scris codul in java, la MySQL Workbench care contine baza de date cu tabelele necesare aplicatiei si un motor de cautare, precum google chrome. Aplicatia se ruleaza din Intellij, iar site-ul apare in motorul de cautare. Utilizatorul are nevoie doar de browser pentru a utiliza aplicatia si a plasa o comanda. Utilizatorul se poate loga sau inregistra daca nu are inca cont, iar apoi poate plasa o comanda.

Interfata aplicatiei de catering:

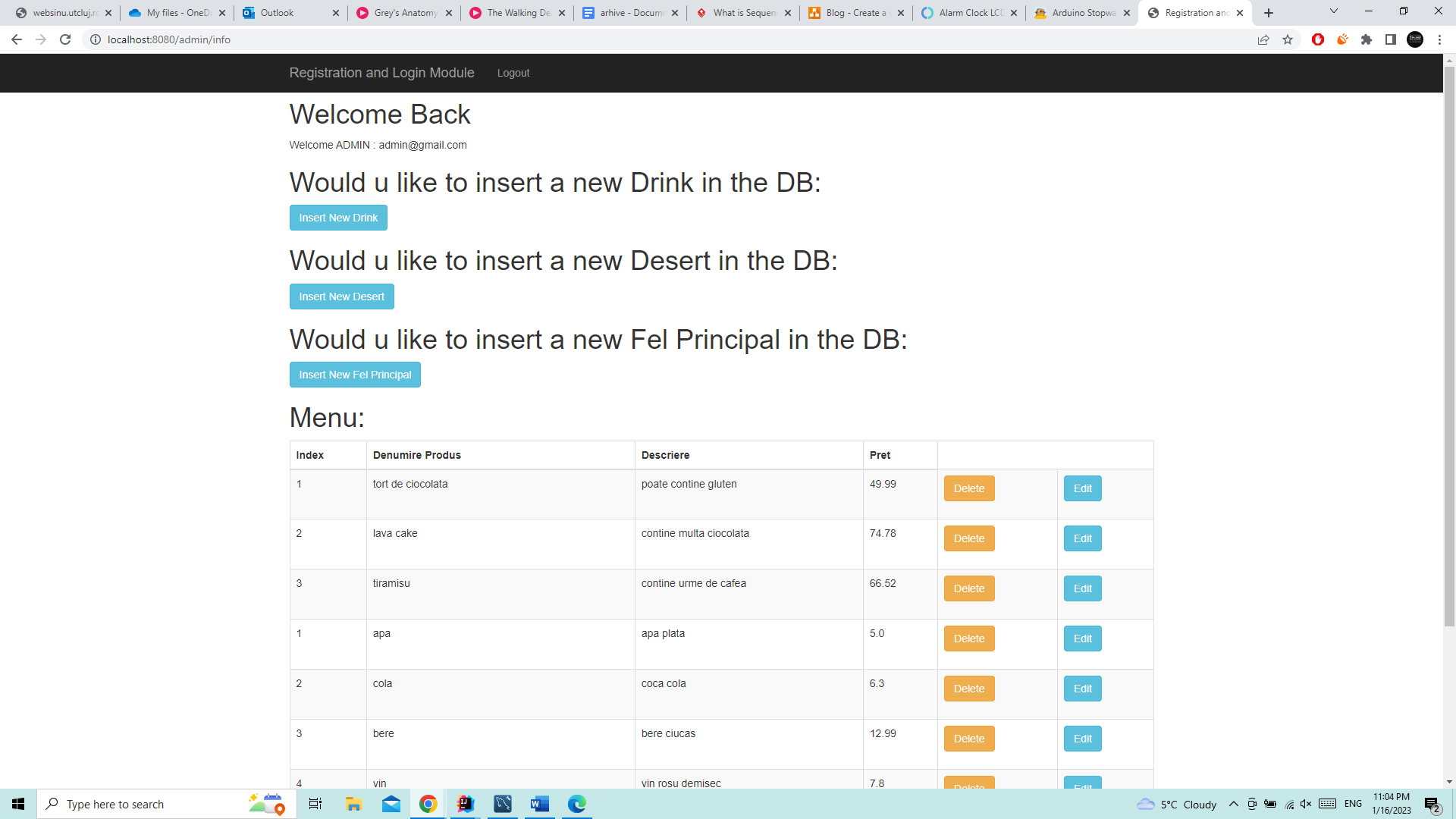
Pagina pentru logare:



