Facultatea de Acutomatică și Calculatoare

Inginerie Software

Aplicație Web

-Pixel Online Coffee Shop-

Profesor îndrumător:
Adrian Burzo

Studenți:
Balog Helga
Brebeanu Theodora
Dunca Denisa
Grupa: 30237

2021-2022

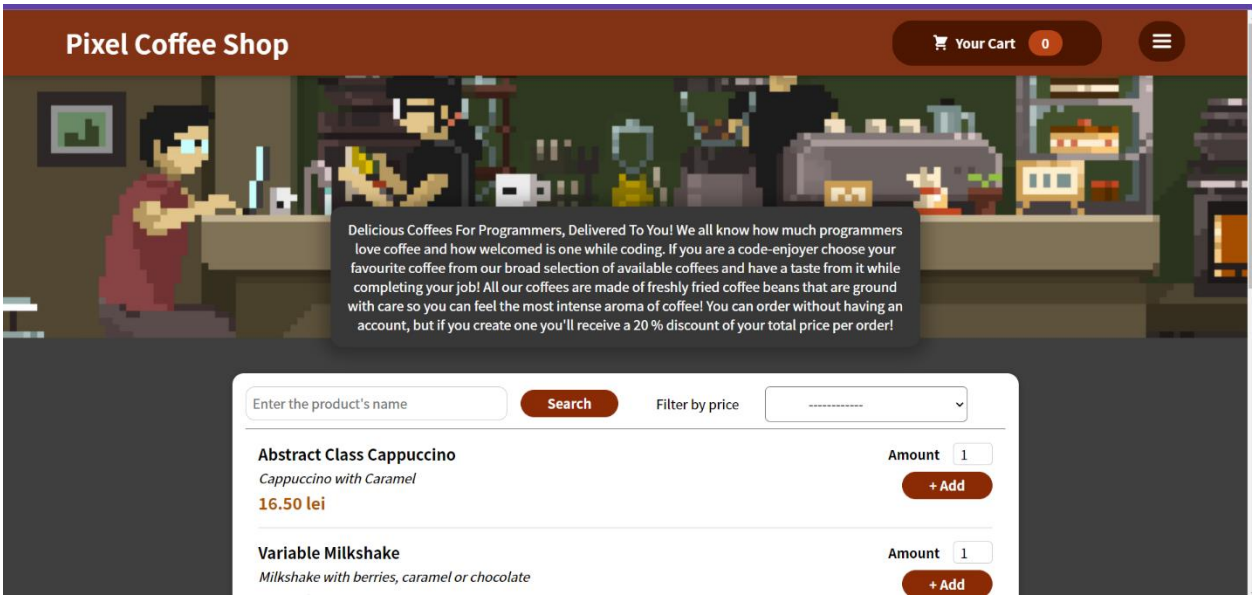
Cuprins

1. Introducere

În cadrul acestui proiect am reușit să implementăm o aplicație de tipul client-server utilizând pe backend Java și SpringBoot, pe frontend React JavaScript, iar pentru baza de date MySQL.

Proiectul este alcătuit dintr-o interfață destinata utilizatorului și o interfață specială pentru administrator. Cele două interfețe sunt puse sub forma unui site web, o cafenea online, și pun la dispoziție utilizatorului posibilitatea de a-și crea un cont, de a da log in și de a pune în coș și comanda băuturile alese din meniu. Administratorul are o interfață în care poate să adauge, să modifice sau să șteargă produse, poate sa observe comenzile primite și să le trimită la destinatar sau să le șteargă și poate să modifice descrierea de la pagina principală a site-ului.

Aplicația are la bază implementarea anumitor design pattern-uri, unele specifice React-ului (Hooks, Builder) și unele specifice aplicațiilor java Spring (Autowired Singleton). Pe parcurs au fost implementate și anumite use-case-uri precum diagrama de secvența, diagrama de acțiuni și diagrama de stare. Designul aplicației este realizat cu ajutorului limbajului javaScrip specific React-ului și limbajul CSS.



Coffee Shop Site

2. Componente

Aplicația este alcătuită din două părți principale, partea destinată utilizării de către client și partea destinată utilizării de către administrator, cea din urmă fiind și ea la rândul ei împărțită în două parți diferite, partea administrator produse, unde acesta poate să adauge, să modifice și sa

șteargă produse, și partea administrator comenzi care poate să trimită sau să șteargă o comandă. În cele ce urmează v-a prezentată în amănunt fiecare parte enumerată mai sus.

2.1 Client

Interfața destinată clientului unește accesul ușor la informație cu multitudinea de acțiuni pe care acesta poate să le facă.

Un prin aspect direct observabil de client este accesul la un meniu în bara de sus a site-ului, meniul în care sunt prezente numele magazinului, coșul de cumpărături și butonul care conține cele două opțiuni de log in sau de creare cont. Butonul ce conține cartul are un număr care sugerează în timp real câte produse au fost adăugate în cart, iar butonul de log in deschide un mic form unde pot fi completate numele si parola pentru a se intra în cont.

Următorul aspect este descrierea magazinului care oferă informații despre produse și despre eventualele reduceri la comenzi, în cazul de față apare o reducere de 20% pentru clienții care si-au creat un cont.

Cel din urma aspect este meniul cu produse, fiecare produs are un nume, o listă de ingrediente sau o descriere și, desigur un preț. Utilizatorul poate să aleagă dintr-o gama largă de produse, iar pentru a-i ușura căutarea am introdus două modalități de filtrare a informației, și anume, un search după numele produsului, unde clientul poate să scrie un nume sau o componentă din nume, iar meniul v-a primi un update, și o componentă de filtru a prețurilor care are două opțiuni de la cel mai mic la cel mai mare și respectiv, de la cel mai mare la cel mai mic. După ce clientul se hotărăște ce produs dorește să cumpere, printr-un simplu click pe butonul Add, poate să adauge în coș produsul, acesta poate să adauge mai multe produse de același tip, fie prin click repetat pe butonul Add, fie prin setarea unui Amount si apoi pe butonul Add.

Odată adăugate produsele dorite în coș, clientul poate să acceseze coșul de cumpărături prin deschiderea de la butonul Cart, aici apar toate produsele deja adăugate. Clientul poate să scoată produse din coș prin apăsarea butonului “-” sau să mai adauge prin apăsarea butonului “+”. După ce utilizatorul s-a hotărând cu privire la produsele pe care dorește să le comande poate apăsa pe butonul Order și trebuie să completeze câteva informații legate de adresă și cod poștal, iar apoi să apese pe butonul Confirm. Pentru confirmare a trimiterii comenzii, apare o notificare.

Odată ce utilizatorul termină de comandat/ utilizat site-ul acesta poate oricând să dea log out din contul său apăsând butonul de log out din aceeași zona unde erau și butoanele de log in si creare cont. De asemenea, clientul poate să observe că nu mai este logat atunci când nu ii va mai apărea numele în header-ul paginii.

În cazul în care utilizatorul dorește să își creeze un cont nou, prin apăsarea butonului Create Account poate să completeze înformații cerute și să adauge un nou cont in baza noastră de date, iar mai apoi se poate conecta cu el in form-ul de log in. Am creat site-ul astfel încât acesta să poată fii utilizat și dacă utilizatorul nu are un cont, acesta poate comanda, dar nu va beneficia de reducerea de 20% din comandă.

