UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Magazin online de electronice

Name: Coroama Vasile
Group:30235

Table of Contents

Deliverable 1 ..... ***3***

Project Specification **3**

Functional Requirements **3**

Use Case Model **4**

Use Cases Identification 4

UML Use Case Diagrams 5

Supplementary Specification **6**

Non-functional Requirements 6

Design Constraints 6

Glossary **6**

Deliverable 2 ..... ***7***

Domain Model **7**

Architectural Design **8**

Conceptual Architecture 8

Package Design 8

Component and Deployment Diagram 9

Deliverable 3 ..... ***10***

Design Model **10**

Dynamic Behavior 10

Data Model 11

Class Diagram **11**

System Testing ***12***

Future Improvements. ***13***

Conclusion ***13***

Bibliography ***13***

Deliverable 1

Project Specification

Proiectul isi propune realizarea unei platforme web dedicate vanzarilor de componente, calculatoare si telefoane. Actorii principali ai acestui proiect vor fi clientul si administratorul, fiecare avand diferite functionalitati.

Functional Requirements

Principalele functionalitati ale proiectului, ce se vor implementa de-a lungul celor trei assignment-uri vor fi:

Assignment 1:

- Operatiile CRUD
- Meniul de logare
- Register

Assignment 2:

- "Am uitat parola" (in cazul in care un client isi uita parola, acesta poate sa si-o recupereze pe mail)
- Factura electronica ce va fi trimisa pe mail dupa ce un client va finaliza o comanda
- Autentificare in doi pasi
- Filtrare produse in functie de pret

Assignemnt 3:

- Notificari – cand pretul unui produs este schimbat de catre administrator, clientii vor fi notificati de acest lucru
- Garantie – Fiecare produs va avea un termen de garantie, care va scadea de la o zi la alta (va fi vizibil acest lucru pentru client). In cazul in care un client fcumpara un anumit produs care prezinta un anumit defect, poate sa faca o cerere catre admin pentru a returna produsul. In cazul in care termenul de garantie nu este depasit, clientul nu va mai putea trimite cererea respectiva. Cand administratorul va primi aceasta cerere, in cazul in care o va accepta, clientului i se va trimite un email in care i se specifica faptul ca cererea a fost aprobata.
- Vouchere – In cazul in care un anumit client cumpara mai multe produse, el va primi automat un voucher pe care il va putea folosi la achizitionarea altor produse.
- Validari Backend (Email, NotNull, Size, Pattern parola)
- Un meniu pentru administrator in care poate sa vada timestamp-urile de logare a tuturor utilizatorilor

Proiect final

- Validari frontend
- Descarcare in format xml/txt a timestamp-urilor de logare a clientilor (doar pentru administrator)
- Cautare de produse pe pagina principala (nu este necesara logarea).
- Criptare parole in baza de date

Use Case Model

Use Cases Identification

a.Use-Case: Finalizarea unei comenzi

Level: user-goal

Primary Actor: client

Main success scenario: Clientul adauga mai multe produse in cos, finalizeaza comanda si va primi un mail cu factura

Extensions: In cazul in care s-a retras un produs de pe platforma

b.Use-Case: Descarcare in format txt/xml a timestamp-urile de logare a clientilor

Level: user-goal

Primary Actor: administrator

Main success scenario: Administratorul poate sa descarce in format txt/xml informatii despre toate sesiunile de logare

Extensions: -

c.Use-Case: Cerere de garantie

Level: user-goal

Primary Actor: client si administrator

Main success scenario: Clientul poate sa returneze un produs defect prin intermediul unei cereri de garantie in cazul in care aceasta este inca valabila. In cazul in care administratorul accepta aceasta cerere, clientul va fi notificat de aceasta printr-un email.

Extensions: Daca cererea de garantie expira, clientul nu va mai putea sa returneze produsul

