



**Academia de Studii Economice din București**  
**Facultatea de Cibernetică, Statistică și Informatică Economică**  
**Specializarea: Informatică Economică**

# **Dezvoltarea unei aplicații pentru gestionarea stocurilor**

---

**LUCRARE DE LICENȚĂ**

Coordonator științific:

**Prof. Dr. Adela BÂRA**

Absolvent:

**Eduard-Gabriel CONSTANTIN**

**București**

**2019**

# CUPRINS

<b>INTRODUCERE .....</b>	<b>2</b>
<b>1.    DESCRIEREA PROBLEMEI ECONOMICE.....</b>	<b>4</b>
1.1.    PREZENTAREA DOMENIULUI ABORDAT .....	4
1.2.    PREZENTAREA ACTIVITĂȚII CARE VA FI INFORMATIZATĂ .....	5
1.3.    ANALIZA SOLUȚIILOR EXISTENTE PE PIAȚĂ.....	10
<b>2.    ANALIZA SISTEMULUI INFORMATIC .....</b>	<b>16</b>
2.1.    SPECIFICAREA CERINȚELOR SISTEMULUI INFORMATIC.....	16
2.2.    ANALIZA SISTEMULUI EXISTENT .....	19
<b>3.    PROIECTAREA SISTEMULUI INFORMATIC.....</b>	<b>26</b>
3.1.    PROIECTAREA NOULUI SISTEM .....	26
3.2.    PROIECTAREA SCHEMEI BAZEI DE DATE.....	28
<b>4.    IMPLEMENTAREA SISTEMULUI INFORMATIC .....</b>	<b>31</b>
4.1.    TEHNOLOGII INFORMATICE UTILIZATE .....	31
4.2.    PREZENTAREA APLICAȚIEI .....	38
<b>CONCLUZII.....</b>	<b>43</b>
<b>BIBLIOGRAFIE .....</b>	<b>44</b>

# Introducere

Încă din cele mai vechi timpuri umanitatea a simțit nevoia evoluției și a dezvoltării, a progresului și a noutății în viața oamenilor. De-a lungul anilor, noi ca întreg, am trăit și experimentat diferite invenții, fie ele mai mult sau mai puțin reușite. De la tancuri-avion<sup>[2]</sup> până la mașini care puteau să se deplaseze pe apă și pe uscat în același timp<sup>[2]</sup>, omul le-a încercat pe toate și a învățat câte puțin din fiecare greșeală a sa... Mai puțin când vine vorba de războaie, acolo se pare că nu va învăța niciodată... Dar să nu intrăm în politică și să ne întoarcem la subiectul nostru.

Invențiile au fost acele idei ambițioase, care la început păreau poate prea mărețe pentru perioada apriției lor, care aveau parte de critici și erau privite cu scepticism, mai ales de cei conservatori, ce nu își doreau neapărat o schimbare și erau mulțumiți cu zona lor de confort. Multe dintre lucrurile pe care în ziua de astăzi le considerăm obișuite și le acceptăm ca atare aveau la vremea lor să fie asupra judecate chiar și de experți. Printre acestea putem aminti:

- **Becul electric** al lui Thomas Edison, despre care Parlamentul Britanic spunea că va fi “un eșec garantat” și că “poate fi destul de bun pentru prietenii lor de peste Atlantic, dar nu și pentru oamenii lor de știință.”<sup>[3]</sup>;
- **Telefonul** lui Graham Bell, despre care Western Union, pe atunci cea mai mare companie de comunicații, afirma că “este cu greu mai mult decât o jucărie și acel dispozitiv nu este util în mod inerent.”<sup>[3]</sup>;
- **Avionul** fraților Wright, pe care mareșarul francez Ferdinand Foch îl considera “o jucărie interesantă, dar fără nicio valoare militară”. Opt ani mai târziu avea să fie folosit în Primul Război Mondial<sup>[4]</sup>;
- Iar ceva mai recent, **shopping-ul online**, despre care revista Time scria în 1966 un articol controversat, în care considera că “este complet posibil, dar va eșua, pentru că femeilor le place să iasă din casă, le place să simtă produsul, le place să se răzgândească”<sup>[4]</sup>.

Evident că afirmațiile de mai sus aveau să fie complet eronate, toate invențiile enumerate devenind în același timp un standard și o necesitate în societatea din vremurile noastre. Poate singura excepție de la această regulă o face ultima, și anume cumpărăturile la distanță.

În timp ce reprezintă un mare avantaj tehnologic de care dispunem față de generațiile trecute, momentan nu a devenit cu totul necesar, căci putem conviețui foarte bine și fără existența acestuia, deoarece încă avem la îndemână varianta fizică a aceluiași serviciu. Putem oricând să ne urcăm în mașină și să mergem într-un supermarket sau un magazin specializat, fie că vorbim de electronice, haine sau alte consumabile, într-un dealer auto să testăm o mașină, într-o agenție de turism să luăm bilete de avion și așa mai departe. Iar toate acestea sunt posibile pentru că industria comerțului și a vânzărilor online este una în plină dezvoltare, care nu a reușit pentru moment să satisfacă întreaga piață și care așteaptă noi idei și invenții pentru a face acest pas.

Astfel, mi-am propus să dau o mână de ajutor în acest sens și prin prezenta lucrare de licență să rezolv micile probleme și impedimente întâlnite pe piață sau cel puțin să încerc să vin cu o nouă alternativă la soluțiile deja existente. Cu siguranță nu va fi vorba de o invenție revoluționară și nu îmi propun să schimb roata, ci doar să îmbin anumite aspecte deja existente în cadrul aceluiași mediu, care să vizeze un public țintă cât mai vast.

Evident, pentru a face acest lucru posibil voi utiliza în implementarea aplicației tehnologii de ultimă generație, care să suporte diverse dispozitive, cu diverse specificații și sisteme de operare. Design-ul trebuie să fie și el plăcut și utilizatorul să aibă o experiență cât mai bună pentru a reveni în cadrul aplicației, deci îmi propun să pun accent și pe partea de Front-End și UI/UX Design (User Interface, respectiv User Experience).