2.2 Administrator

În momentul în care se loghează administratorul cafenelei, acesta este dus la pagina principală.

Administratorul, la fel ca și clienții, poate vizualiza pagina principală, unde se observă descrierea și meniul cafenelei și poate, de asemenea, să trimită o comandă.În comparație cu clienții, amdinistratorul poate să editeze descrierea cafenelei. Cu un singur click pe descriere, textul devine editabil, iar pentru salvarea modificărilor se poate da click înafara câmpului de descriere sau se poate apăsa tasta Enter, ambele variante fiind la fel de simple și rapide.

Pentru a modifica produsele sau comenzile, utilizatorul poate apăsa pe butonul denumit Actions, din bara de sus a site-ului. Apăsând acest buton, administratorul este dus la interfața destinată lucrului cu produse, descris mai jos.

Tot în pagina principală, acesta are posibilitatea de a se deloga, apăsând pe cele trei linii din bara de meniu și alegând opțiunea de log out.

Deoarece am considerat că administratorul are mai multe întrebuințări, am decis să le împărțim în două subcapitole diferite, produse și comenzi.

2.2.1 Produse

Interfața destinată administratorului pentru întreținerea meniului de cafele, prezintă posibilitatea de a modifica produse, a șterge și de a adăuga produse noi.Produsele sunt afișate într-un tabel cu următoarele coloane: **nume, descriere, preț și acțiune**. În câmpul acțiune, administratorul poate alege între a edita sau a șterge produsul curent. Dacă apasă pe butonul de ștergere produsul va fi șters definitiv din baza de date a cafenelei. Dacă însă apasă pe butonul de editare, câmpurile din tabel, de la produsul curent vor putea fi editabile. Utilizatorul poate alege între a salva modificările făcute și între a renunța la ele, apăsând una dintre butoanele care se găsesc la câmpul **acțiune.**

De asemenea, administratorul are posibilitatea de a adăuga produse noi. Această acțiune este posibilă prin completarea câmpurilor de sub tabel cu datele necesare și prin apăsarea butonul pe care scrie add. În cazul în care în câmpul de preț nu se trece un număr real, utilizatorul va primi o notificare în care este informat de greșeala făcută și rugat să o corecteze pentru a putea adăuga produsul în meniu. O altă condiție pentru a putea adăuga produse este de a completa toate câmpurile. Niciunul nu poate rămâne gol.. Urmărind aceste condiții, administratorul poate cu ușurință să adauge un produs nou.

În bara de sus a site-ului, se află un buton Orders, prin care administratorul poate să comute simplu și rapid la interfața dedicată vizualizării comenzilor. La fel, se află un buton Home, care îl conduce pe utilizator la pagina principală, descrisă mai sus, unde se poate deloga

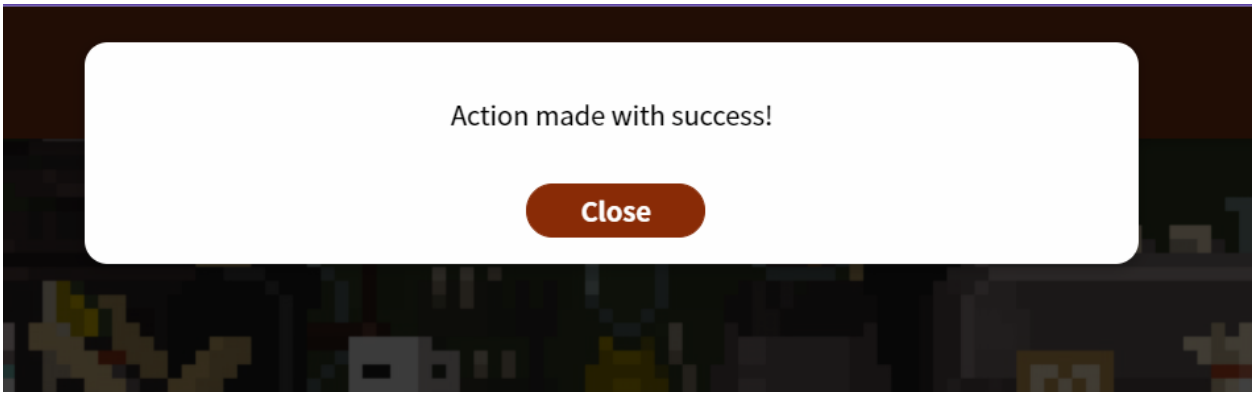
2.2.2 Comenzi

Printr-o simplă apăsare a butonului Orders din header-ul adminului, acesta are acces la tabelul cu toate comenzile primite de la clienți. Administratorul poate sa observe în tabel trei coloane principale și anume **userData , Products și Actions.**

Coloana userData conține toate informațiile introduse de către client la completarea chestionarului de comandă. Conține numele clientului, strada unde v-a fi livrata comanda, orașul și codul poștal. Coloana Products conține, desigur, lista cu produse pe care clientul dorește să le comande. Aceasta listă poate fi alcătuită din unul sau mai multe produse, și desigur prețul total al comenzii. Fiecare produs are anumite date despre el puse, și anume: numele produsului, numărul de produse din acel tip și prețul acestuia. Toate prețurile sunt adunate și puse la final in Total amount pentru ca administratorul să observe ușor cât trebuie să achite clientul pe comanda respectivă. Atât datele despre client cât și datele despre produsele comandate sunt luate din tabelul orders din baza de date, acolo unde au fost stocate atunci când clientul a trimis comanda.

Ultima coloana reprezintă acțiunile pe care le poate face administratorul pentru fiecare comandă. Acesta poate să trimită comanda spre client sau poate să o șteargă dacă observă că ceva nu a fost scris corect sau clientul vrea să o anuleze. Întrucât nu am implementat și partea de livrare, considerăm că administratorul comunică curierului datele, astfel butonul de send order doar va șterge comanda respectivă, atât din tabelul administratorului cât și din tabelul din baza de date.

Indiferent de acțiunea pe care o alege administratorul, acesta, după ce apasă pe unul dintre butoane va primi o notificare unde ii va fi confirmată acțiunea făcută, prin textul ”Action made with success!” și un buton pentru a închide notificarea. Pe pagină se află și două butoane products și home, care îl direcționează pe administrator la pagina cu tabelul de produse, respectiv la pagina de home.



Notificarea primită de administrator

3. Diagrame

3.1 Diagrame de secvență

Diagrama de secvență pentru partea de client.