d.Use-Case: Adaugarea, stergerea si update-ul unui produs

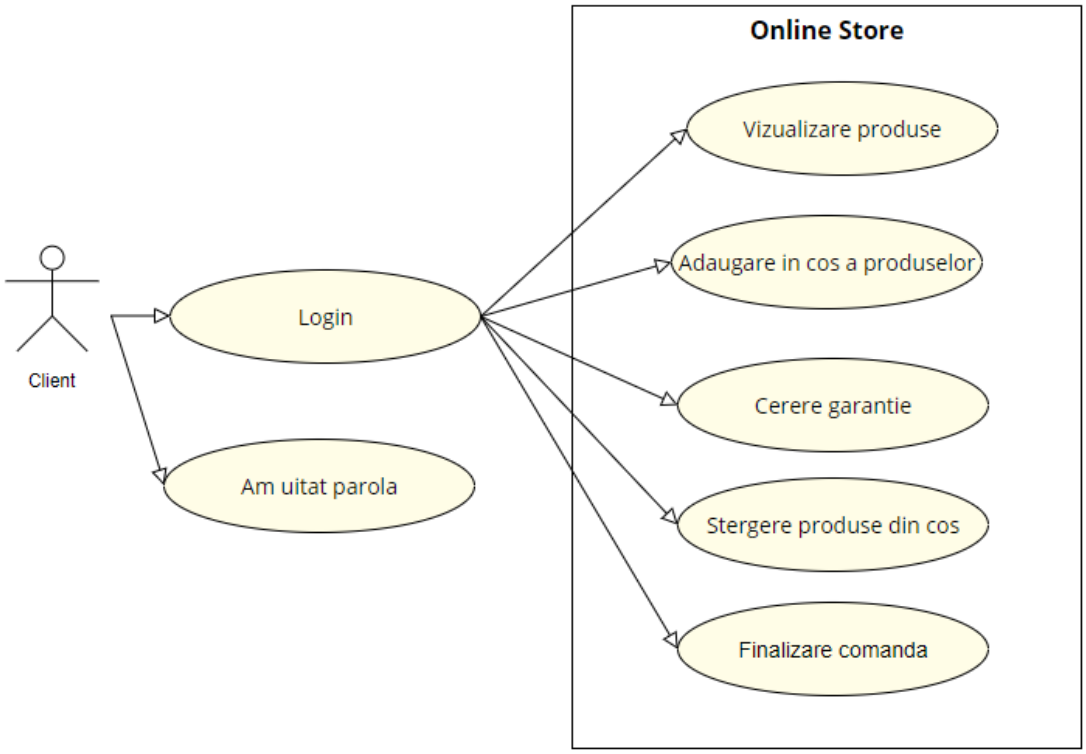
Level: function

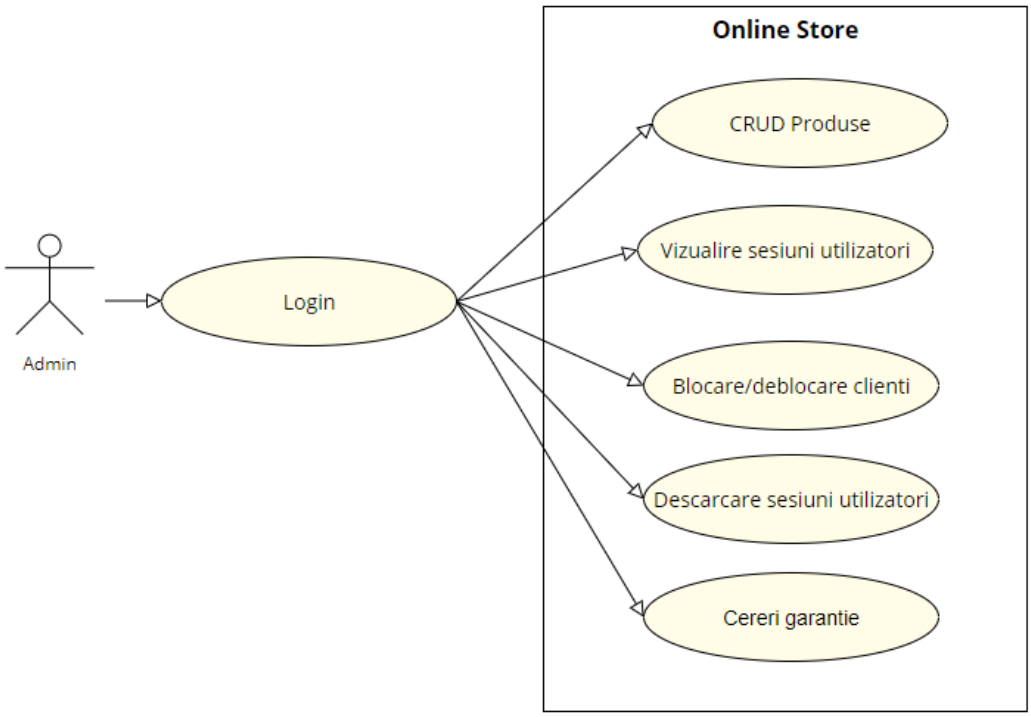
Primary Actor: administrator

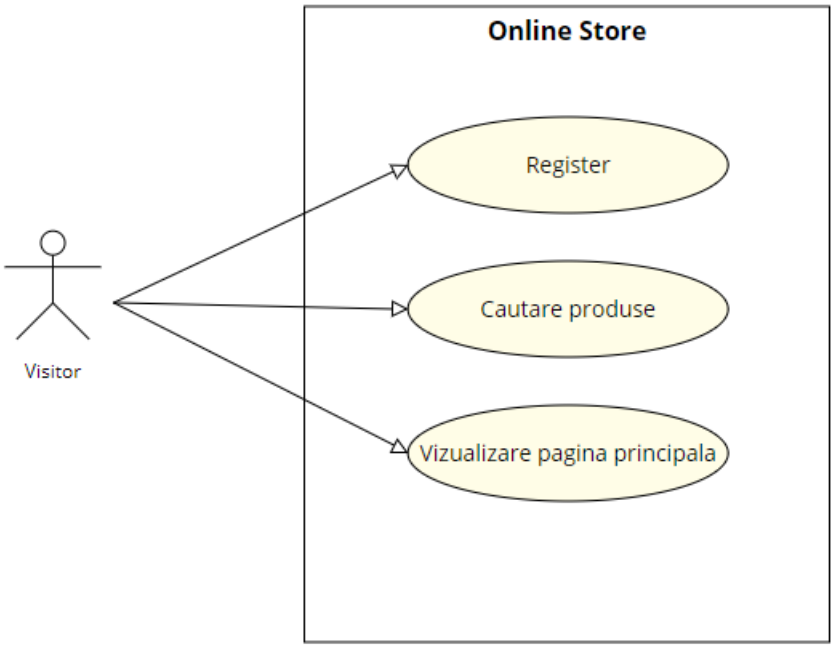
Main success scenario: Administratorul va avea drept de a modifica informatii despre un anumit produs aflat pe stoc, il poate sterge sau poate sa adauge unul nou.

