Smart Depot

Inginerie software

Autor: Serbanescu Narcis, Tarta Xenia si Timbuc Iulia
Grupa: 30237

Facultatea de Automatica
si Calculatoare

14 Ianuarie 2026

Cuprins

1 Introducere

Acest proiect are ca scop realizarea unei aplicatii web de tip e-commerce pentru comercializarea produselor electrocasnice. Aplicatia a fost dezvoltata pentru a simula functionarea unui magazin online real si pentru a oferi utilizatorilor posibilitatea de a cumpara produse intr-un mod rapid si usor.

In prezent, aplicatiile web de tip e-commerce sunt din ce in ce mai utilizate, deoarece permit utilizatorilor sa achizitioneze produse online fara a se deplasa fizic intr-un magazin. Acest proiect urmareste realizarea unei aplicatii moderne care sa imbine functionalitatile de baza ale unui magazin online cu tehnologii actuale din domeniul dezvoltarii web.

Scopul principal al aplicatiei este de a permite utilizatorilor sa vizualizeze produse, sa le adauge in cosul de cumparaturi si sa plaseze comenzi online. Aplicatia include si functionalitati precum autentificarea utilizatorilor, resetarea parolei prin email, plati online si alegerea metodei de livrare.

Aplicatia este dezvoltata ca un proiect full-stack, incluzand partea de frontend realizata cu React si partea de backend dezvoltata folosind Node.js si Express. De asemenea, este utilizata o baza de date pentru stocarea informatiilor, iar pentru imbunatatirea experientei utilizatorului este integrat un chatbot bazat pe inteligenta artificiala.

2 Prezentarea generala a aplicatiei

Aplicatia dezvoltata in cadrul acestui proiect este o aplicatie web de tip e-commerce, destinata comercializarii produselor electrocasnice. Aceasta a fost conceputa pentru a oferi utilizatorilor o experienta completa de cumparare online, similara cu cea a magazinelor online reale. Aplicatia se adreseaza tuturor categoriilor de varsta si poate fi utilizata atat de persoane cu experienta in mediul online, cat si de utilizatori aflati la primele interactiuni cu astfel de platforme.

Pentru a avea acces la functionalitatile aplicatiei, utilizatorul are posibilitatea de a-si crea un cont nou sau de a se autentifica intr-un cont existent. Procesul de autentificare este simplu si intuitiv. In situatia in care utilizatorul uita parola, aplicatia pune la dispozitie optiunea "Am uitat parola". In urma selectarii acestei optiuni, utilizatorul primeste pe email un cod de resetare, care este introdus ulterior in aplicatie actualizarea parolei. Acest mecanism contribuie la cresterea securitatii si la recuperarea rapida a accesului in cont.

Dupa autentificare, utilizatorul poate accesa pagina principala a aplicatiei, unde sunt afisate produsele disponibile. Fiecare produs poate fi vizualizat individual, avand o pagina dedicata care contine informatii detaliate, precum denumirea produsului, pretul, descrierea si recenziile altor utilizatori. Aplicatia permite cautarea produselor dupa mai multe criterii, precum nume, pret, produse reduse sau produse reambalate, facilitand astfel gasirea rapida a produselor dorite.

Un element important al aplicatiei il reprezinta sistemul de review-uri. Utilizatorii pot lasa recenzii pentru produsele achizitionate, oferind feedback si ajutand alti clienti sa ia decizii informate. Acest sistem contribuie la cresterea increderii in platforma si la imbunatatirea transparentei.

Pentru a imbunatati experienta utilizatorului, aplicatia include si un asistent virtual bazat pe inteligenta artificiala. Acesta are rolul de a oferi suport utilizatorilor, de a raspunde la intrebari legate de produse, comenzi sau procesul de cumparare si de a ghida utilizatorul pe parcursul utilizarii aplicatiei.

Produsele pot fi adaugate in cosul de cumparaturi, unde utilizatorul poate vizualiza lista produselor selectate, cantitatea acestora si pretul total. In momentul finalizarii comenzii, utilizatorul poate alege metoda de livrare, fie livrare la domiciliu, fie livrare la EasyBox. De asemenea, aplicatia ofera mai multe metode de plata, cum ar fi plata la livrare, plata cu cardul sau plata prin intermediul unor solutii de finantare.

Un element distinct al aplicatiei il reprezinta functionalitatea de finantare a comenzilor. Utilizatorii au posibilitatea de a achizitiona produsele in rate, prin completarea unei cereri de finantare direct din aplicatie. Aceasta cerere este inregistrata in sistem si procesata ulterior, permitand utilizatorului sa beneficieze de o metoda alternativa si flexibila de plata. Functionalitatea de finantare este integrata in fluxul de plasare comenzii si simuleaza comportamentul real al platformelor comerciale moderne care ofera optiuni de in rate.