# 1. Descrierea problemei economice

## 1.1. Prezentarea domeniului abordat

Pentru a putea înțelege mai bine necesitățile domeniului abordat va trebui mai întâi să ne întoarcem puțin în trecut, ca să vedem ce a determinat apariția vânzărilor la scară mare, prin supermarketuri și bineînțeles, ce stă la baza conceptului de supermarket.

Ei bine, supermarketul este unul dintre lucrurile pe care societatea le acceptă fără a-și pune prea multe semne de întrebare, și deși îi lipsește oarecum farmecul unei mici afaceri specializată pe o anumită zonă de produse alimentare, cum ar fi o brutărie, măcelarie, sau un chioșc cu fructe, reprezintă de departe cea mai populară metodă de cumpărare pentru consumatori, datorită prețurilor mai mici, gamei mai mari de produse și a unei locații care îmbină "totul într-unu". Supermarketurile au ajuns atât de populare și de comune încât a devenit greu să ne imaginăm că la un moment dat acest concept nu exista. Deci, cum a apărut ideea unui magazin multifuncțional?

În urmă cu 100 de ani, pentru a-și face cumpărăturile, oamenii ar fi apelat fie la un magazin specializat, precum cele menționate mai devreme sau ar fi avut varianta un magazin de vânzare cu amănuntul (retail), unde erau nevoiți să listeze articolele pe care le doresc înainte ca un asistent să le colecteze din depozit pentru aceștia. Acest lucru avea să se schimbe în 1916, când Clarence Sanders, un vânzător local din Memphis(SUA) introducea o modalitate mai eficientă pentru afacerea sa și crea conceptul de „magazin tip autoservire”<sup>[5]</sup>.

Conceptul de cumpărături pe bază de autoservire a prins repede pe atunci, iar supermarketul a fost declarat responsabil pentru declanșarea puterii de marcă, consumatorii putând din acel moment să se identifice din ce în ce mai mult cu brandurile și companiile care fabricau produse. Toată această „explozie” a avut loc pentru că până atunci nimeni nu se mai confruntase cu posibilitatea de a alege produsul dorit și în magazinele de tip retail clientul nu vedea numele companiei fabricante.

Pentru Sanders supermarketul a fost primul dintre mai multe concepte cu care a fost patentat, cum ar fi "keedoozle" - o versiune mai mare a tonomatelor din zilele noastre sau "foodelectric" – un serviciu de checkout care avea să se răspândească și să devină un standard chiar și acum, la aproape un secol distanță. Tocmai din aceste considerente, Sanders este văzut acum ca un personaj care era cu mult înainte de timpul să, prin acțiunile și inițiativele sale, dar mai ales prin modul de gândire<sup>[6]</sup>.

## 1.2. Prezentarea activității care va fi informatizată

Dacă până acum am vorbit despre online shopping și supermarketuri, a venit momentul să îmbinăm cele 2 concepte, prin găsirea unei soluții ce digitalizează procesul cumpărăturilor și bineînțeles, îl eficientizează în beneficiul consumatorului. Iar de aceea trebuie să ne punem întrebarea **“Cum pot magazinele și supermarketurile să creeze experiențe mai bune și mai personalizate pentru clienții lor?”**

Având în vedere că industria alimentară se reinventează pentru a răspunde obiceiurilor de cumpărături foarte schimbătoare și a concurenței acerbe, întâlnim tot mai multe schimbări și abordări la nivelul agenților de vânzare, care încearcă să atragă cât mai mulți clienți noi și să păstreze în același timp și o mare parte dintre cei vechi. Una dintre aceste schimbări o reprezintă introducerea aplicațiilor, prin care magazinele vor să ofere o valoare mai mare consumatorilor lor, fie că vine ca o alternativă sau un adaos la serviciile deja existente. De fapt, peste 50% dintre comercianți se află fie în procesul stabilirii sau deja în plină desfășurare a unei strategii care combină experiențele online, digitale și în magazine<sup>[7]</sup>.

Aplicațiile joacă un rol esențial în aceste noi abordări multi-canal. Doar în ultimul an, numărul clienților care au început să folosească astfel de “asistenți” la cumpărături s-a dublat și trendul pare să se păstreze și în continuare, dacă nu cumva chiar să escaladeze. Cumpărătorii insistă asupra experiențelor oferite de comercianți și așteaptă din ce în ce mai mult în zona digitală, dar din păcate, multe dintre aplicațiile magazinelor de astăzi nu oferă o valoare reală pentru aceștia.

Din punctul de vedere al unui cumpărător pasionat de tehnologie, am analizat soluțiile deja existente pe piață și am încercat să înțeleg ce nevoi nu sunt satisfăcute deloc sau poate doar parțial, ce mai poate fi îmbunătățit și ce ar revoluționa cu totul această zonă digital, încă prea puțin explorată.

**În primul rând, este nevoie de timp pentru a înțelege cumpărătorii și competiția.** Multe companii care lansează aplicații nu iau în considerare nevoile unice ale utilizatorilor lor. Câștigarea loialității acestora nu este obținută printr-un card sau un program de reduceri, ci se bazează pe adresarea nevoilor cumpărătorilor mai mult decât o fac concurenții. Pentru a îndeplini acest obiectiv, trebuie să aflăm interesele cumpărătorilor, ce fel de experiență de cumpărături doresc și de ce ar putea să se adreseze altor magazine. De aceea efectuarea unei cercetări ample a utilizatorilor și analiza competitivă este de departe cel mai important pas înainte de a începe dezvoltarea.

Potrivit unui raport recent al Food Marketing Institute, cumpărătorii prioritizează "prețurile bune, produsele de calitate, magazinele curate, serviciul pentru clienți și liniile scurte de plată."<sup>[5]</sup> O aplicație de succes va lucra pentru a satisface aceste nevoi fie pe deplin în cadrul aplicației, sau prin intermediul întregii experiențe de cumpărături.

Acum, înainte de a începe proiectarea trebuie să răspundem la câteva întrebări, pentru a ne face o mai bună părere asupra percepției și așteptărilor cumpărătorilor:

- Cum acționează în prezent cumpărătorii cu produsele din magazin sau în mediul digital?
- Cât de des vizitează magazinul și la ce ore/ zile ale săptămânii?
- Sotesc cu liste de cumpărături sau planifică în magazin?
- Ce deviceuri și tehnologii folosesc?
- Ce își doresc cumpărătorii în prezent și nu li se oferă?