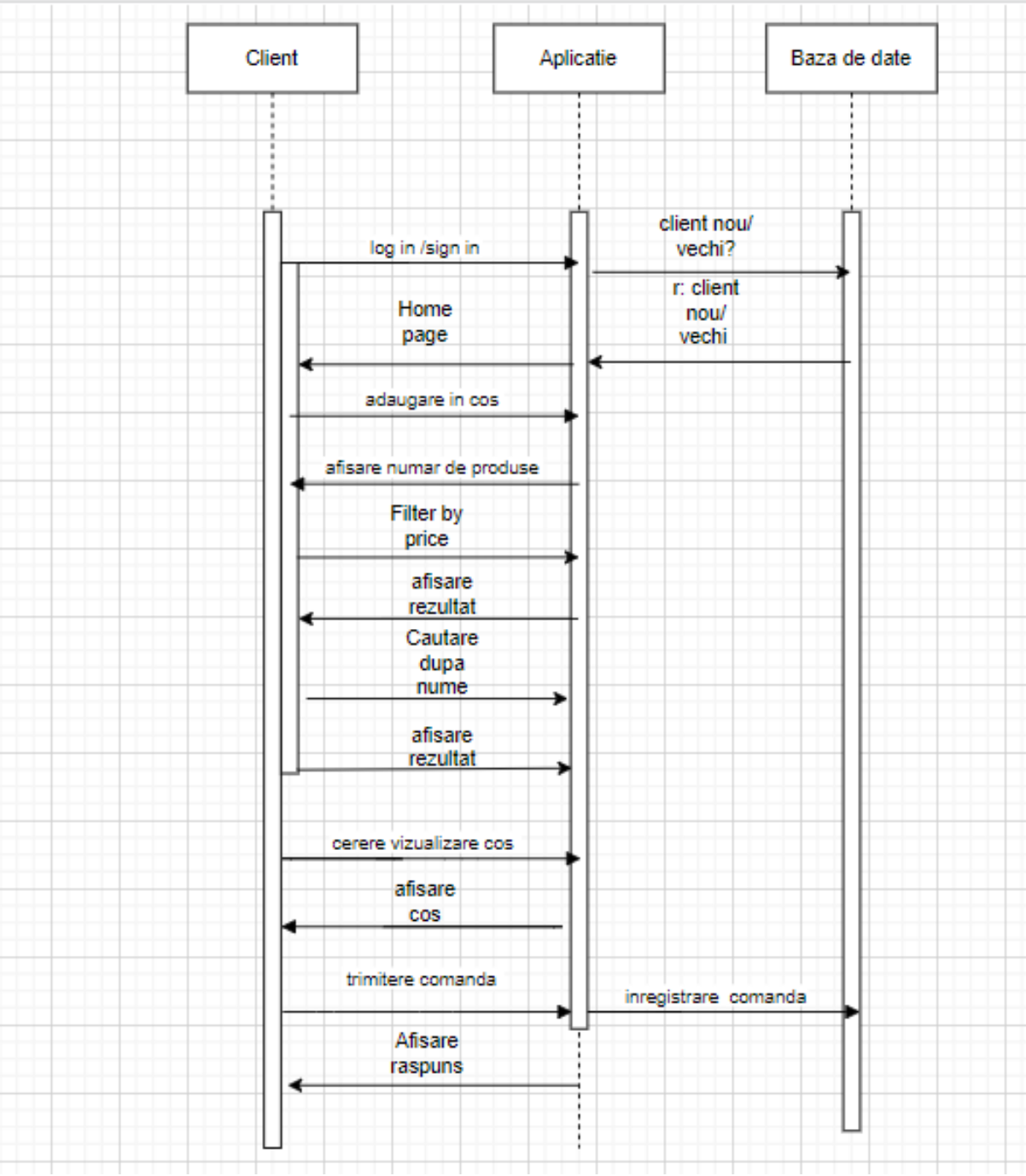
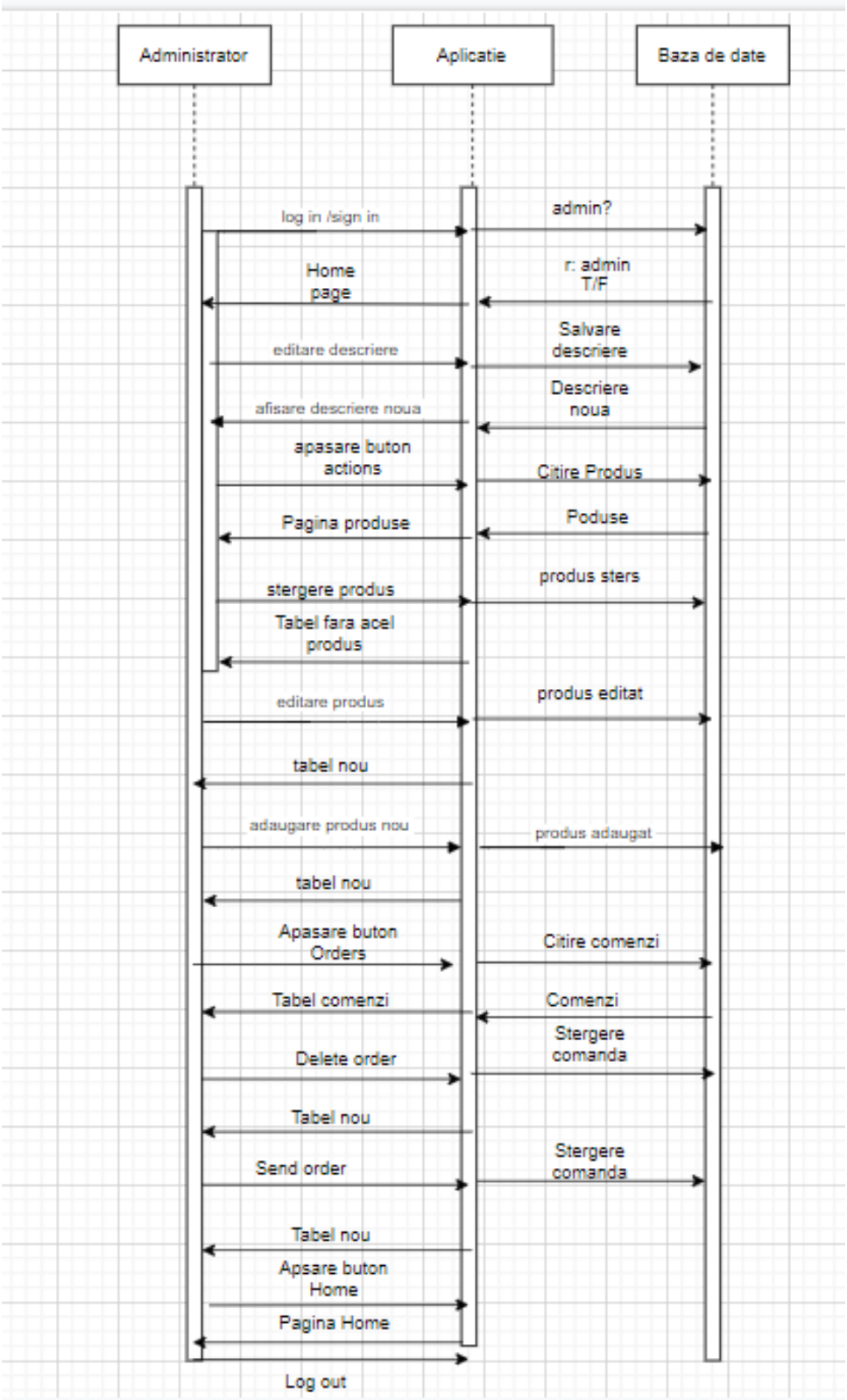
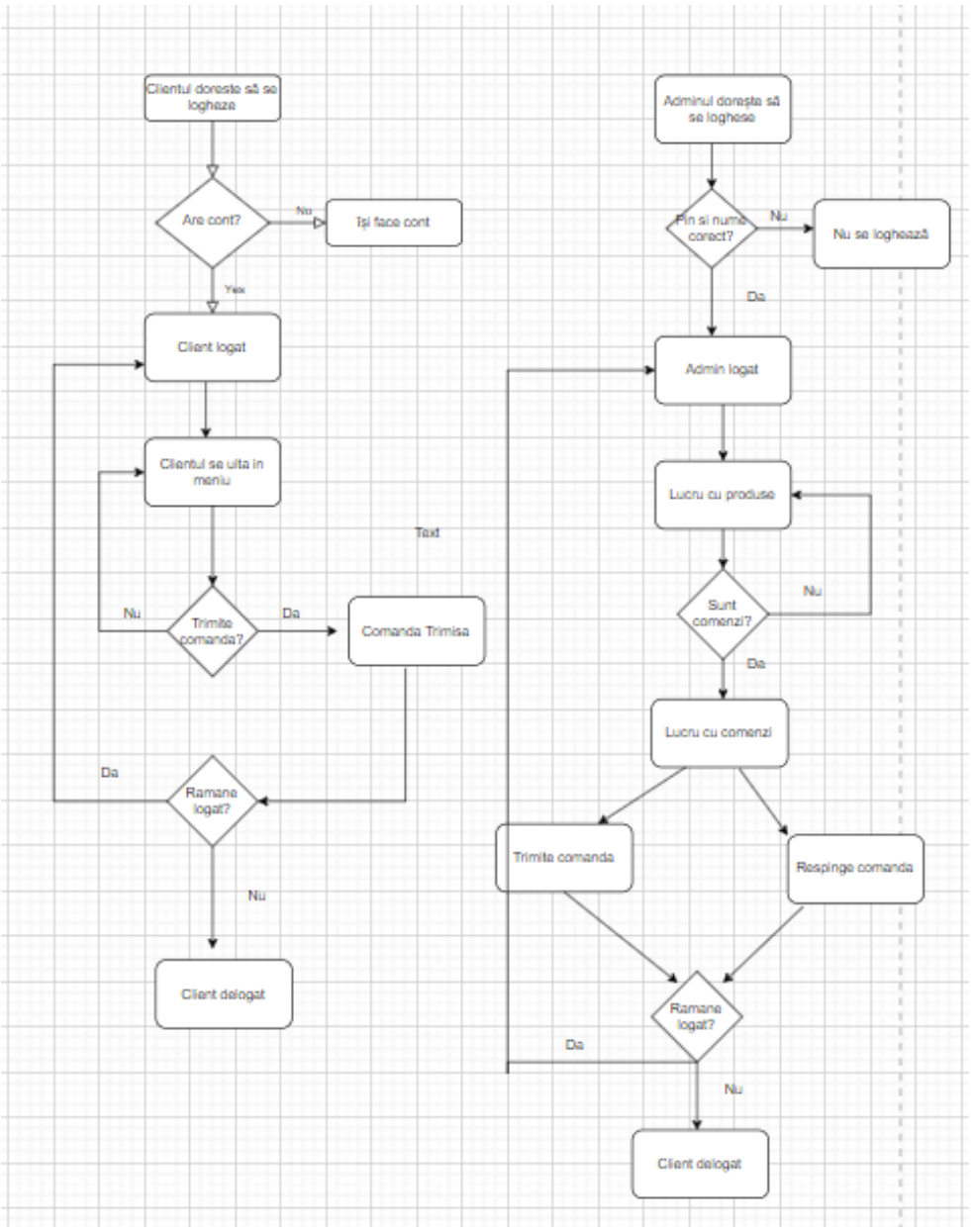