Aplicatia dispune si de o pagina de profil dedicata fiecarui utilizator. Din aceasta sectiune, utilizatorul isi edita datele personale, precum nume, adresa sau date de contact. Tot aici, utilizatorul poate verifica daca detine un cont standard sau un cont premium. Contul premium poate fi obtinut la atingerea pragului de 10000 de puncte acumulate in urma comenzilor plasate. In plus, utilizatorul poate vizualiza istoricul comenzilor, poate anula o comanda, poate initia cereri de retur, poate urmari cererile de finantare si poate reseta parola direct din contul sau.

Prin toate aceste functionalitati, aplicatia ofera o solutie completa de e-commerce, punand accent pe usurinta in utilizare, securitate, flexibilitate in metodele de plata si imbunatatirea experientei utilizatorului.

3 Arhitectura aplicatiei

Aplicatia este construita folosind o arhitectura de tip client-server, in care partea de frontend si partea de backend sunt separate. Aceasta abordare este frecvent utilizata in dezvoltarea aplicatiilor web moderne, deoarece permite o organizare clara a componentelor, o mentenanta mai usoara si posibilitatea de extindere ulterioara a functionalitatilor. Separarea responsabilitatilor intre componentele aplicatiei contribuie la cresterea stabilitatii si la imbunatatirea experientei utilizatorului.

Frontend-ul reprezinta partea vizibila a aplicatiei si este responsabil de interactiunea directa cu utilizatorul. Prin intermediul interfetei grafice, utilizatorul poate naviga in aplicatie, poate vizualiza produsele disponibile, poate accesa detalii despre fiecare produs in parte, poate adauga produse in cosul de cumparaturi si poate plasa comenzi. De asemenea, frontend-ul ofera acces la functionalitati precum autentificarea utilizatorilor, gestionarea contului personal, vizualizarea comenzilor si interactiunea cu asistentul virtual. Comunicarea dintre frontend si backend se realizeaza prin intermediul cererilor HTTP, folosind API-uri de tip REST, ceea ce permite un schimb de date eficient si structurat.

Backend-ul aplicatiei gestioneaza logica principala a sistemului si proceseaza cererile primite de la frontend. Acesta actioneaza ca un intermediar intre interfata utilizatorului si baza de date, asigurand validarea datelor, securizarea accesului si corectitudinea operatiilor efectuate. Backend-ul se ocupa de autentificarea si autorizarea utilizatorilor, gestionarea produselor, a comenzilor si a cosului de cumparaturi, precum si de alte functionalitati esentiale pentru functionarea aplicatiei. Prin intermediul backend-ului sunt gestionate si operatiile sensibile, precum procesarea platilor sau trimiterea mesajelor prin email, astfel incat datele importante sa nu fie expuse direct in partea de frontend.

Un alt rol important al backend-ului este asigurarea securitatii aplicatiei. Acesta implementeaza mecanisme de autentificare si control al accesului, astfel incat doar utilizatorii autorizati sa poata accesa anumite functionalitati. De asemenea, backend-ul gestioneaza sesiunile utilizatorilor si protejeaza aplicatia impotriva accesului neautorizat. Prin centralizarea logicii in backend, aplicatia devine mai sigura si mai usor de controlat.

Baza de date este utilizata pentru stocarea persistenta a datelor aplicatiei. Aceasta contine informatii despre utilizatori, produse, comenzi, cosuri de cumparaturi si review-uri. Utilizarea unei baze de date permite pastrarea informatiilor pe termen lung si accesarea acestora ori de cate ori este necesar. Baza de date asigura consistenta si integritatea datelor pe parcursul utilizarii aplicatiei si permite realizarea operatiilor de cautare, filtrare si actualizare a informatiilor intr-un mod eficient.

Pe langa componentele principale, aplicatia integreaza si servicii externe, care contribuie la extinderea functionalitatilor sistemului. Serviciul de email este utilizat pentru trimiterea mesajelor de confirmare a comenzilor si pentru procesul de resetare a parolei. Serviciul de plati online permite utilizatorilor sa efectueze tranzactii cu cardul intr-un mod sigur, fara ca datele sensibile sa fie stocate direct in aplicatie. De asemenea, aplicatia integreaza un serviciu de inteligenta artificiala utilizat pentru implementarea asistentului virtual, care ofera suport utilizatorilor si raspunde la intrebari legate de produse si comenzi.