Abia după ce am stabilit aceste aspecte putem trece mai departe, la **perfecționarea caracteristicilor din cadrul aplicației.**

- **Căutare și stoc**

Pentru a crea o experiență bună pentru cumpărători, aceștia trebuie să găsească rapid și ușor ceea ce caută. Având în vedere numărul și gama de articole disponibile, nu este deloc o misiune ușoară, dar o modalitate de a realiza acest lucru este optimizarea structurii de căutare și a căutării aplicației. Gruparea elementelor din stoc într-o ordine logică și utilizarea fotografiilor sau a iconițelor pentru a facilita misiunea cumpărătorilor reprezintă un mare plus și fac, de asemenea, produsele mai atractive. Mai mult decât atât, opțiunea vizualizării stocurilor pentru un articol este esențială, mai ales la magazinele din preajma cumpărătorului.

- **Salvarea și sincronizarea listelor de cumpărături**

Cumpărătorii vor putea crea și utiliza liste de cumpărături pentru multe scenarii diferite. Unul dintre cazurile cele mai neglijate de utilizare în aplicațiile magazinelor este opțiunea de a salva liste de cumpărături sau de a afișa articole achiziționate frecvent. Acest lucru permite cumpărătorilor care tind să cumpere aceleași lucruri să își creeze rapid listele de cumpărături.

Adăugarea funcției de sincronizare la caracteristica listei de cumpărături este o modalitate puternică de a răspunde așteptărilor cumpărătorului. Acest lucru permite mai multor utilizatori să acceseze și să actualizeze aceeași listă, eliminând suprapunerea și confuzia. O altă modalitate de a utiliza sincronizarea listei de cumpărături este prin sincronizarea site-ului și aplicației. Acest lucru permite unui utilizator să creeze o listă de cumpărături pe site, în timp ce acasă, apoi să o acceseze direct de pe dispozitivul său, în momentul în care face cumpărăturile.

Pe lângă aceste scenarii, aplicația ar trebui să includă o modalitate eficientă de a bifa toate articolele din listă. Poate părea evident, dar furnizarea de informații importante despre locație, cum ar poziția raioanelor, duce la simplificarea călătoriilor de cumpărături.



O altă funcție utilă ar fi vizualizarea prețului total intermediar și a celui estimat pentru elementele de pe listele lor de cumpărături.

- **Navigarea în magazin și scanarea codurilor de bare**

Un alt pas evident pentru o experiență mai bună a utilizatorilor este oferirea de locații pentru produse. Cu toate acestea, funcția de localizare poate fi dusă la următorul nivel și să ofere o vizualizare pe hartă pentru a oferi cumpărătorilor informații mai detaliate despre magazin. Ceea ce ar face-o cu totul remarcabilă ar fi implementarea unui mini GPS sau a unui configurator de traseu, pentru a determina cea mai eficientă cale de a bifa toate produsele dorite.

Unele magazine oferă și posibilitatea scanării codurilor de bare, caracteristică prin care cumpărătorii pot verifica prețuri, recenzii și informații mai generale despre produsele scanate. Dacă toate aceste funcții ar putea fi integrate în cadrul aceleiași aplicații ar facilita și mai mult experiența de shopping și ar ajuta cumpărătorii înainte de luarea unei decizii.

- **Checkout rapid și mobil**

Unul dintre dezavantajele cumpărăturilor din supermarket este timpul petrecut în așteptare, la coadă. Dincolo de tehnologia de auto-checkout, care a început să fie implementată în ultimii ani în toate lanțurile mari de magazine, portofelele mobile merg cu un pas și mai departe în accelerarea acestui proces. Unii comercianți iau în considerare funcțiile de scanare, în timp ce le permit cumpărătorilor să depășească în întregime cozile.

Pentru a realiza totuși în mod corect acest aspect, aplicația trebuie să integreze și opțiuni de plată rapide precum PayPal, Apple Pay sau Google Pay, care să poată fi folosite la toate liniile de plată, inclusiv serviciul de self-service și casierul. În același timp pot interveni dificultăți și restricții cu privire la permisiunile primite și protecția datelor personale.

- **Înțelegerea cazurilor de utilizare și a coerenței între platforme**

Pentru început, este important să înțelegem cum vor utiliza cumpărătorii componentele aplicației noastre, fie web, mobile sau în timp real în magazin. Trebuie să răspundem din nou la câteva întrebări, precum dacă utilizatorii își vor crea liste de cumpărături de acasă sau de la serviciu, dacă vor accesa aplicația pentru comandarea și preluarea în magazin sau o vor folosi pentru a localiza și a verifica prețul elementelor din coloane și așa mai departe.

Cumpărătorii ar trebui să înțeleagă beneficiile utilizării fiecărei platforme și să se miște ușor între ele pentru a-și atinge obiectivele. De exemplu, un cumpărător creează o listă de cumpărături acasă și se întoarce la ea în timp ce se află în magazin sau poate plasa o comandă prin intermediul site-ului și primește o notificare atunci când produsele sale sunt pregătite pentru ridicare.

- **Eliminarea barierelor de intrare**

Aplicația trebuie să fie simplă și să nu pună obstacole sau bariere utilizatorilor. Multe aplicații au sisteme de înregistrare complicate și prea multe impedimente vor confunda și vor stârni cumpărătorii, împiedicându-i să le mai utilizeze și în continuare.

Nu toate funcțiile trebuie să fie neapărat disponibile din prima, dar utilizatorii se pot înscrie în schimbul unor avantaje, ceea ce face procesul de înregistrare mai atractiv și mai puțin anevoios. Apoi, este important ca întregul proces să fie simplificat, iar utilizatorii să se poată conecta ușor, prin intermediul programelor sau conturilor deja existente, precum Facebook sau Google.