Extensions: In cazul in care administratorul doreste sa adauge un nou produs cu o suma invalida, acesta va fi notificat si produsul respectiv nu va fi adaugat in baza de date.

UML Use Case Diagrams







Supplementary Specification

Non-functional Requirements

- Portability – platforma va fi disponibila pe orice sistem cu un browser instalat
- Security – Parolele din baza de date ale utilizatorilo vor fi criptate prin intermediul SpringSecurity
- Usability – Site-ul va fi usor de folosit de orice tip de utilizator
- Flexibility – Administratorul poate adauga functionalitati noi pe viitor sau sa schimbe interfata utilizator

Design Constraints

1. Dependinte de server (Tomcat – Java Spring)

2. Serverul de baze de date

3. Conexiune la internet (ruleaza pe local host)

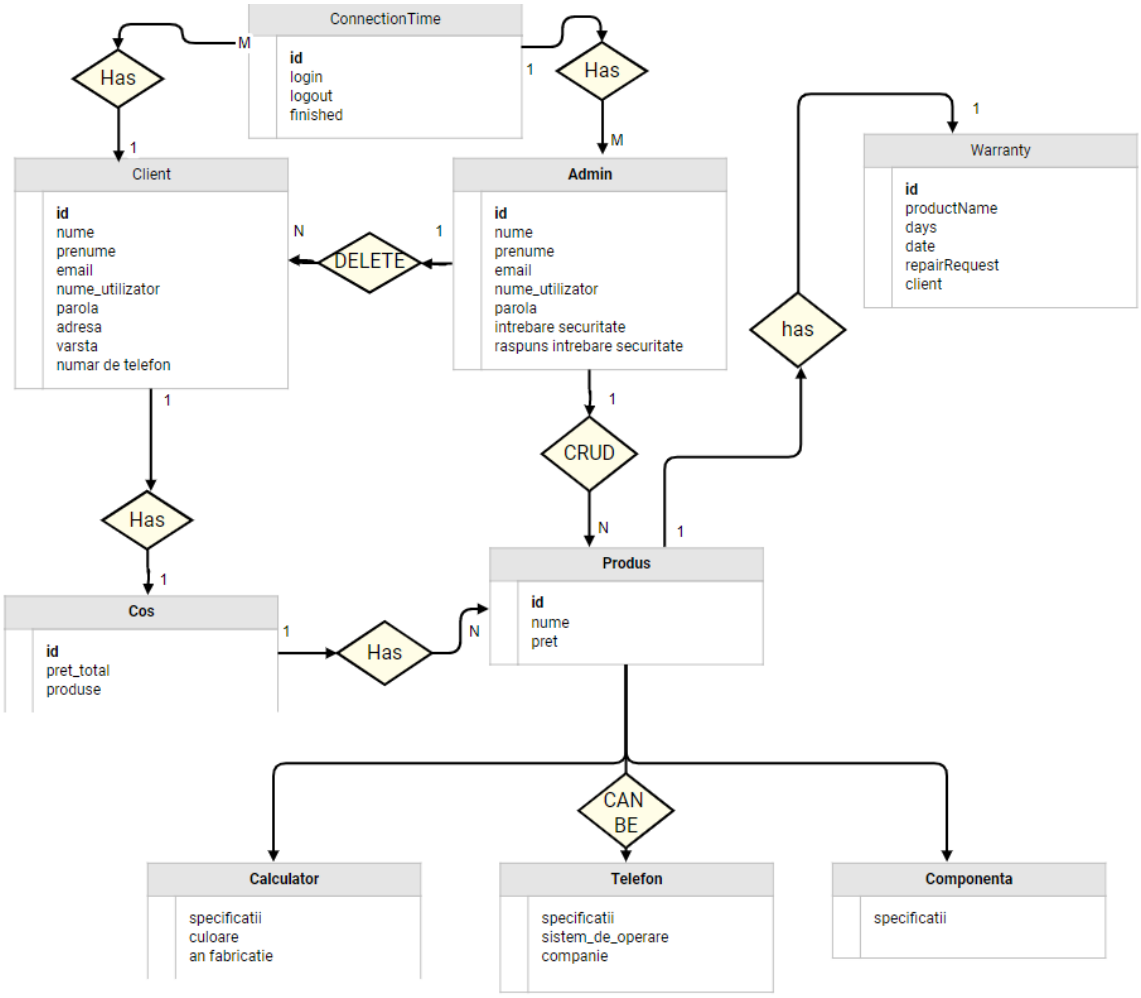
4. Dependinte hardware (pentru a rula aplicatia sunt necesare niste cerinte minime de hardware).