Toate aceste servicii externe sunt accesate prin intermediul backend-ului, ceea ce asigura un nivel ridicat de securitate si control asupra fluxului de date. Aceasta abordare permite izolarea componentelor externe si faciliteaza gestionarea eventualelor erori sau modificari ale serviciilor utilizate. In ansamblu, arhitectura aplicatiei este conceputa pentru a fi modulara, scalabila si usor de intretinut, oferind o baza solida pentru dezvoltari viitoare.

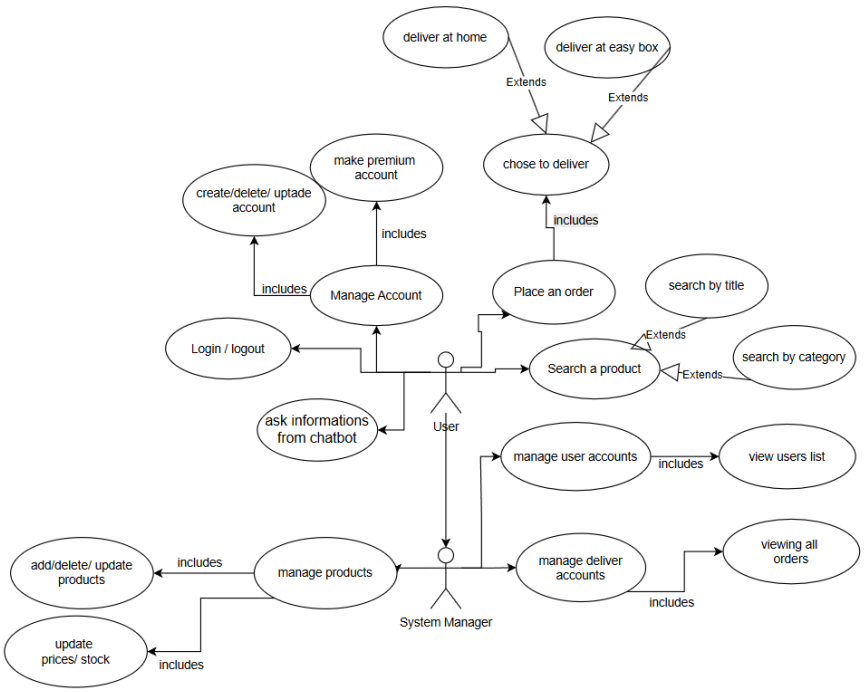

Figura 1: Diagrama Use Case a aplicatiei

Diagrama prezentata mai sus ilustreaza principalele functionalitati ale aplicatiei si interactiunea dintre utilizatori si sistem. Actorul User are acces la functionalitati precum autentificarea, cautarea produselor, plasarea comenzilor, alegerea metodei de livrare si utilizarea asistentului virtual. Acesta poate gestiona contul personal, poate vizualiza comenzile plasate si poate interactiona cu sistemul de review-uri. Actorul System Manager este responsabil de administrarea aplicatiei, avand acces la functionalitati precum gestionarea produselor, comenzilor si conturilor de utilizator. Relatiile de tip include si extend evidentiaza dependentele dintre diferitele actiuni ale sistemului si subliniaza modul in care functionalitatile sunt corelate intre ele.

4 Baza de date

Pentru stocarea si gestionarea datelor aplicatiei a fost utilizata o baza de date relationala, realizata folosind sistemul de gestiune a bazelor de date PostgreSQL. Alegerea PostgreSQL a fost facuta datorita fiabilitatii, performantei si suportului extins pentru aplicatii web moderne. Baza de date este gazduita in cloud folosind platforma Neon, ceea ce permite accesul facil la date, scalabilitate si o disponibilitate ridicata a sistemului.

Baza de date are rolul de a stoca informatiile esentiale necesare functionarii aplicatiei, precum datele utilizatorilor, produsele disponibile, comenzile plasate, cosurile de cumparaturi, review-urile si alte informatii auxiliare. Structura bazei de date este conceputa astfel incat sa permita o organizare clara a datelor si o relationare corecta intre diferitele entitati ale aplicatiei.

Tabela *users* este utilizata pentru stocarea informatiilor despre utilizatori. Aceasta contine date precum informatii de autentificare, date personale si rolul utilizatorului in cadrul aplicatiei. Rolurile permit diferentierea intre utilizatorii obisnuiti si administratorii sistemului.

Tabela *products* contine informatii despre produsele disponibile in magazin. In aceasta tabela sunt stocate date precum denumirea produsului, pretul, stocul, categoria si alte informatii necesare afisarii corecte a produselor in interfata aplicatiei.