- **Personalizarea experienței cumpărătorului**

Personalizarea face cumpărătorii să simtă că nevoile lor unice sunt îndeplinite. În loc de o abordare unică pentru toți, aplicațiile magazinelor ar trebui să includă funcții personalizate îi fac pe utilizatori să se simtă ca și cum sunt înțelese nevoile și dorințele. Modalitățile prin care acest lucru poate fi realizat sunt:

- Capacitatea aplicației de a învăța modelul de cumpărături al unui utilizator;
- Oferirea unor oferte speciale pe baza obiceiurilor de cumpărături ale utilizatorilor;
- Sugerarea de produse relevante, de sezon și a articolelor achiziționate frecvent;
- Promovarea ofertelor relevante în timpul deplasării prin magazin;
- Maparea unui traseu în funcție de locațiile produselor din lista de cumpărături.

Aceste tactici nu numai că facilitează cumpărăturile clienților, ci ajută și la creșterea încrederii și a loialității acestora, ceea ce duce în final la mai multe cumpărături.

### • **Distrație prin cumpărături**

Există o serie de strategii care au fost și pot fi în continuare utilizate pentru a introduce elemente de joc în cardul unor astfel de aplicații. Aceasta ar putea recompensa cumpărătorii pentru vizitarea frecventă sau completarea coșurilor de cumpărături cu un anumit număr sau tip de articole, plasarea unor termene limită pentru cumpărături, mai multă implicare pe partea de social media și crearea unei comunități, care să vină constant cu feedback, prin care aplicația să poată fi îmbunătățită ulterior.

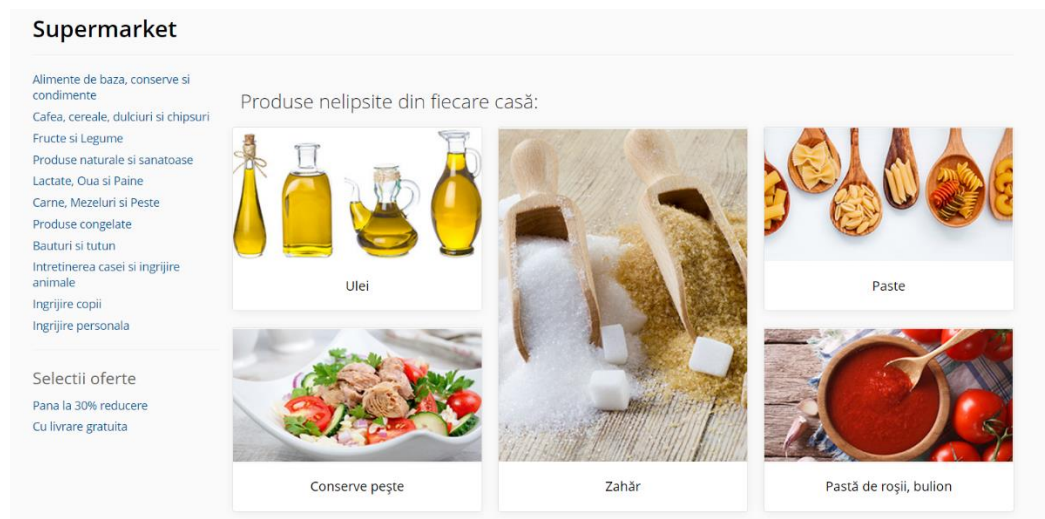
Există practic o oportunitate enormă de care supermarketurile și magazinele de profil nu au reușit să profite la maximum până acum, dar de care poate încă profita oricine dacă își setează o strategie corectă și analizează îndeaproape atât comportamentul clienților, cât și al concurenților. Bineînțeles, după asta urmează dezvoltarea efectivă și testarea ulterioară, dar fără un start bun și o direcție precisă toate planurile vor eșua la fel ca până acum.

## 1.3. Analiza soluțiilor existente pe piață

Este evident faptul că alternative există momentan, atât la nivel local, cât și global, fie că vorbim de aplicații și soluții specializate în domeniul cumpărăturilor și al supermarket-urilor sau variante de uz general, ce pot fi folosite în mai multe contexte.

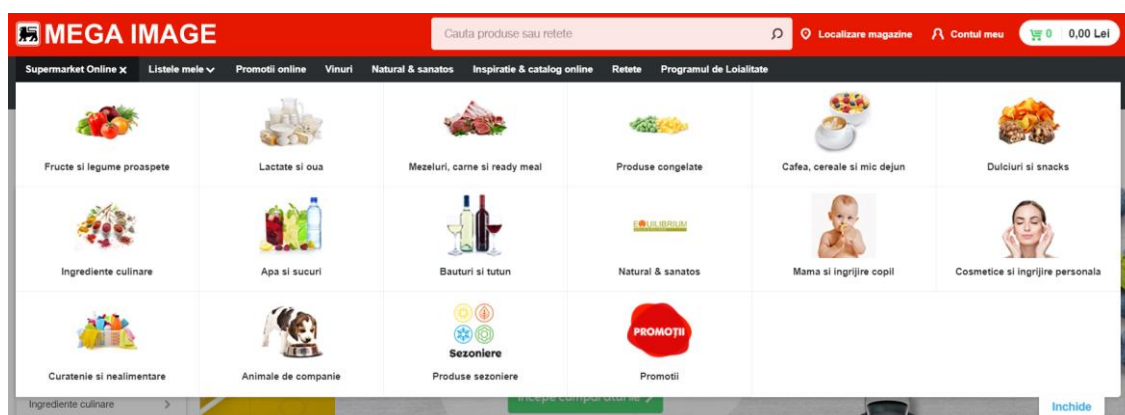
Astfel, pe plan local putem distinge următoarele soluții:

- **Aplicațiile dedicate ale lanțurilor mari de magazine sau ale comercianților cu amănuntul (retaileri)** – acestea păstrează în mare aceleași funcționalități de bază, la care se adaugă diferite alte utilități, în funcție de specificul fiecărui magazin în parte.