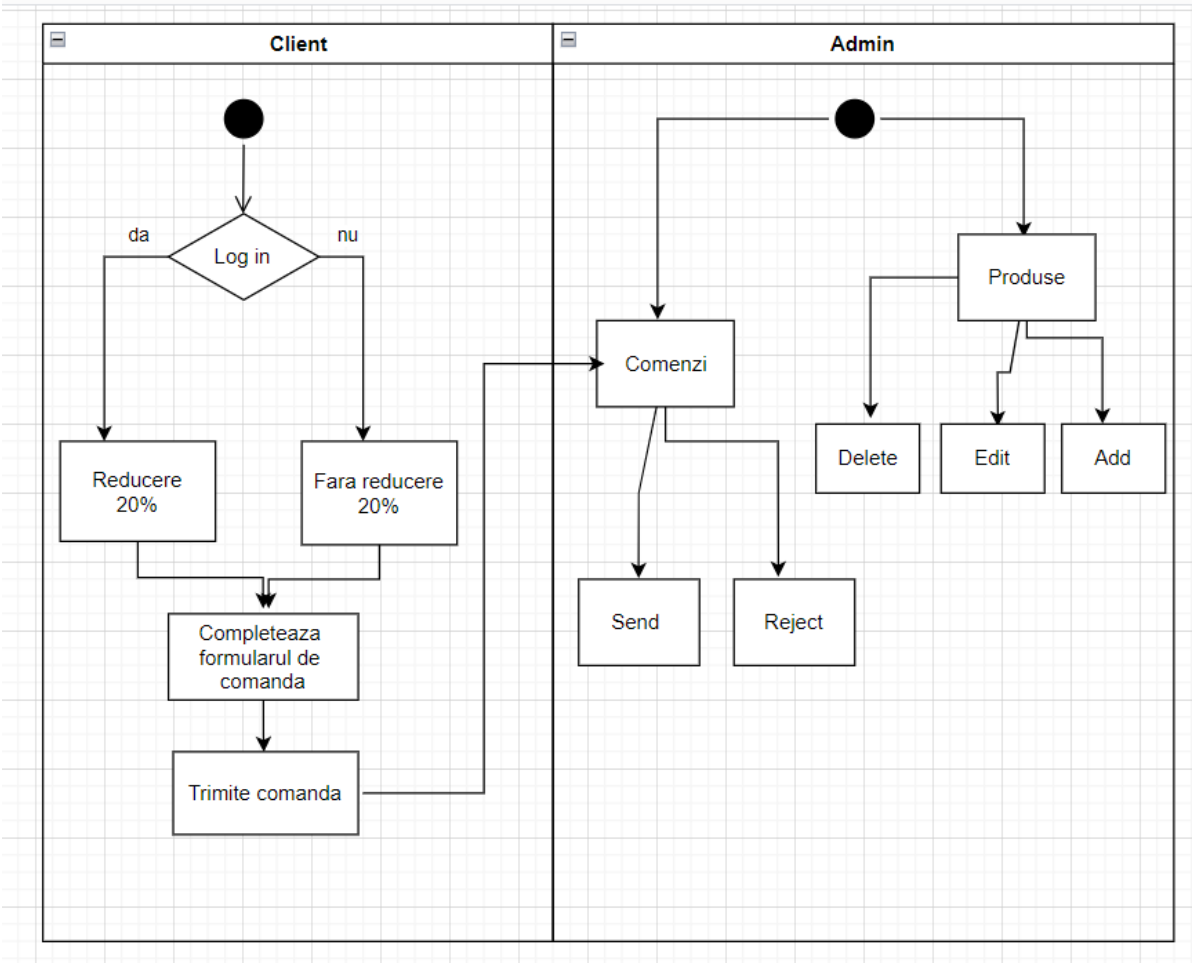
Diagrama de secventa pentru partea de administrator.