Pentru gestionarea cosului de cumparaturi sunt utilizate tabelele *carts* si *cart_items*. Tabela *carts* este asociata unui utilizator, iar tabela *cart_items* contine produsele adaugate in cos, impreuna cu cantitatea acestora. Aceasta structura permite utilizatorilor sa adauge sau sa modifice produsele din cos inainte de finalizarea comenzii.

Gestionarea comenzilor este realizata prin intermediul tabelelor *orders* si *order_items*. Tabela *orders* stocheaza informatii generale despre fiecare comanda, precum utilizatorul care a plasat comanda, statusul acesteia si data plasarii. Tabela *order_items* contine produsele incluse in fiecare comanda, permitand asocierea mai multor produse unei singure comenzi.

Pentru functionalitatea de retur a comenzilor este utilizata tabela *order_returns*, care stocheaza informatii despre comenzile returnate si statusul acestora. Aceasta tabela permite gestionarea cererilor de retur intr-un mod organizat.

Aplicatia include si un sistem de recenzii, implementat prin tabela *reviews*. Aceasta permite utilizatorilor sa lase feedback pentru produsele achizitionate, contribuind la cresterea transparentei si la imbunatatirea experientei altor utilizatori.

Pentru functionalitatile avansate, aplicatia utilizeaza si tabele suplimentare. Tabela *chat_logs* este folosita pentru stocarea interactiunilor dintre utilizatori si asistentul virtual bazat pe inteligenta artificiala. De asemenea, tabela *financing_applications* este utilizata pentru gestionarea cererilor de finantare sau plata in rate, acolo unde aceasta optiune este disponibila.

Structura relationala a bazei de date permite mentinerea integritatii datelor si realizarea legaturilor dintre entitati prin chei primare si chei straine. Aceasta abordare asigura coerenta informatiilor si faciliteaza operatiile de cautare, actualizare si stergere a datelor. In ansamblu, baza de date reprezinta un element central al aplicatiei, asigurand functionarea corecta si eficienta a tuturor functionalitatilor implementate.

5 Backend – Node.js

5.1 Structura backend-ului

Backend-ul aplicatiei este organizat intr-o structura modulara, care permite separarea clara a si o mentenanta usoara a codului. Fiecare director din backend are un rol bine definit, iar clasele sunt grupate in functie de functionalitatea pe care o implementeaza.

Directorul *routes* defineste rutele API prin care frontend-ul comunica cu backend-ul. In acest director regasesc fisierele responsabile de gestionarea rutelor pentru autentificare, contul utilizatorului, produse, cosul de cumparaturi, comenzi, retururi, review-uri, chat, finantare si plati. Exista de asemenea rute dedicate pentru administratori, care permit gestionarea utilizatorilor, produselor, comenzilor si review-urilor. Fiecare ruta preia cererile HTTP si le directioneaza catre controllerul corespunzator.

Directorul *controllers* contine logica principala a aplicatiei. In acest director sunt implementate clasele care proceseaza cererile primite de la rute si interactioneaza cu baza de date sau cu serviciile externe. Controllerul de autentificare gestioneaza inregistrarea utilizatorilor, autentificarea si resetarea parolei. Controllerul de cont permite vizualizarea si actualizarea datelor utilizatorului, precum si gestionarea contului premium. Controllerul de produse se ocupa de listarea si cautarea produselor, iar controllerul de comenzi gestioneaza plasarea comenzilor si afisarea acestora. Exista controllere dedicate pentru cosul de cumparaturi, review-uri, retururi, chat si cereri de finantare. In plus, sunt implementate controllere separate pentru zona de administrare, care permit gestionarea completa a datelor aplicatiei.

Directorul *middleware* este utilizat pentru implementarea mecanismelor de securitate. In cadrul acestuia se regaseste clasa responsabila de verificarea autentificarii utilizatorilor si de validarea token-urilor de acces. Middleware-ul controleaza accesul la rute si asigura faptul ca doar utilizatorii autorizati pot accesa anumite functionalitati ale aplicatiei.

Directorul *auth* contine clasele responsabile de autentificare si autorizare. Clasa de baza defineste comportamentul general al procesului de autentificare, iar clasele concrete implementeaza metodele de autentificare folosind email si parola sau numar de telefon si parola. Selectarea metodei de autentificare este realizata prin intermediul unei clase de tip Factory, ceea ce permite extinderea usoara a sistemului cu metode suplimentare in viitor.

Directorul *db* este responsabil de gestionarea conexiunii cu baza de date PostgreSQL. In cadrul acestui director se afla clasa care realizeaza conexiunea la baza de date si gestioneaza pool-ul de conexiuni. Aceasta clasa este utilizata de celelalte componente ale aplicatiei pentru executarea interogarilor si accesul eficient la datele stocate.