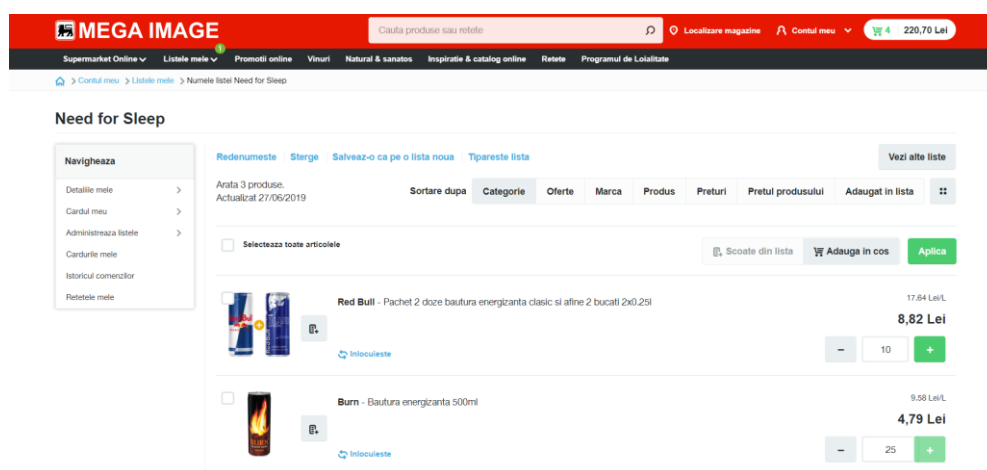
**Figura 1.3.1** Interfața categoriei Supermarket din catalogul eMAG

Spre exemplu, în figura de mai sus se regăsește pagina “de aterizare” a unui comerciant online din România. Deși acesta nu deține magazine fizice, oferă pe pagina sa o alternativă rapidă și ușoară a cumpărăturilor de zi cu zi, conferind posibilitatea plasării de comenzi online.



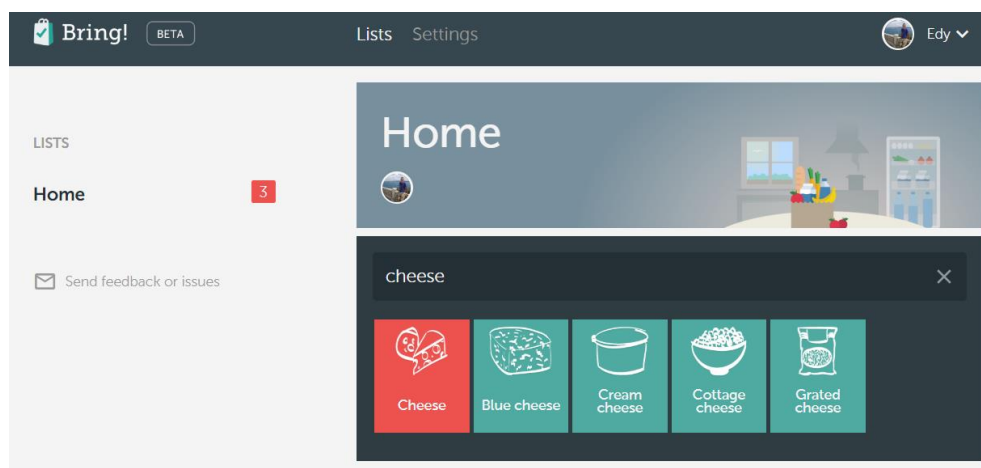
**Figura 1.3.2** Meniu cu categoriile disponibile din catalogul online Mega Image

Similar, un lanț mare de magazine din București și din țară oferă o alternativă digitală la deja existentă variantă a cumpărăturilor fizice și adună în cadrul aceluiași site date actualizate în timp real cu privire la propriile produse. De curând a fost introdusă și posibilitatea creării de liste personalizate, pe baza unui cont și a unui card de fidelitate, unde un client poate să își păstreze liste prestabilite și să vizualizeze istoricul cumpărăturilor sale, așa cum se poate observa în figura imediat următoare.



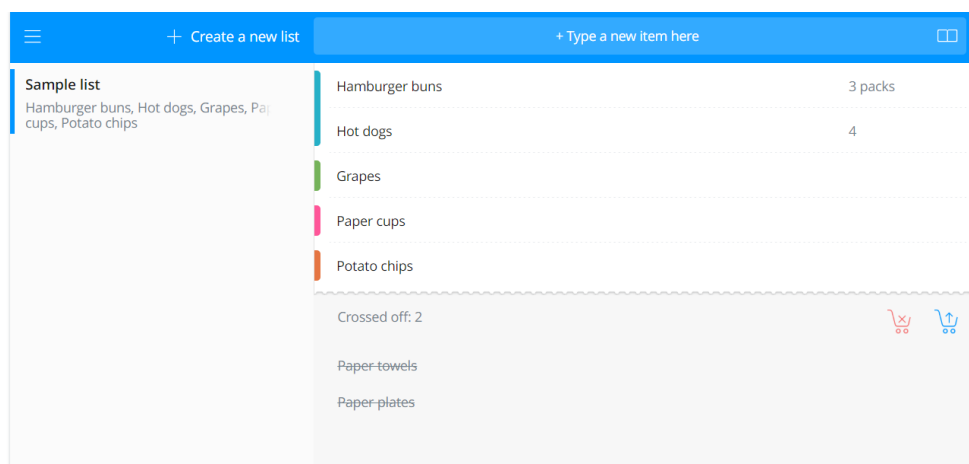
**Figura 1.3.3** Listă personalizată din cadrul site-ului *Mega Image*

- **Aplicații de sine stătătoare, ce au ca scop principal crearea de liste** – iar aici pot fi incluse o multitudine de astfel de instanțe, cu diferite design-uri și cazuri de utilizare pentru modificarea, editarea sau folosirea listelor create. De obicei acestea nu sunt conectate cu niciun supermarket sau serviciu, iar singura bază de date pe care o au în spate este cea cu produsele predefinite de către dezvoltatori.