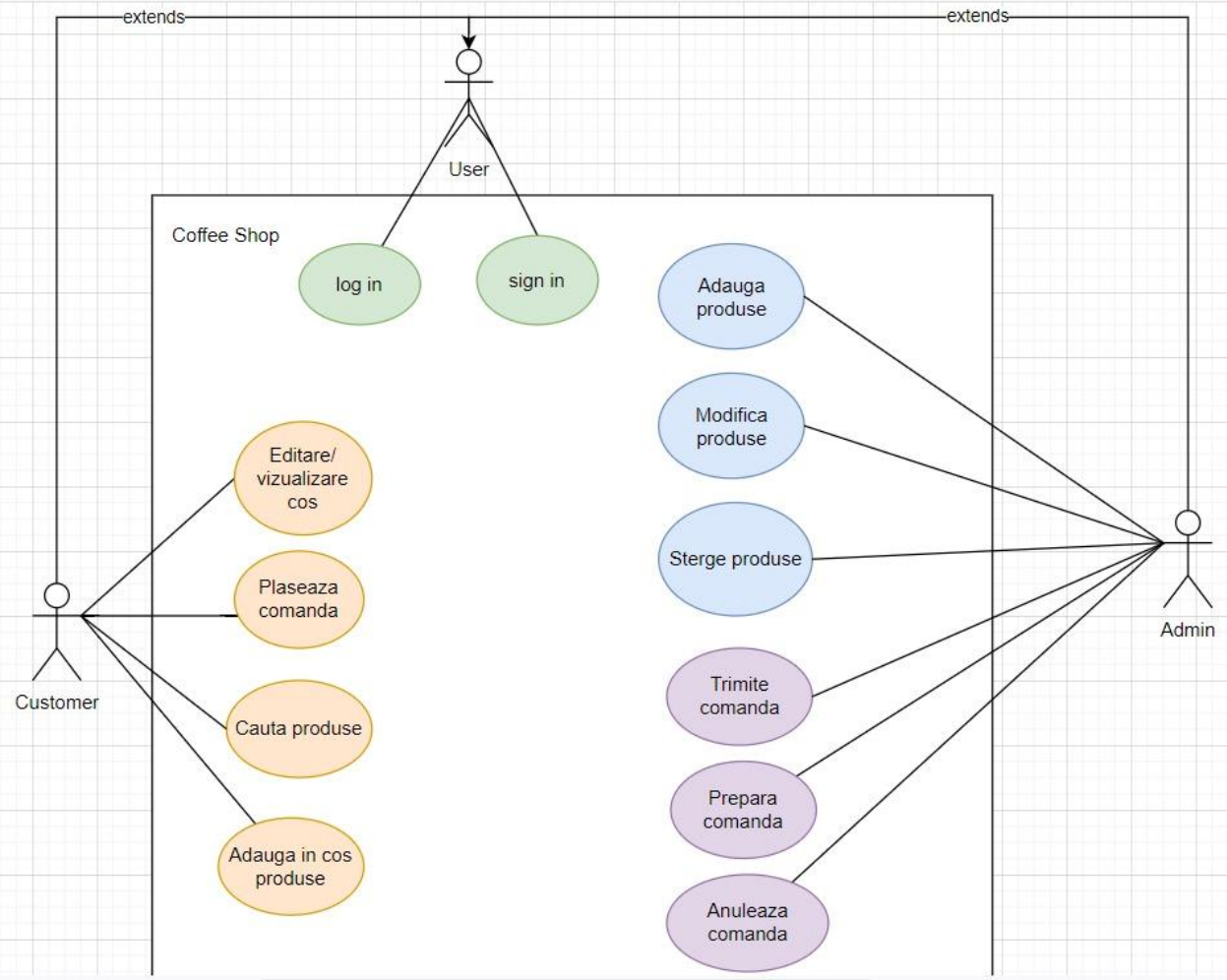
3.2 Diagrama de stare

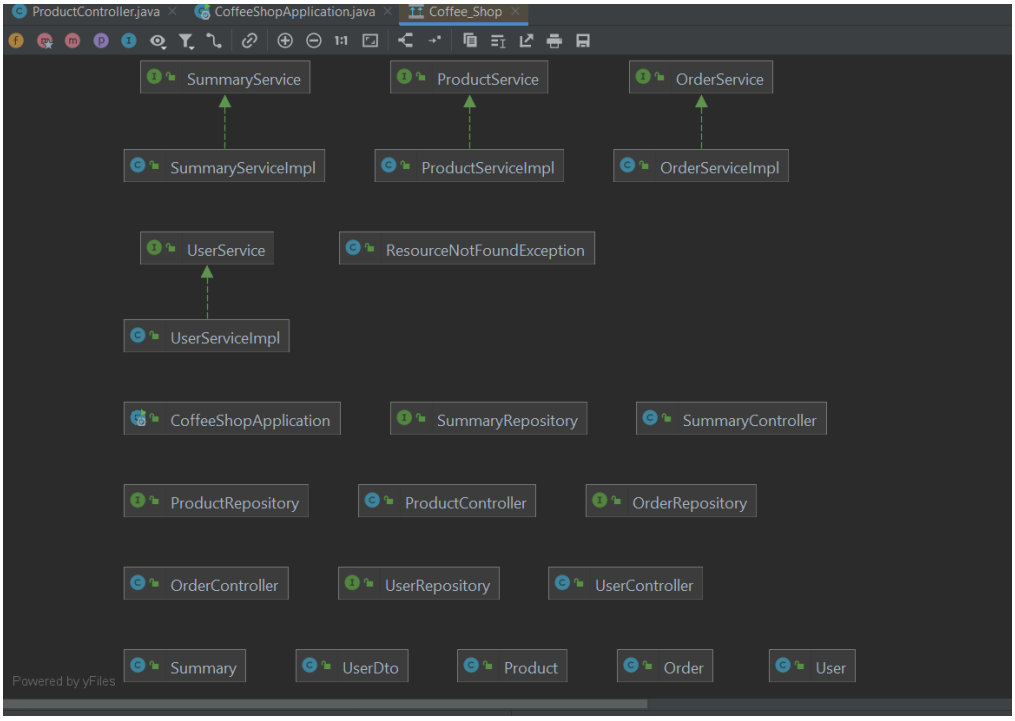


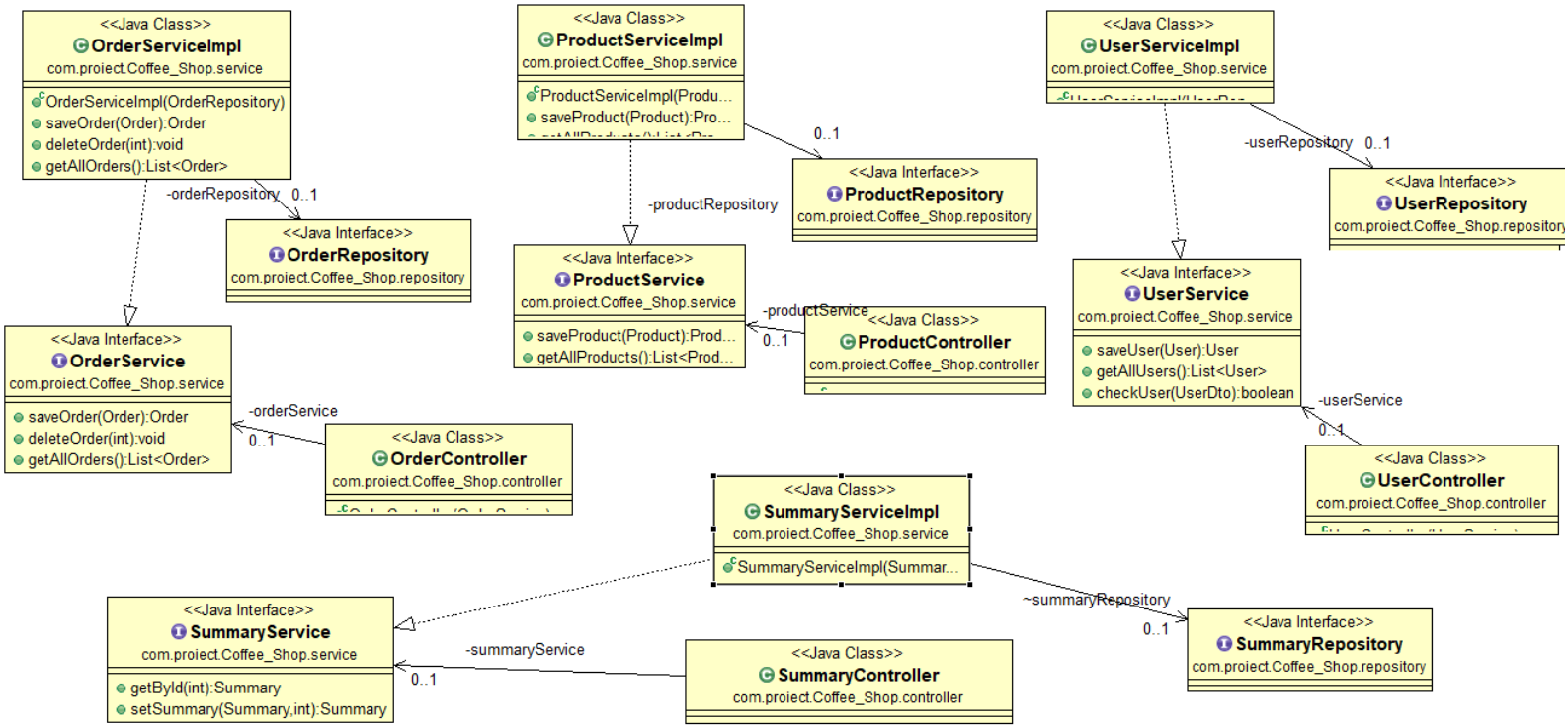
3.3 Diagrama de activități



4. Diagrame UML și Use Case




ProductController.java
CoffeeShopApplication.java
Coffee_Shop
SummaryService
ProductService
OrderService
SummaryServiceImpl
ProductServiceImpl
OrderServiceImpl
UserService
ResourceNotFoundException
UserServiceImpl
CoffeeShopApplication
SummaryRepository
SummaryController
ProductRepository
ProductController
OrderRepository
OrderController
UserRepository
UserController
Summary
UserDto
Product
Order
User
Powered by yFiles


<<Java Class>>
OrderServiceImpl
com.proiect.Coffee_Shop.service
OrderServiceImpl(OrderRepository)
saveOrder(Order):Order
deleteOrder(int):void
getAllOrders():List<Order>
-orderRepository
0..1
<<Java Interface>>
OrderRepository
com.proiect.Coffee_Shop.repository
<<Java Interface>>
OrderService
com.proiect.Coffee_Shop.service
saveOrder(Order):Order
deleteOrder(int):void
getAllOrders():List<Order>
-orderService
0..1
<<Java Class>>
OrderController
com.proiect.Coffee_Shop.controller
<<Java Class>>
ProductServiceImpl
com.proiect.Coffee_Shop.service
ProductServiceImpl(Produ...
saveProduct(Product):Pro...
0..1
<<Java Interface>>
ProductRepository
com.proiect.Coffee_Shop.repository
-productRepository
<<Java Interface>>
ProductService
com.proiect.Coffee_Shop.service
saveProduct(Product):Prod...
getAllProducts():List<Prod...
-productService
0..1
<<Java Class>>
ProductController
com.proiect.Coffee_Shop.controller