Directorul *facades* este utilizat pentru abstractizarea proceselor mai complexe. In cadrul acestuia se regaseste o clasa dedicata procesului de checkout, care centralizeaza logica necesara plasarii unei comenzi, incluzand validarea datelor, procesarea platilor si salvarea informatiilor in baza de date. Utilizarea acestui tip de structura contribuie la pastrarea controllerelor simple si bine organizate.

Directorul *services* contine clasele care implementeaza logica reutilizabila si integrarea serviciilor externe. In acest director se regasesc clase pentru gestionarea comenzilor, platilor si generarea facturilor, precum si pentru trimiterea notificarilor prin email. De asemenea, sunt implementate clase pentru gestionarea produselor si utilizatorilor, precum si un builder pentru construirea obiectelor de tip produs. Aceasta separare permite reutilizarea codului si imbunatatirea organizarii aplicatiei.

Fisierul principal al aplicatiei reprezinta punctul de intrare al backend-ului, unde sunt initializate rutele, middleware-urile si configuratiile necesare pornirii serverului. Fisierul de configurare a mediului contine variabilele necesare conectarii la baza de date si integrarii serviciilor externe.

In ansamblu, structura backend-ului este conceputa pentru a sustine o aplicatie complexa de tip e-commerce, oferind o organizare clara, securitate si posibilitatea de extindere a functionalitatilor in viitor.

5.2 Autentificare si autorizare

Autentificarea si autorizarea utilizatorilor reprezinta componente esentiale ale backend-ului aplicatiei, avand rolul de a controla accesul la functionalitatile sistemului. Aplicatia implementeaza un mecanism complet de gestionare a identitatii utilizatorilor, care include crearea contului, autentificarea, resetarea parolei si controlul accesului in functie de rol.

Crearea contului de utilizator este realizata printr-un proces de inregistrare, in cadrul caruia sunt colectate datele necesare pentru identificarea utilizatorului. Informatiile introduse sunt validate si stocate in baza de date, iar parola este salvata intr-o forma criptata pentru a asigura securitatea datelor. Dupa crearea contului, utilizatorul poate accesa functionalitatile aplicatiei prin autentificare.

Autentificarea utilizatorilor permite accesul in aplicatie folosind datele de identificare. Sistemul suporta autentificarea atat prin email si parola, cat si prin numar de telefon si parola. Selectarea metodei de autentificare este realizata prin intermediul unui mecanism de tip Factory, care permite alegerea dinamica a strategiei de autentificare in functie de tipul solicitat. Aceasta abordare contribuie la o organizare clara a codului si la extinderea usoara a sistemului cu metode suplimentare de autentificare.

In continuare este prezentat un fragment de cod care ilustreaza utilizarea pattern-ului Factory pentru selectarea metodei de autentificare:

```
class AuthFactory {
    static create(type, userService) {
        switch (type) {
            case "email": return new EmailPasswordAuth(userService);
            case "phone": return new PhonePasswordAuth(userService);
            default: throw new Error("Tip␣de␣autentificare␣invalid");
        }
    }
}
```

Pentru situatiile in care utilizatorul uita parola, aplicatia ofera o functionalitate de resetare a parolei. Utilizatorul poate solicita resetarea parolei, iar sistemul genereaza un cod unic care este transmis prin email. Acest cod este utilizat pentru verificarea identitatii utilizatorului si pentru actualizarea parolei in conditii de siguranta.

Autorizarea accesului la resursele aplicatiei este realizata prin utilizarea token-urilor de autentificare. Dupa autentificare, utilizatorul primeste un token care este transmis la fiecare cerere ulterioara catre backend. Un middleware de securitate verifica validitatea token-ului si drepturile de acces ale utilizatorului. In functie de rolul utilizatorului, accesul la anumite rute este permis sau restrictionat, asigurand astfel protectia functionalitatilor sensibile ale aplicatiei.

5.3 Gestionarea utilizatorilor

Gestionarea utilizatorilor reprezinta o componenta importanta a backend-ului aplicatiei, fiind responsabila de administrarea informatiilor asociate conturilor de utilizator si de oferirea functionalitatilor necesare interactiunii acestora cu sistemul. Backend-ul permite gestionarea atat a utilizatorilor obisnuiti, cat si a utilizatorilor cu roluri speciale, precum administratorii.

Fiecare utilizator dispune de un profil personal, care contine informatii precum datele de identificare si datele de contact. Backend-ul permite vizualizarea si actualizarea acestor informatii, asigurand validarea datelor introduse. Modificarile efectuate sunt salvate in baza de date si reflectate ulterior in interfata aplicatiei.