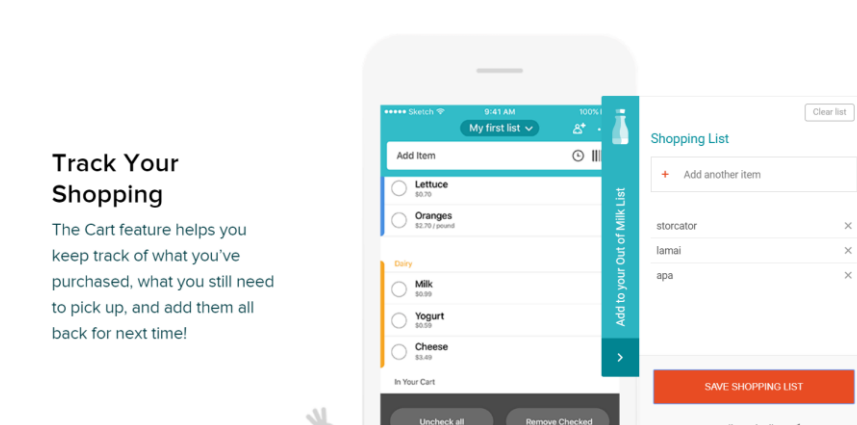
**Figura 1.3.4** Meniul de căutare al aplicației *Bring!*

Spre exemplu, în figura 1.3.4 se regăsește o parte din interfața aplicației web Bring!, o aplicație cu un design minimalist, ce păstrează un minim necesar de funcționalități, dar în același timp oferă utilizatorului o experiență îndeajuns de bună pentru procesul de cumpărături. Avantajul său principal este disponibilitatea și co-integrarea mai multor platforme, astfel putând comuta destul de ușor între un dispozitiv mobil și un PC, de exemplu.



**Figura 1.3.5** Vizualizarea unei liste în aplicația *Buy Me A Pie!*

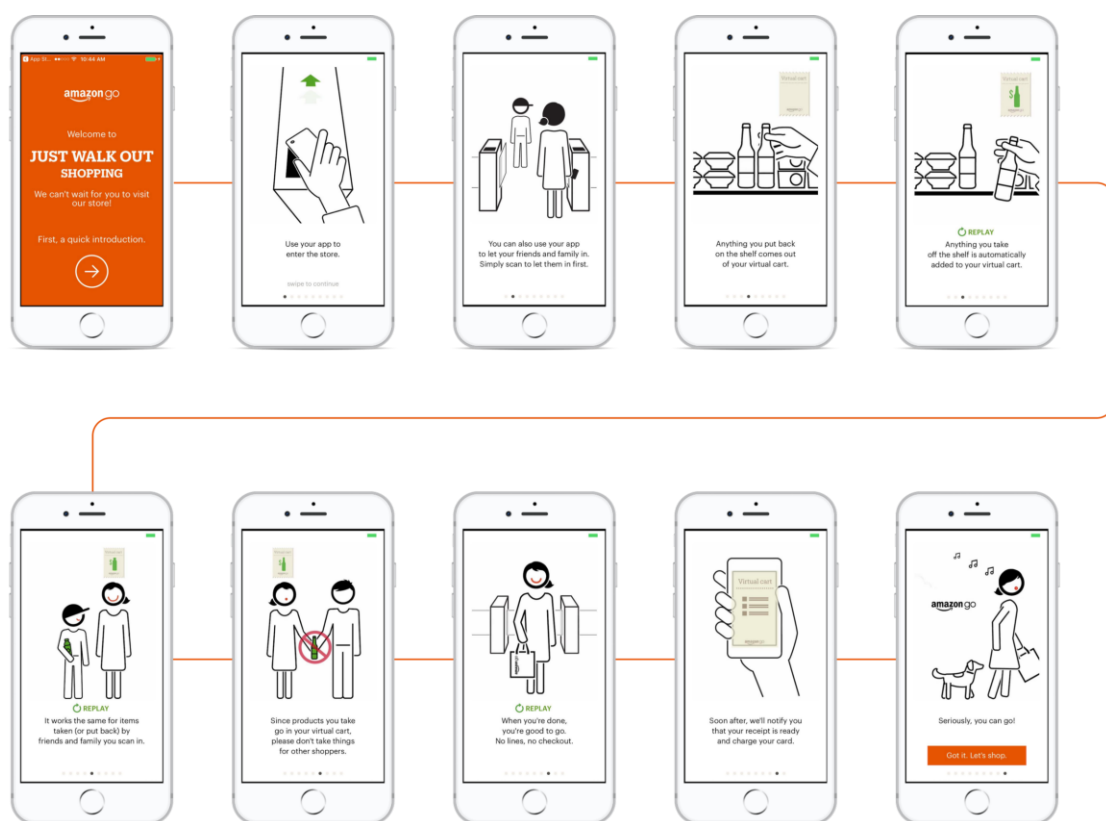
O altă aplicație de acest tip este Buy Me A Pie!, din nou o aplicație destul de simplă, dar cu câteva funcționalități interesante. Ca un mare avantaj față de alte soluții concurente, aceasta oferă posibilitatea de a crea și gestiona un număr mare de liste, mai departe în cadrul listelor putând fi adăugate produse care nu sunt stocate pe server, ci pur și simplu reprezintă datele de intrare definite de utilizator. În același timp, există și alternativa adăugării de produse cu caracter general, deja existente într-o bază de date, produse pe care aplicația le filtrează și le colorează specific categoriei din care fac parte.



**Figura 1.3.6** Crearea unei liste simple în aplicația *Out Of Milk*

Ultima aplicație de acest tip pe care am ales-o pentru prezentare poartă numele de “Out of Milk” sau în traducere “Am rămas fără lapte” și deși funcționalitățile principale pe care le oferă se regăsesc în aplicația Android dedicată, creatorii oferă și o alternativă web, ce-i drept, cu mai puține module implementate. Aceasta oferă posibilitatea de a adăuga, edita și șterge o singură listă de cumpărături, la final putând fi salvată sub două forme: fie ca o **copie locală**, în calculator, fie **sub forma unui email**, pe baza unei adrese de mail introdusă la momentul finalizării sale.

- **Aplicații deja integrate cu un sistem de componente hardware implementat în cadrul magazinului** – domeniu unde nu există foarte multă concurență momentan, din cauza dificultății pe care o conferă un astfel de ecosistem, dar spre care se pare că ne îndreptăm în viitorul nu foarte îndepărtat.