Glossary

CRUD = Create, Read, Update, Delete

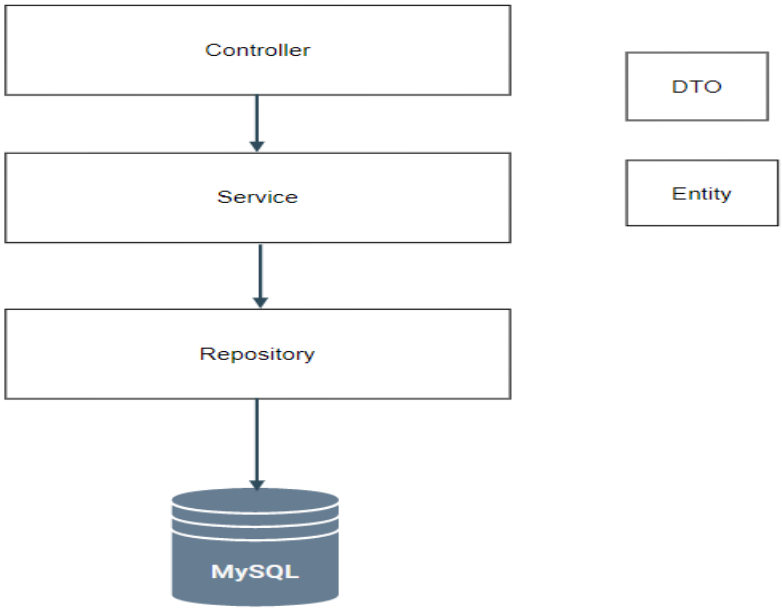
Deliverable 2

Domain Model

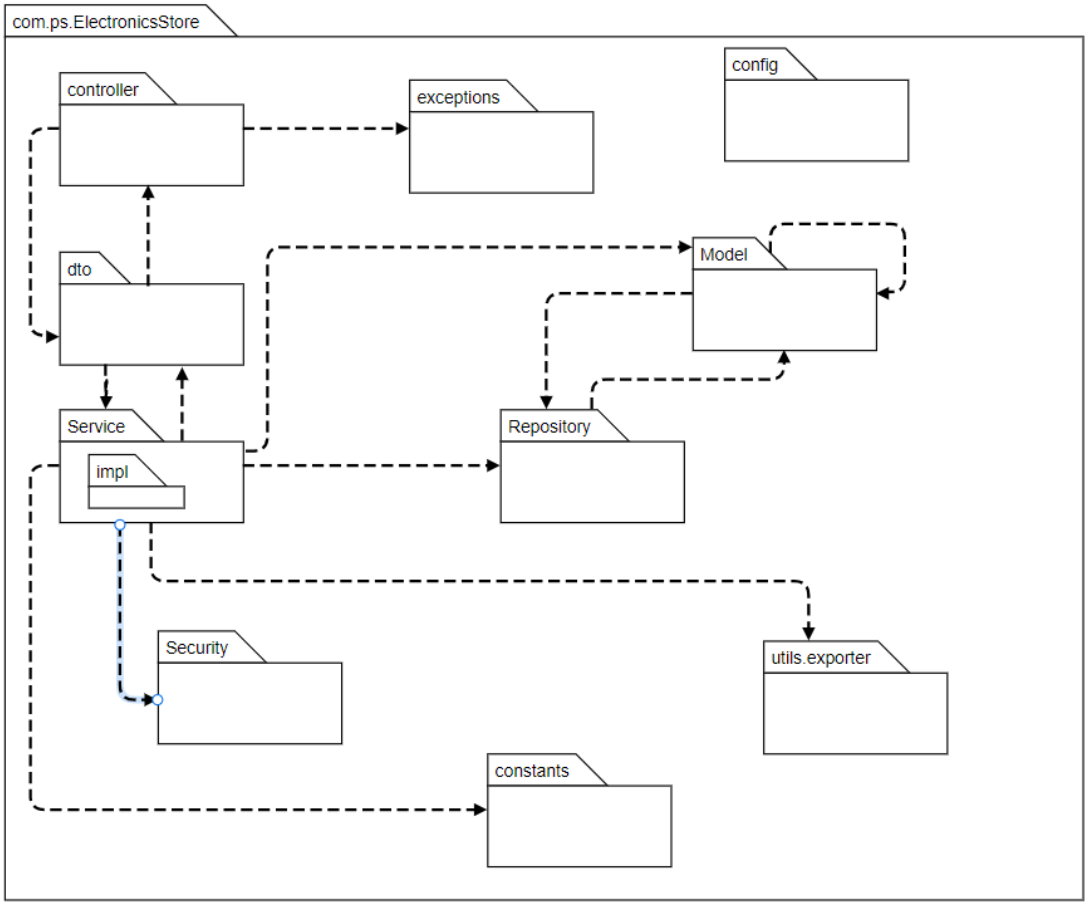