Aplicatia utilizeaza un sistem de roluri pentru a diferentia tipurile de utilizatori. Utilizatorii obisnuiti au acces la functionalitatile standard ale aplicatiei, in timp ce administratorii beneficiaza de drepturi suplimentare, precum gestionarea utilizatorilor, produselor si comenzilor. De asemenea, aplicatia include un sistem de cont premium, care poate fi activat in urma acumularii unui numar prestabilit de puncte.

Punctele sunt acumulate de utilizatori in urma comenzilor plasate, iar acestea sunt utilizate pentru determinarea statusului contului. Backend-ul gestioneaza calculul si actualizarea punctelor, precum si activarea automata a contului premium atunci cand pragul necesar este atins.

Utilizatorii pot accesa istoricul comenzilor plasate, avand posibilitatea de a vizualiza detalii precum produsele comandate, statusul comenzii si data plasarii. In functie de starea comenzii, utilizatorii pot anula o comanda sau pot initia o cerere de retur. Aceste actiuni sunt procesate de backend si actualizate corespunzator in baza de date.

Pentru zona de administrare, backend-ul ofera functionalitati dedicate pentru gestionarea utilizatorilor. Administratorii pot vizualiza lista completa a utilizatorilor, pot modifica anumite informatii si pot gestiona rolurile acestora. Accesul la aceste operatii este restrictionat prin mecanisme de autorizare, asigurand securitatea sistemului.

5.4 Gestionarea produselor

Gestionarea produselor reprezinta una dintre functionalitatile centrale ale aplicatiei, avand rolul de a permite afisarea, cautarea si administrarea produselor electrocasnice disponibile in magazin. Backend-ul asigura accesul controlat la informatiile despre produse, in functie de tipul de utilizator care interactioneaza cu sistemul.

Pentru utilizatorii obisnuiti, aplicatia permite vizualizarea listei de produse disponibile si accesarea detaliilor fiecarui produs in parte. Fiecare produs este prezentat prin informatii relevante precum denumirea, pretul, descrierea, categoria, stocul disponibil si eventualele reduceri. Backend-ul permite cautarea si filtrarea produselor dupa diferite criterii, precum nume, pret, categorie, produse reduse sau produse reambalate, facilitand astfel identificarea rapida a produselor dorite.

Aplicatia include si un sistem de recenzii asociat produselor. Utilizatorii pot lasa review-uri pentru produsele achizitionate, oferind feedback bazat pe experienta proprie. Aceste recenzii sunt stocate in baza de date si sunt afisate in pagina produsului, contribuind la cresterea transparentei si la informarea altor utilizatori.

Din punct de vedere al implementarii backend-ului, produsele sunt gestionate prin intermediul unor controllere dedicate, care interactioneaza cu baza de date pentru preluarea si actualizarea informatiilor. Pentru construirea obiectelor de tip produs este utilizat un pattern de tip Builder, care permite crearea structurilor de date intr-un mod clar si flexibil. Aceasta abor-

dare contribuie la lizibilitatea codului si la separarea logicii de constructie a obiectelor de restul aplicatiei.

Gestionarea produselor in zona de administrare este realizata prin functionalitati dedicate, accesibile doar utilizatorilor cu rol de administrator. Administratorii pot adauga produse noi, pot modifica informatiile existente si pot sterge produse din sistem. De asemenea, acestia pot actualiza stocurile si preturile produselor, asigurand mentinerea informatiilor corecte in aplicatie. Accesul la aceste operatii este restrictionat prin mecanisme de autorizare, pentru a proteja integritatea datelor.

Prin implementarea acestor functionalitati, backend-ul asigura un control complet asupra ciclului de viata al produselor, de la afisarea acestora catre utilizatori pana la administrarea lor in sistem.

5.5 Gestionarea cosului si comenzilor

Gestionarea cosului de cumparaturi si a comenzilor reprezinta un flux esential al aplicatiei de tip e-commerce, permitand utilizatorilor sa selecteze produse, sa finalizeze achizitia si sa urmareasca comenzile plasate. Backend-ul este responsabil de procesarea acestui flux si de mentinerea consistentei datelor pe parcursul intregului proces.

Cosul de cumparaturi permite utilizatorilor autentificati sa adauge produse, sa modifice cantitatea acestora sau sa elimine produse din cos. Informatiile din cos sunt gestionate de backend si sunt asociate contului de utilizator, astfel incat continutul cosului sa fie pastrat intre sesiunile de utilizare. Backend-ul valideaza disponibilitatea produselor si actualizeaza valorile totale in functie de cantitatile selectate.