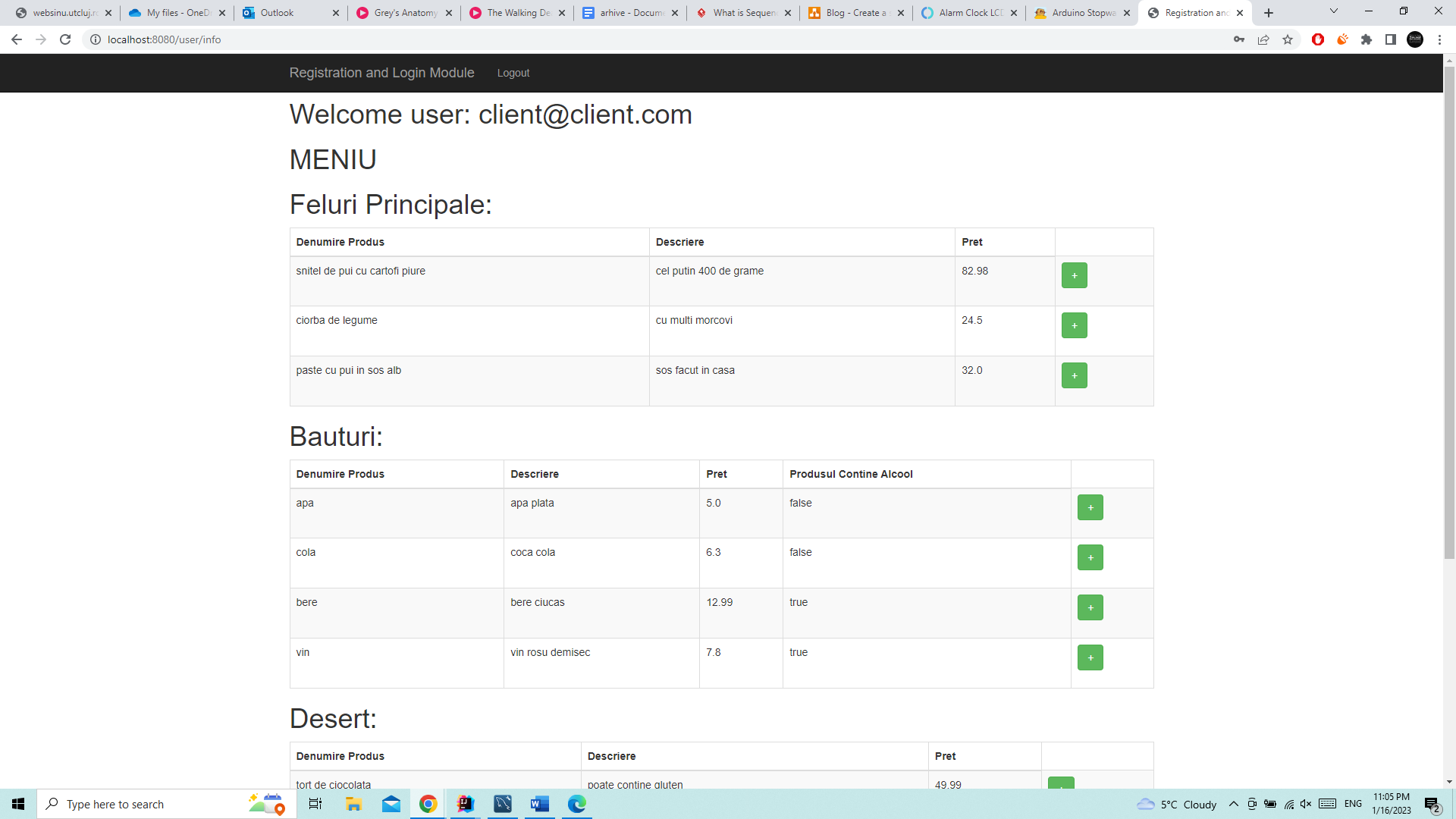
Pagina pentru inregistrare:

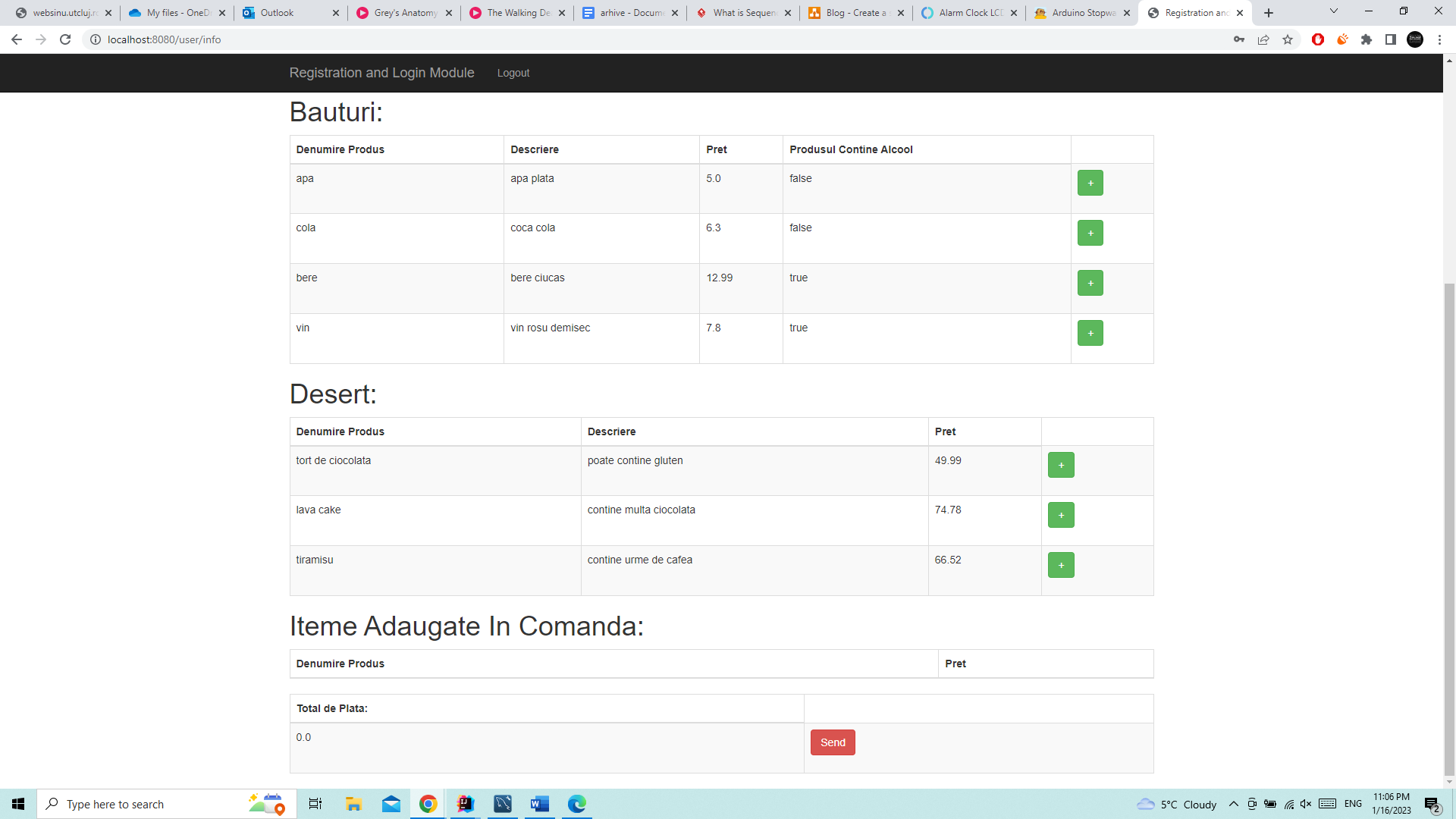


Pagina pentru admin:



Pagina pentru client:





1. **Tutorial de utilizare**

Aplicatia este implementata astfel incat sa fie foarte usor de utilizat pentru orice utilizator.

Ceea ce se intampla in spate: se ruleaza programul din Intellij avand MySQL deschis, acolo generandu-se tabelele.

Utilizatorul trebuie doar sa deschida un browser, sa caute si sa intre la adresa <http://localhost:8080>. Se va deschide pagina de logare reprezentata mai sus. Daca are deja cont, utilizatorul poate sa intre in cont cu emailul si cu parola, iar daca nu are se poate inregistra, sa-si creeze un cont apasand pe “Register here”, unde trebuie sa isi introduca numele, prenumele, emailul si parola.

Cand utilizatorul a intrat in cont, daca acesta este client apare pagina pentru client reprezentata mai sus. Clientul poate vedea, odata intrat in cont, o lista de meniuri cu bauturi desert si fel principal si poate sa comande ce isi doreste apasand pe butonul “+”. Daca clientul doreste sa renunte la ceva din lista de comandat, poate apasa butonul “-“, iar acel item se sterge din lista. Dupa ce a finalizat comanda, clientul poate vedea pretul total al comenzii si poate apasa pe “Send” pentru a trimite comanda.

Daca utilizatorul este admin, apare pagina pentru admin reprezentata mai sus. Adminul poate vizualiza toate produsele, sa adauge noi produse, sa stearga produse sau sa modifice produsele deja existente.

1. **Bibliografie**

<https://www.jetbrains.com/help/idea/spring-support-tutorial.html>

<https://app.diagrams.net/#G1ik4kH982Ov1lIrPwOoqrmmOpQP73266O>

<https://www.javatpoint.com/spring-tutorial-dependency-injection-by-constructor>