Architectural Design

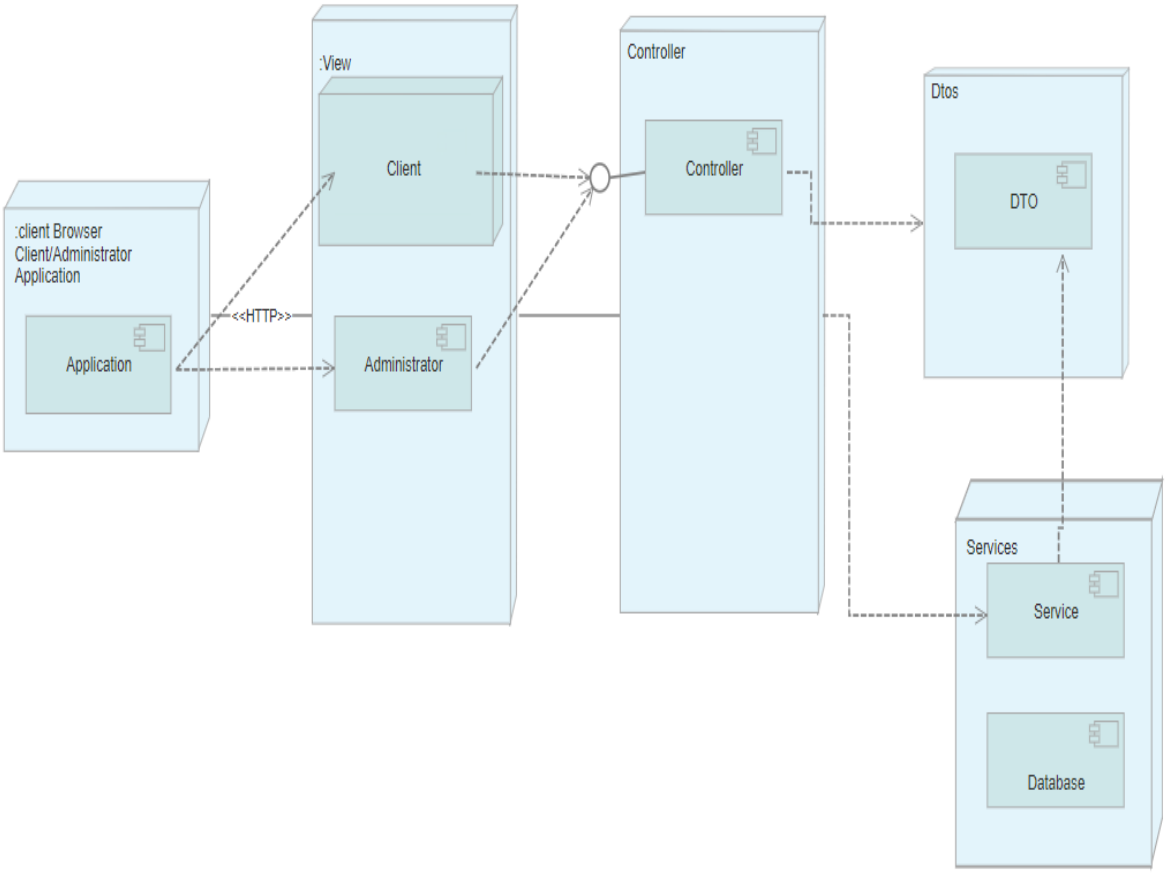
Conceptual Architecture

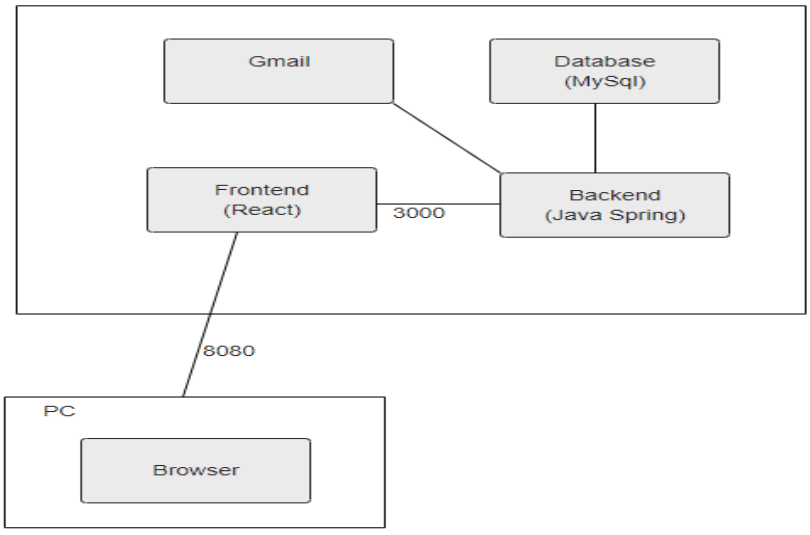


Package Design



Component and Deployment Diagram