Pentru gestionarea informatiilor despre produse utilizate in cos si comenzi, backend-ul utilizeaza un pattern de tip Builder, care permite construirea obiectelor intr-un mod clar si flexibil. Urmatorul fragment de cod ilustreaza modul in care este creat un obiect de tip produs folosind acest pattern:

```
const product = new ProductBuilder()
    .setId(row.id)
    .setTitle(row.title)
    .setPriceCents(row.price_cents)
    .setStock(row.stock)
    .build();
```

In momentul plasarii comenzii, datele din cosul de cumparaturi sunt procesate de backend pentru a crea o comanda noua in sistem. Procesul de checkout presupune validarea produselor, calcularea pretului final si salvarea informatiilor relevante in baza de date. Odata finalizata, comanda este asociata utilizatorului si primeste un status initial, care reflecta starea acesteia.

Utilizatorii pot accesa istoricul comenzilor plasate, avand posibilitatea de a vizualiza detalii precum produsele comandate, valoarea totala, metoda de livrare si statusul comenzii. In functie de starea comenzii, utilizatorul poate anula comanda sau poate initia o cerere de retur. Aceste actiuni sunt procesate de backend si actualizate corespunzator in baza de date.

Aplicatia include si un mecanism de gestionare a returilor, care permite utilizatorilor sa solicite returnarea unei comenzi sau a unor produse. Cererile de retur sunt inregistrate in sistem si au un status specific, permitand urmarirea acestora. Backend-ul asigura corelarea returilor cu comenzile corespunzatoare si mentine integritatea datelor.

Din punct de vedere al implementarii, logica de gestionare a cosului si a comenzilor este organizata in controllere si servicii dedicate, care interactioneaza cu baza de date pentru stocarea si preluarea informatiilor. Pentru procesele mai complexe, precum finalizarea comenzii,

este utilizata o structura de tip Facade, care centralizeaza operatiile necesare si contribuie la mentinerea unei arhitecturi clare.

Prin aceste functionalitati, aplicatia ofera un flux complet de achizitie, de la selectarea produselor pana la gestionarea comenzilor si a returilor, asigurand o experienta coerenta si controlata pentru utilizator.

5.6 Plati si finantare

Platile si finantarea reprezinta o componenta esentiala a aplicatiei de tip e-commerce, fiind integrate in procesul de finalizare a comenzilor. Backend-ul este responsabil de gestionarea metodelor de plata disponibile, de procesarea tranzactiilor si de inregistrarea informatiilor corespunzatoare in baza de date, asigurand securitatea si consistenta operatiilor efectuate.

Aplicatia ofera utilizatorilor mai multe metode de plata. Plata la livrare permite finalizarea comenzii fara efectuarea unei tranzactii online, aceasta fiind inregistrata direct in sistem cu un status specific. Plata cu cardul este realizata prin integrarea unui serviciu extern de procesare a platilor, backend-ul avand rolul de a initia sesiunea de plata si de a primi confirmarea acesteia.

Procesul de checkout este gestionat prin intermediul unei structuri de tip Facade, care centralizeaza logica necesara plasarii unei comenzi. Aceasta structura ascunde complexitatea operatiilor interne, precum validarea datelor, calculul totalului, persistenta comenzii, procesarea platii si trimiterea notificarilor. Astfel, controllerul care apeleaza procesul de checkout interactioneaza cu o singura metoda, fara a gestiona detaliile interne.

In continuare este prezentat un fragment de cod care ilustreaza modul in care este initiata plata in functie de metoda selectata:

```
if (paymentMethod === 'card') {
    const paymentResult = await PaymentService.createCheckoutSession({
        orderId,
        email,
        items,
        totalCents
    });
    return { status: 'pending_payment', paymentUrl: paymentResult.
       sessionUrl };
} else {
    return { status: 'placed', message: 'Plata la livrare' };
}
```

Confirmarea platii online este realizata ulterior, in urma notificarii primite de la serviciul de plata. In acest moment, backend-ul actualizeaza statusul comenzii si salveaza informatiile asociate tranzactiei. Dupa confirmarea platii, utilizatorul primeste un email de confirmare, impreuna cu factura aferenta comenzii.

Pe langa metodele clasice de plata, aplicatia include si o functionalitate de finantare a comenzilor. Utilizatorii pot opta pentru achizitionarea produselor in rate, prin completarea unei cereri de finantare in cadrul procesului de comanda. Aceasta cerere este salvata in baza de date si asociata contului utilizatorului.