<<Java Class>>
UserServiceImpl
com.proiect.Coffee_Shop.service
-userRepository
0..1
<<Java Interface>>
UserRepository
com.proiect.Coffee_Shop.repository
<<Java Interface>>
UserService
com.proiect.Coffee_Shop.service
saveUser(User):User
getAllUsers():List<User>
checkUser(UserDto):boolean
-userService
0..1
<<Java Class>>
UserController
com.proiect.Coffee_Shop.controller
<<Java Class>>
SummaryServiceImpl
com.proiect.Coffee_Shop.service
SummaryServiceImpl(Summar...
~summaryRepository
0..1
<<Java Interface>>
SummaryRepository
com.proiect.Coffee_Shop.repository
<<Java Interface>>
SummaryService
com.proiect.Coffee_Shop.service
getById(int):Summary
setSummary(Summary,int):Summary
-summaryService
0..1
<<Java Class>>
SummaryController
com.proiect.Coffee_Shop.controller

5. Design Patterns

5.1 Hooks

Unul dintre design pattern-urile regăsite în acest proiect este un design pattern mai aparte, specific React și anume Hooks. În această aplicație am reușit să rezolvam problemele cu ajutorul anumitor hook-uri printre care UseContext

Având în vedere faptul că React-ul se bazează în mare parte pe componente, intervine problema de transmitere a datelor de la o componentă la alta, acest lucru se face de obicei printr-un parametru dat funcției principale a componentei, parametru numit props. Acesta poate fi apelat în alta componentă primind toate constantele și metodele din alta componentă. De multe ori datele trebuie să treacă prin mai multe componente pentru a putea ajunge la componenta dorita exemplu figura 1.