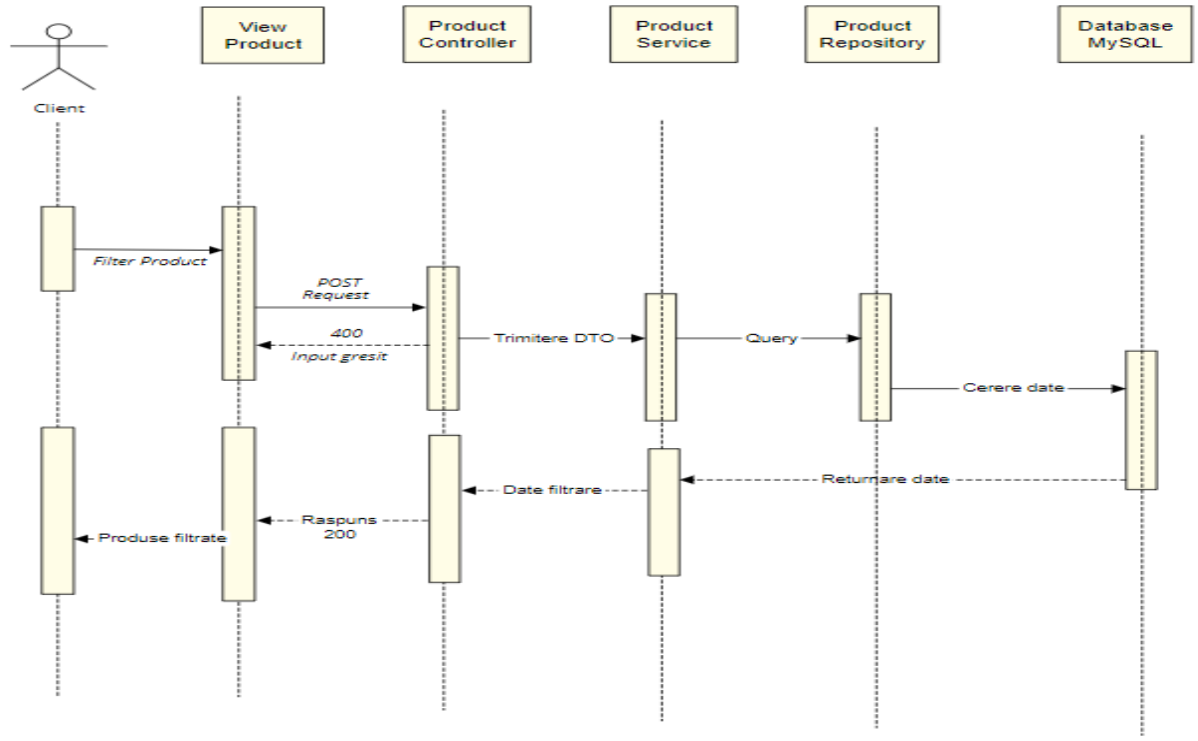


Deliverable 3

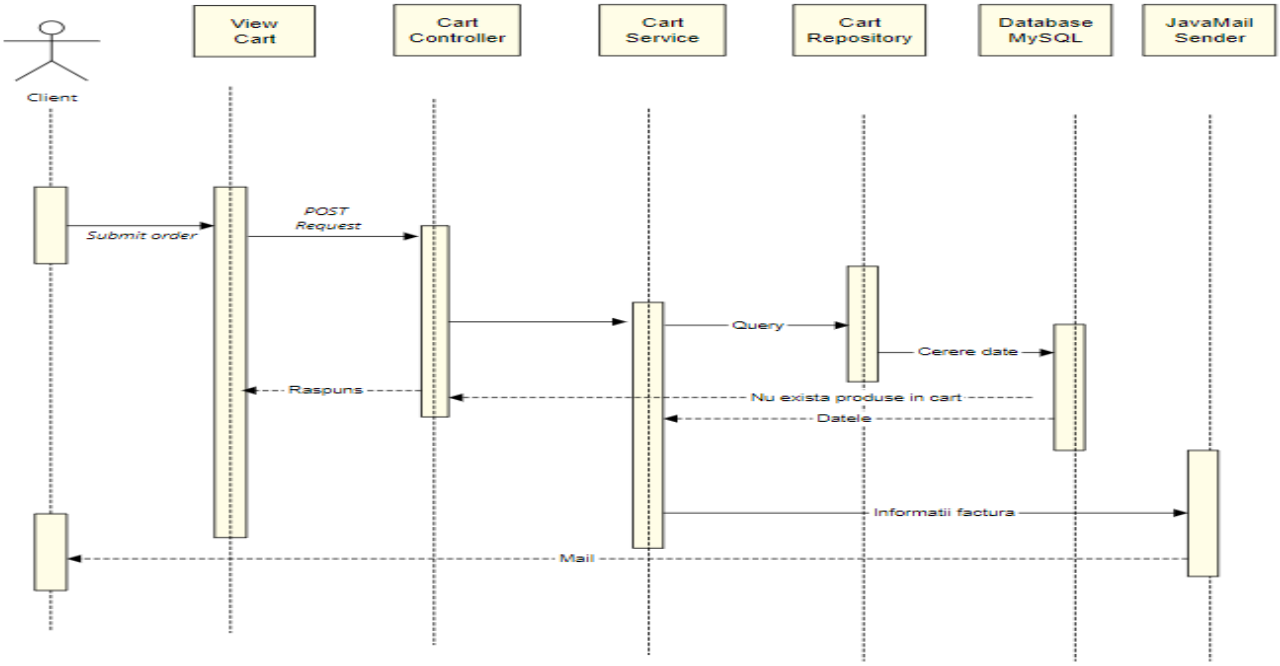
Design Model

Dynamic Behavior