**Figura 1.3.7 Instrucțiuni pentru utilizarea Amazon Go**

După cum este foarte bine descrisă în figura de mai sus, experiența cumpărăturilor într-un astfel de magazin bazat pe autoservire este una pe cât de simplă, pe atât de complicată. Pentru

client au fost eliminate toate elementele consumatoare de timp și de resurse, cum ar fi scanarea tuturor produselor selectate pentru achiziționare sau durata necesară pentru realizarea plății și au fost înlocuite cu alte procese automatizate, ce au la bază o mulțime de camere și senzori, pentru detectarea în timp real a coșului de cumpărături pentru fiecare utilizator/client. La final nu este nevoie decât de scanarea unui cod unic personal, ce se găsește în aplicația dedicată, iar pe baza produselor selectate de la raft se autogenerează factura și se efectuează plata. Astfel, clientul are centralizate într-un singur loc toate datele și rapoartele necesare.<sup>[8]</sup>

Iar deși tehnologia din spate este cu adevărat uimitoare, există și un mare dezavantaj, cel puțin momentan, și anume disponibilitatea foarte mică a acestor tipuri de magazine și doar pe arii restrânse, în Statele Unite. Așadar, probabil o să mai dureze ceva timp până vor fi implementate alternative și la noi în țară, iar până atunci trebuie să eficientizăm procesul clasic de cumpărături, așa cum îl știm noi.



## 2. Analiza sistemului informatic

### 2.1. Specificarea cerințelor sistemului informatic

Scopul proiectului este realizarea unei aplicații informatice pentru gestiunea stocurilor și a produselor dintr-un magazin/depozit, ce oferă funcționalități atât pentru personal, cât și pentru clienți. Aceasta aduce un plus de utilitate ambelor părți și eficientizează atât procesul de cumpărare, cât și cel de inventar/gestiune a produselor oferite spre vânzare.

În primul rând, focusul principal se va orienta asupra clientului și cele mai multe funcționalități vor avea directă legătură cu acesta, întrucât aplicația dorește mai presus de toate transformarea procesului de cumpărături într-unul mai ușor și mai plăcut. Printre cerințele de bază care trebuie îndeplinite se numără:

- Posibilitatea de a crea liste de cumpărături, fie pe baza produselor dintr-un magazin, fie pe baza propriului input
- Posibilitatea distribuirii listelor mai departe, către alte persoane și alți utilizatori
- Posibilitatea de a crea filtre pentru categorii și preț atunci când caută un produs
- Posibilitatea de a ordona listele de cumpărături în funcție de locația produselor la raft, pentru a eficientiza traseul clientului în magazin
- Posibilitatea de a plasa din nou orice comandă făcută în trecut

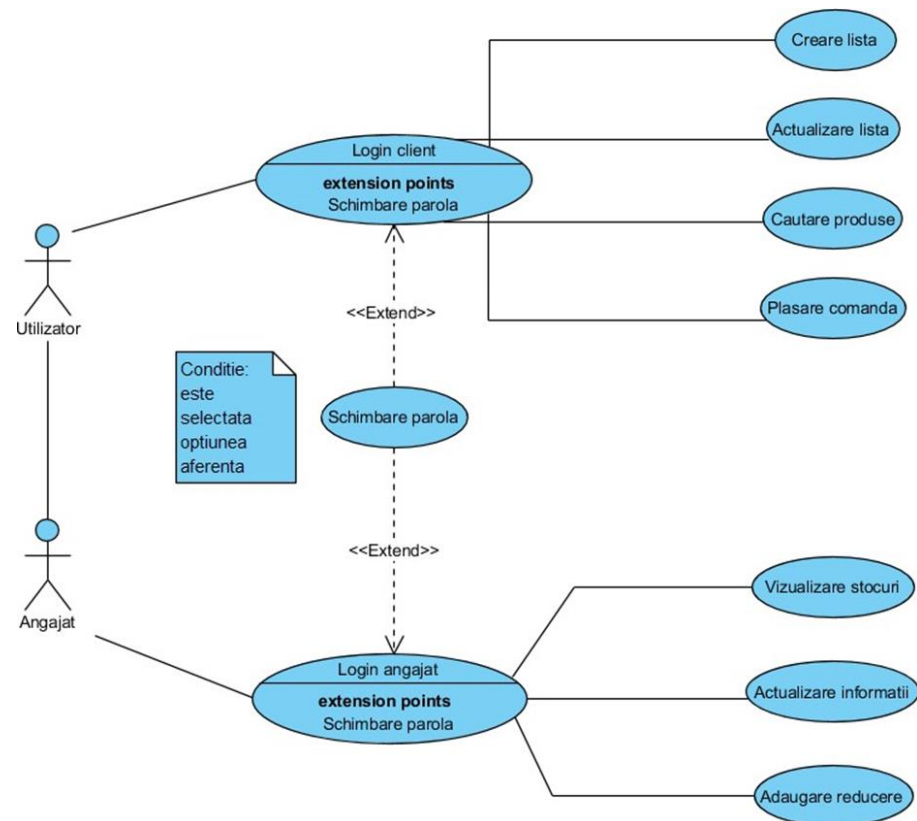
Pe de altă parte, există și funcționalitățile special dedicate angajaților și în general personalului, dintre care cele mai importante ar fi:

- Accesul la baza de date cu toate produsele specifice departamentului său
- Posibilitatea adăugării, ștergerii sau editării informațiilor unui produs
- Posibilitatea adăugării reducerilor

La modul general se va face o diferențiere a stocurilor între raft și depozit, cele de la raft actualizându-se automat pe măsură ce sunt cumpărate, iar cele din depozit, manual, de către personalul administrativ.