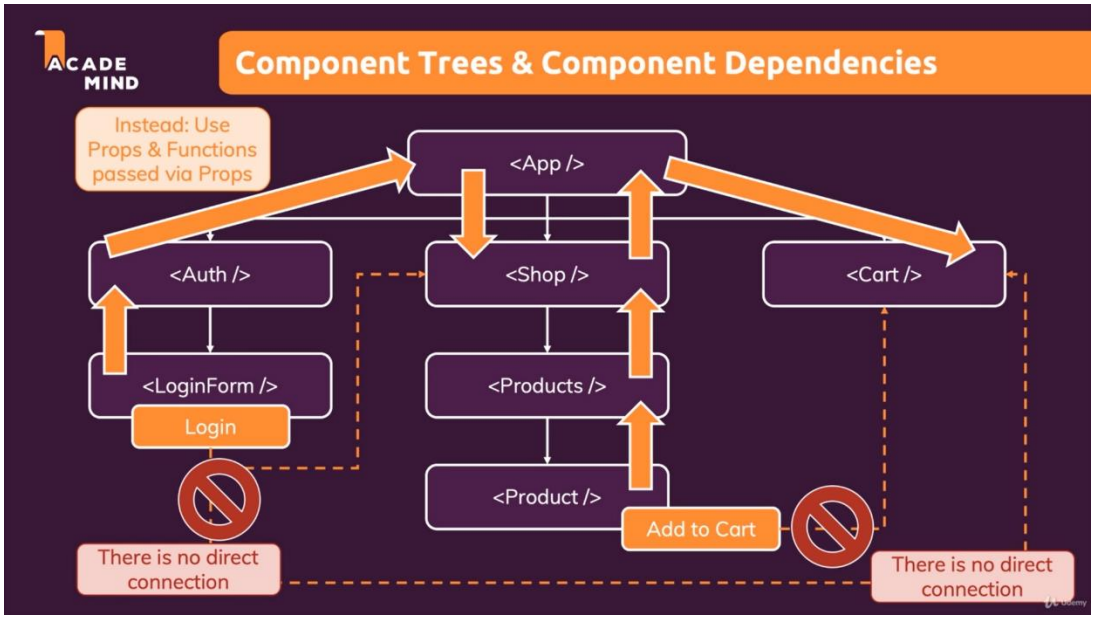


Figura 1 (sursa Udemy)

În figură se poate observa necesitatea de a ajunge informația introdusă de client de la login la shop, și de la adăugarea în coș la coșul de cumpărături. Dar nu există o legătură directă între aceste componente, iar săgețile portocali indică drumul pe unde trebuie chemat parametrul props din componentă în componentă pentru a ajunge la destinație. Această problemă poate fi rezolvată prin UseContext. UseContext creează o componentă ”globală” la care vin datele de la componenta care trimite și pleacă datele la componenta care trebuie să primească, fără a trece prin mai multe componente. Figura 2 arată cum funcționează useContext.

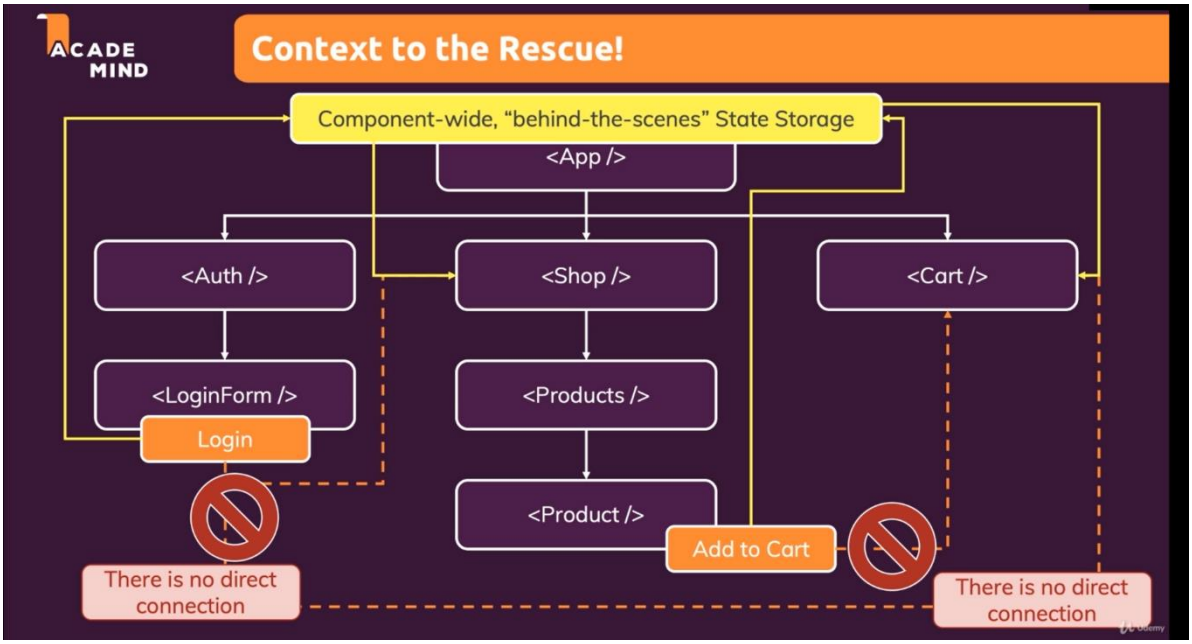
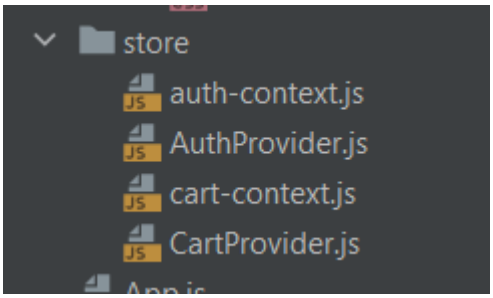


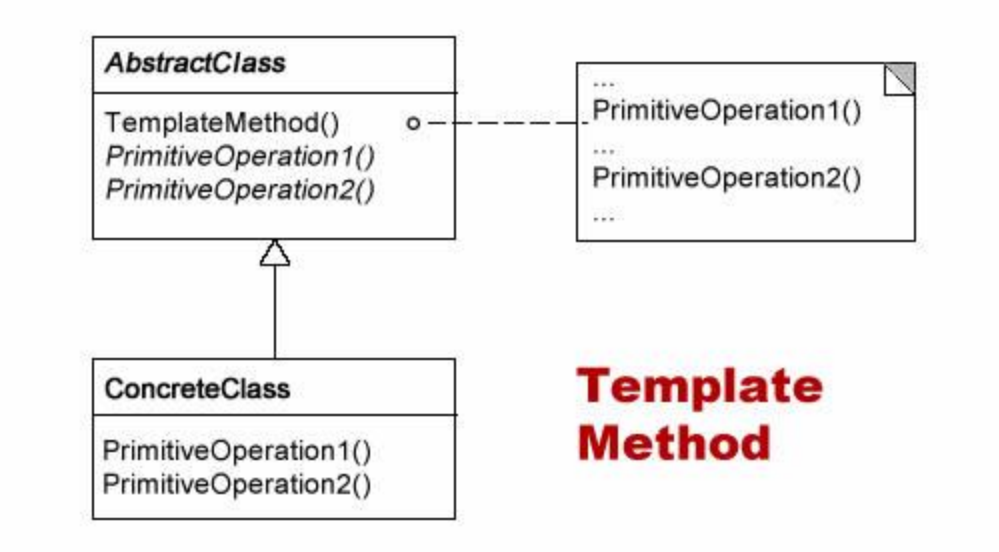
Figura 2 (sursa Udemy)

În aplicația noastră useContext este folosit de două ori, odată pentru partea de log in și odată pentru partea de coș de cumpărături. Pentru fiecare dintre situații au fost create două componente cart-context, respectiv auth-context și provider-ele.