In continuare este prezentat un fragment de cod care ilustreaza inregistrarea unei cereri de finantare:

```
await pool.query(
    `INSERT INTO financing_applications (user_id, amount, months, status
       )
     VALUES ($1, $2, $3, 'pending')`,
```

```
    [userId, amount, months]
);
```

Cererile de finantare pot fi consultate de utilizator din contul personal, iar statusul acestora poate fi actualizat ulterior. Aceasta functionalitate simuleaza comportamentul real al platformelor comerciale care ofera optiuni flexibile de plata.

Prin integrarea platilor online, a platii la livrare si a sistemului de finantare, aplicatia ofera un flux complet si realist de achizitie, asigurand utilizatorilor flexibilitate si siguranta in procesul de cumparare.

5.7 Servicii auxiliare

Pe langa functionalitatile principale ale aplicatiei, backend-ul integreaza o serie de servicii auxiliare care contribuie la automatizarea proceselor si la imbunatatirea experientei utilizatorului. Aceste servicii sunt utilizate pentru trimiterea notificarilor prin email, generarea facturilor in format PDF si procesarea platilor online prin intermediul unor servicii externe.

5.7.1 Serviciul de notificari prin email

Trimiterea notificarilor prin email este realizata prin intermediul unui serviciu dedicat, responsabil de comunicarea dintre sistem si utilizatori. Acest serviciu este utilizat pentru confirmarea comenzilor, confirmarea platilor, transmiterea facturilor si notificarea utilizatorilor cu privire la statusul cererilor de retur.

Serviciul utilizeaza un transportor de email configurat prin variabile de mediu, asigurand separarea datelor sensibile de codul aplicatiei. In functie de tipul evenimentului, sunt generate mesaje personalizate care contin informatii relevante despre comanda sau actiunea efectuata.

Un exemplu de apel al serviciului de notificare este prezentat mai jos:

```
await NotificationService.sendOrderConfirmationWithInvoice(
    email,orderId, total, items, address, invoicePath );
```

Acest mecanism permite trimiterea automata a emailurilor fara a afecta fluxul principal al aplicatiei, eventualele erori fiind gestionate intern.

5.7.2 Generarea facturilor PDF

Pentru comenzile platite online, aplicatia genereaza automat o factura in format PDF, care este atasata emailului de confirmare. Generarea facturii este realizata prin intermediul unui serviciu dedicat, care construieste documentul PDF pe baza informatiilor comenzii.

Factura contine datele clientului, lista produselor comandate, valorile financiare si informatii despre comanda. Fisierele generate sunt salvate temporar pe server si sunt sterse dupa trimiterea emailului, pentru a evita ocuparea inutila a spatiului de stocare.

Fragmentul de cod de mai jos ilustreaza generarea unei facturi:

```
const invoicePath = await InvoiceService.generateInvoice({
    orderId, customerName, email, address, items, totalCents});
```

Aceasta abordare permite automatizarea completa a procesului de facturare, fara interventie manuala.

5.7.3 Procesarea platilor online

Platile online sunt gestionate prin intermediul unui serviciu dedicat de procesare a platilor, care integreaza o platforma externa de tip Stripe. Acest serviciu este responsabil de initierea sesiunilor de plata, verificarea statusului tranzactiilor si procesarea evenimentelor primite prin webhook.

In cadrul procesului de checkout, backend-ul creeaza o sesiune de plata care este utilizata de utilizator pentru finalizarea tranzactiei. Dupa confirmarea platii, informatiile sunt transmise catre sistem pentru actualizarea statusului comenzii.

Un exemplu de initiere a unei sesiuni de plata este prezentat mai jos:

```
const session = await PaymentService.createCheckoutSession({
    orderId, email, items, totalCents });
```

Serviciul gestioneaza si evenimentele de tip webhook, care permit confirmarea automata a platilor si declansarea actiunilor ulterioare, precum trimiterea emailului de confirmare sau generarea facturii.

5.7.4 Rambursari si tratarea evenimentelor de plata

In cazul retururilor sau al situatiilor speciale, aplicatia permite efectuarea rambursarilor pentru comenzile platite online. Acest proces este realizat prin intermediul serviciului de plati, care interactioneaza cu platforma externa pentru initierea refund-ului.

Fragmentul de cod de mai jos ilustreaza crearea unei rambursari:

```
await PaymentService.createRefund(paymentIntentId,amountCents);
```

Prin utilizarea acestui mecanism, aplicatia asigura un flux complet de gestionare a platilor, de la initierea tranzactiei pana la eventualele rambursari.

Prin integrarea serviciilor auxiliare, backend-ul reuseste sa automatizeze procesele critice si sa mentina o arhitectura modulara, in care fiecare serviciu are un rol bine definit si poate fi extins sau modificat independent.