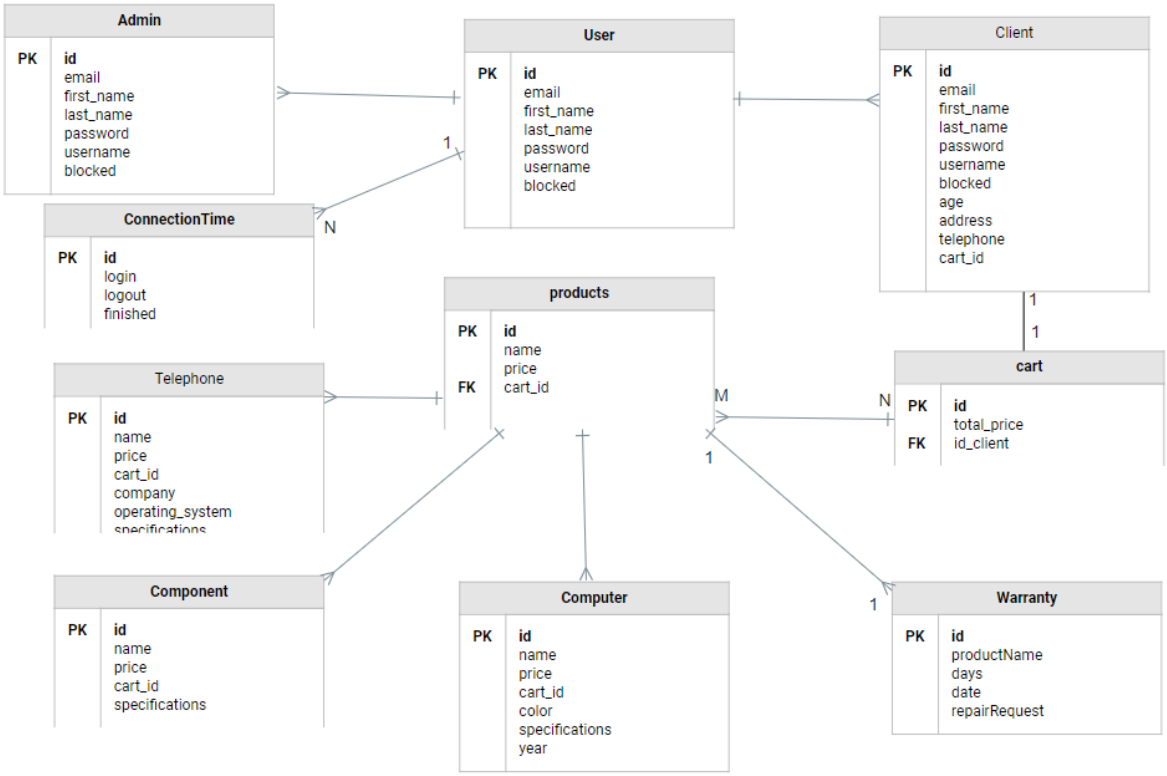
1.Diagrama de secvente pentru filtrarea produselor



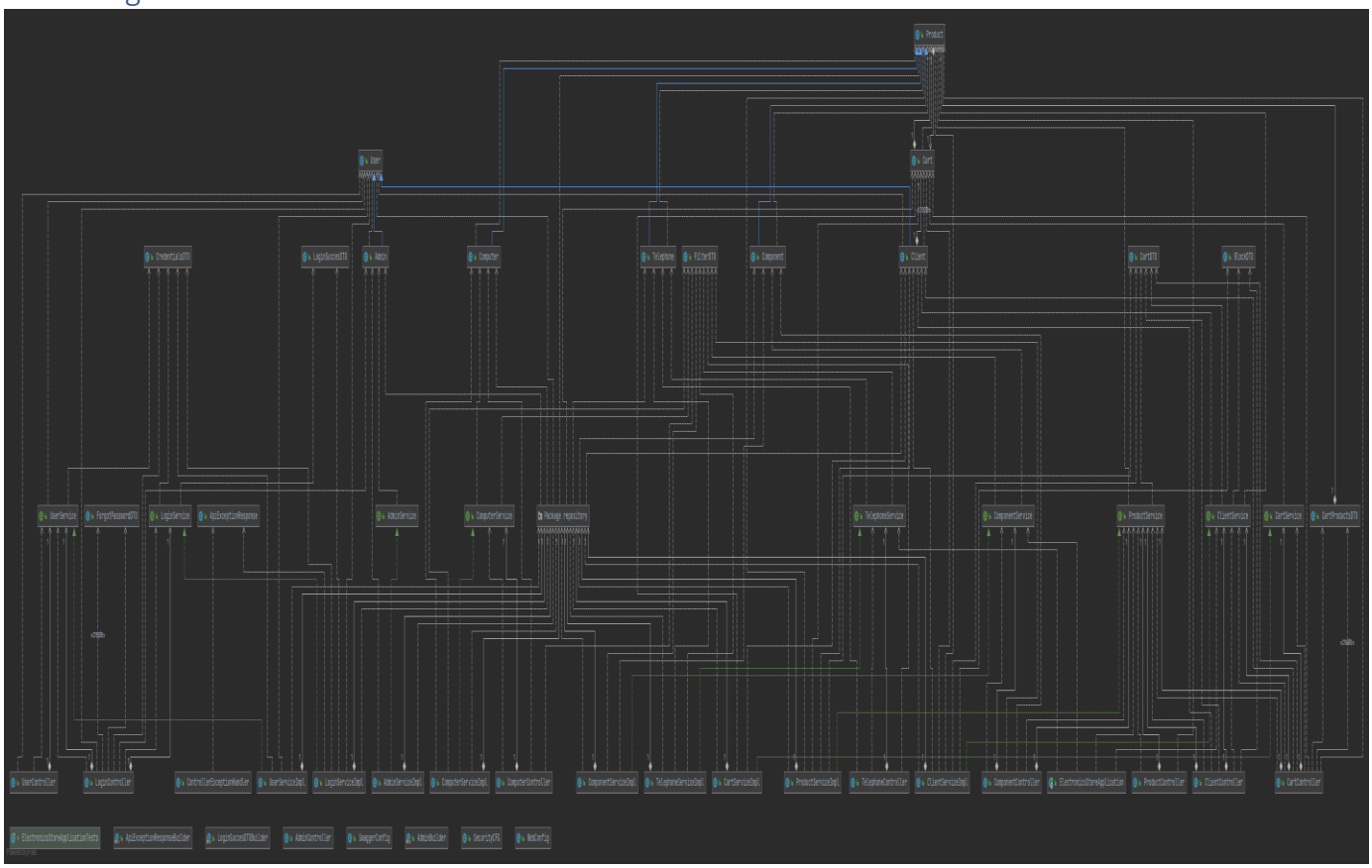
2. Diagrama de secvente pentru submit order