5.2 Template

În această aplicație se utilizează, de asemenea, și Template design pattern. Acestă metodă presupune reutilizarea unui 'schelet', adică o clasă părinte definește anumite metode, care mai apoi sunt suprascrise de clasele care o extind pe aceasta. În Pixel Coffee Shop am extins clasa JPARepository care ne oferă metode pentru comunicarea cu baza de date. Am folosit aceleași metode de selecție, editare și ștergere atât pentru produse cât și pentru utilizatori și comenzi.

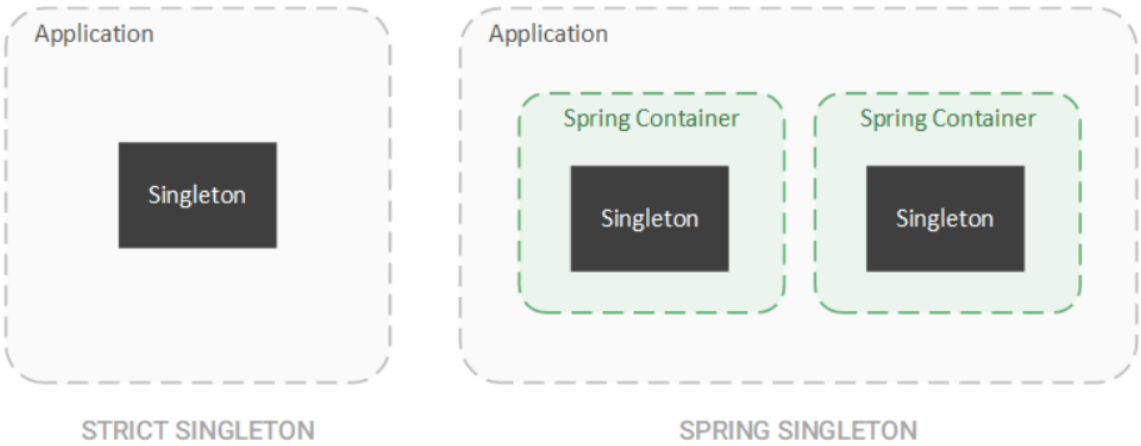


5.3 Autowired Singleton

Ultimul design pattern regăsit în acest proiect este cel de Autowired Singleton. Singleton pattern este un mecanism care asigură existența unei singure instanțe al unui obiect per aplicație.

Acest pattern este de folos atunci când lucrăm cu resurse partajate sau când furnizăm servicii transversale,cum ar fi logarea. În general, singleton este unic per aplicație, dar în Spring această constrângere este relaxată.

Spring restricționează o singură instanță pentru un anumite obiect per un container spring.



În aplicație, putem observa că pentru comenzi, produse și utilizatori se folosește acest pattern. Pentru fiecare se face un bean unic, astfel se păstrează corectitudinea informațiilor din baza de date.

6. Aplicații utilizate

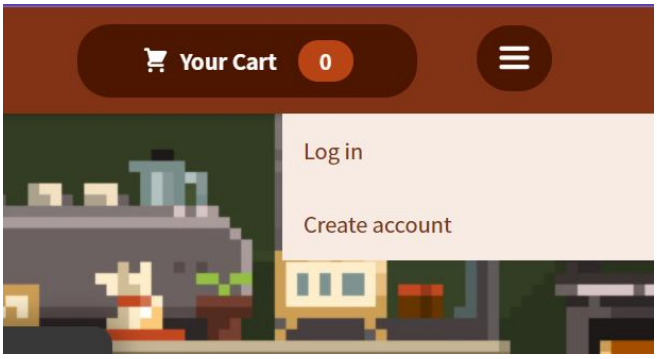
Aplicația de față a fost implementată utilizând mai multe medii de programare. Mai întâi am implementat baza aplicației utilizând Java Spring Boot în IntelliJ, aici au fost implementate clasele și interfețele necesare pentru a se crea tabele și pentru a se lucra cu datele din tabele. Cu ajutorul MySQL-ului am implementat tabelele si am adăugat date în tabele, iar cu ajutorul site-ului Postman (https://www.postman.com/) am reușit să testăm dacă funcțiile făcute de noi în Spring funcționează pentru lucrul cu tabelele.

Pe partea de frontend am utilizat React Java Script și am folosit mediul de programare WebStorm. Aici am creat un proiect nou unde cu ajutorul mai multor componente am reușit să dăm o forma site-ului pentru a fii ușor de utilizat de către client și pentru a avea un aspect frumos (să arate ca un site de cumpărături).

7. Mod de utilizare a aplicației

În pachet sunt disponibile două arhive, una ce conține aplicația pe frontend (Coffee_Shop_front) și una ce conține aplicația pe backend și scriptul pentru baza de date (Coffee_Shop_back). Pentru ca aplicația sa funcționeze ambele ambele aplicații trebuie să fie pornite. Recomandăm pentru a da run la cele două părți să se utilizeze IntelliJ pentru back și respectiv WebStorm pe front.

Odată adăugate toate componentele și clasele, ar trebui să se adauge și datele din tabele, se pot adăuga date proprii utilizând MySQL sau postman, sau se pot adăuga datele din scriptul direct in MySQL. Odată ce este făcută și legătura la baza de date, se poate da run la aplicația de pe backend, iar apoi la aplicația de pe frontend. Se va deschide un server localhost unde va apărea site-ul.



Utilizarea site-ului pentru client:

Există mai multe funcționalități oferite clientului, acesta poate să se logeze sau sa își creeze un cont nou, poate să dea search la produse si să le adauge în cart, poate să trimită o comandă și în cele din urmă să de-a log out.

Utilizarea site-ului pentru admin:

Pe partea de admin, la care se poate ajunge doar dacă se introduc numele si parola ”admin” la log in, apar alte funcționalități. Un tabel unde poate să adauge, să editeze sau să șteargă produse, iar la orders un tabel cu comenzile primite pe care poate să le trimită sau să le șteargă. De asemenea pe pagina de home a administratorului, acesta poate să modifice descrierea print-un simplu click pe form-ul cu descrierea.

8. Atribuțiile membrilor grupului

Fiecare membru a grupului a avut diferite atribuții, am încercat să ne împărțim egal, dar ne-am și ajutat reciproc, am colaborat pentru a reuși să facem tot ce ne-am propus la proiect.

Balog Helga s-a ocupat de partea administratorului, cea cu produsele (adăugare, modificare, ștergere de produse și modificare descriere), atât pe frontend cât și pe backend, a implementat diagrama de secvență și a scris despre autowired Singleton ca și design pattern.

Brebeanu Theodora a implementat partea de administrator comenzi (acceptare sau respingere comandă), atât pe frontend cât și pe backend, a implementat diagrama de stări și a scris despre Template ca și design pattern.

Dunca Denisa a implementat partea destinată clientului (log in, creare cont, trimitere comandă), atât pe frontend cât și pe backend, a implementat diagrama de activități și a scris despre Hooks (UseContext) ca și design pattern.

9. Bibliografie

https://app.diagrams.net/

https://www.udemy.com/course/react-the-complete-guide-incl-redux/

https://youtu.be/HE8jtK4FSZY

https://www.digitalocean.com/community/tutorials/how-to-add-login-authentication-to-react-applications

https://www.tutorialspoint.com/design_pattern/factory_pattern.htm

https://www.uxpin.com/studio/blog/react-design-patterns/

https://youtu.be/O_XL9oQ1_To?t=2444

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/