Data Model



Class Diagram



System Testing

Testele realizate pana acest moment se refera la functionalitatea operatiilor CRUD pentru clasa Computer.

1. Prima data testam daca un anumit computer este este gasit dupa nume, in cazul in care stim ca el exista in baza de date.

```
@Test
void givenExistingCalculator_whenFindByNume_thenFindOne() {
    //given
    calculatorService = new CalculatorServiceImpl(calculatorRepository);

    //when
    Calculator calculator1 = calculatorService.findByNume(NUME);
    //then

    assertNotNull(calculator1);
    assertEquals(NUME,calculator1.getNume());
}
```

2. Al doilea test se refera la modificarea unui anumit computer din baza de date.

```
@Test
void givenExistingCalculator_whenFindByNume_thenModificaPretCalculator() {

    //given
    calculatorService = new CalculatorServiceImpl(calculatorRepository);

    //when
    Calculator calc3 = calculatorService.findByNume("Asus2");
    assertNotNull(calc3);
    Long idCalculator = calc3.getId();
    System.out.println(calc3.getId());

    //then
    calculatorService.modificaPretCalculator(calc3.getId(),PRET_NOU);
    Calculator calculator3 = calculatorService.findByNume("Asus2");
    assertEquals(PRET_NOU,calculator3.getPret());
}
```

3. Cel de-al treilea test verifica daca un anumit calculator este sters, el nu va mai fi gasit in baza de date

```
@Test
void givenExistingCalculator_whenFindByNume_thenDeleteCalculator() {
    //given
    calculatorService = new CalculatorServiceImpl(calculatorRepository);
    //when
    Calculator calculator1 = calculatorService.findByNume(NUME);
    Long idCalculator = calculator1.getId();

    //then
    calculatorService.deleteCalculator(idCalculator);
    Calculator calculator2 = calculatorService.findByNume(NUME);
    assertEquals( expected: null,calculator2.getCuloare());

}
```

Future Improvements.

1. Adaugarea unui forum
2. Chat intre utilizatori
3. Adaugare poze la fiecare produs
4. Review pentru produse
5. Inregistrare cu retele de socializare sau google (Facebook, Twitter, Linkedin, etc)

Conclusion

Prin intermediul acestui proiect am reusit sa imi dezvolt abilitatea de a proiecta o platforma web cu ajutorul a doua tehnologii noi pentru mine: pe partea de backend framework-ul Spring, iar pe partea de frontend, biblioteca ReactJs.

Primul lucru pe care l-am invatat a fost legat de organizarea unei aplicatii in diferite pachete (service, repository, controller, model, etc) pe care o consider vitala, deoarece o aplicatie bine organizata creste cu mult eficienta scrierii de cod. Un alt lucru pe care am reusit sa il consolidez, a fost legat de bazele de date, mai exact alegerea corecta a relatiilor de mapare a tabelelor in diferite circumstante.

Legat de conexiunea dintre backend si frontend am reusit sa invat cum ar trebui sa fie realizat un controller si cum ar trebui sa creez diferite dto-uri in functie de requesturile necesare unui anumit task. Tot prin intermediul controllerului am reusit sa incorporez Swagger-UI, unde mi-a fost foarte usor sa testez daca request-urile aplicatiei sunt functionale.

Pe partea de frontend am invatat cum as putea sa folosesc diferite componente din @material-ui, cum sa trimit un request catre backend prin intermediul axios, sa descarc un anumit fisier de pe platforma si cum ar trebui sa lucrez cu functiile implementate in asa fel ca body-urile trimise catre backend sa contina informatii valide.

Pe final, am reusit sa implementez diferite validari cum ar fi verificarea unui email inainte de a fi adaugat in baza de date, un pattern pentru parola (utilizatorul va trebui sa isi aleaga o parola care contine cel putin o cifra) si o criptarea a parolelor din baza de date.

Bibliography