## Diagrame detaliate ale cazurilor de utilizare

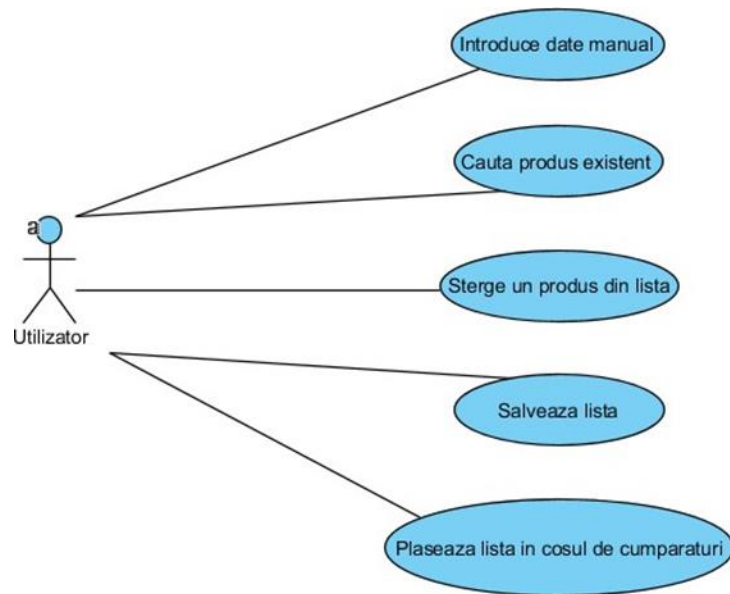
După ce am prezentat problemele ce ar trebui rezolvate și abordarea propusă, urmează ca prin cadrul diagramelor cazurilor de utilizare să fie reprezentate funcționalitățile sistemului informatic.



**Figura 2.1.1** Diagrama generală a cazurilor de utilizare

Diagrama de mai sus are ca scop principal gestionarea stocurilor dintr-un magazin, privită din ambele perspective, atât de client, cât și de angajat. Fluxul de bază include 6 pași, ce denotă funcțiile principale pe care le îndeplinește sistemul:

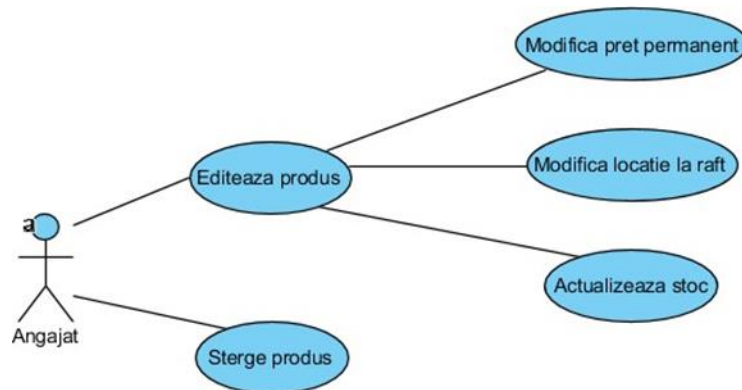
1. Utilizatorul se conectează [Curs alternativ: Utilizatorul a uitat parola]
2. Utilizatorul crează o listă nouă de cumpărături
3. Utilizatorul actualizează lista cu noi produse
4. Utilizatorul salvează lista
5. Utilizatorul plasează o comandă pe baza listei sau merge în magazinul fizic
6. Cumpărăturile sunt făcute cu succes



**Figura 2.1.2** Diagrama detaliată a cazului de utilizare “Actualizare listă de cumpărături”

Pentru cazul de utilizare de mai sus avem un singur actor principal, utilizatorul aplicației, iar scopul fluxului este actualizarea și gestiunea unei liste de cumpărături deja creată. Cei patru mari pași sunt:

1. Utilizatorul accesează lista
2. Utilizatorul adaugă sau șterge un produs
3. Utilizatorul salvează noua listă
4. Lista poate fi refolosită în viitor



**Figura 2.1.3** Diagrama detaliată a cazului de utilizare “Actualizare informații produs”

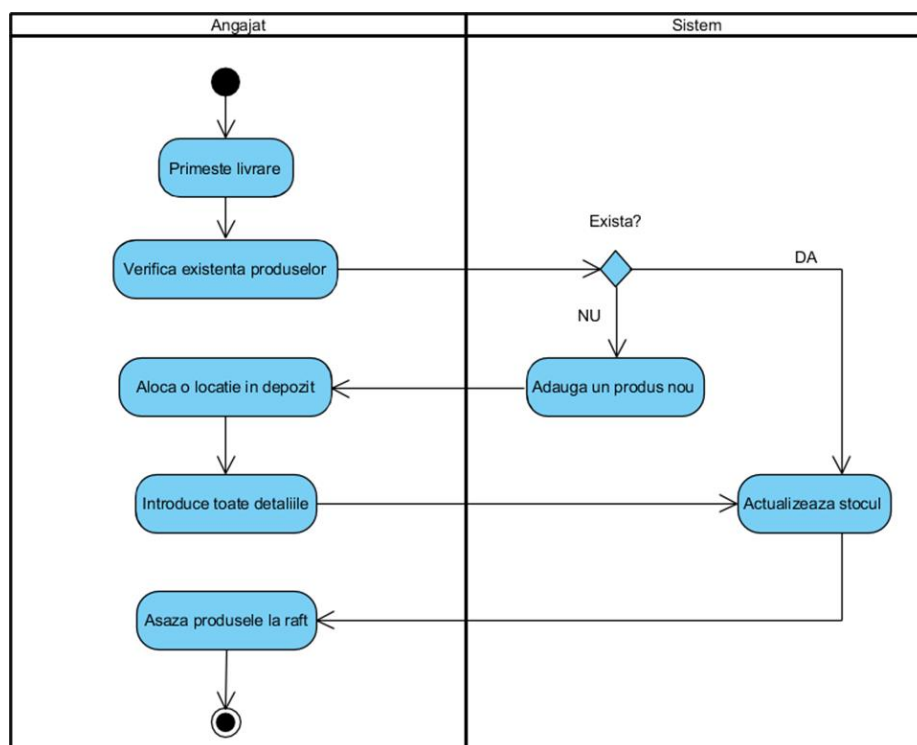
În cazul figurii 2.1.3, avem ca actor principal al cazului de utilizare un angajat, ce are ca scop actualizarea informațiilor cu privire la un produs. Fluxul complet presupune gestiunea tuturor informațiilor unui produs din partea angajatului, incluzând varianta ștergerii acelui produs din ofertă, iar organizarea pe pași ar fi următoarea:

1. Angajatul accesează detaliile despre produs
2. Angajatul modifică aceste detalii sau șterge produsul
3. Utilizatorul salvează noul produs, dacă este cazul
4. Produsul poate fi folosit și adăugat în listele clienților

## 2.2. Analiza sistemului existent

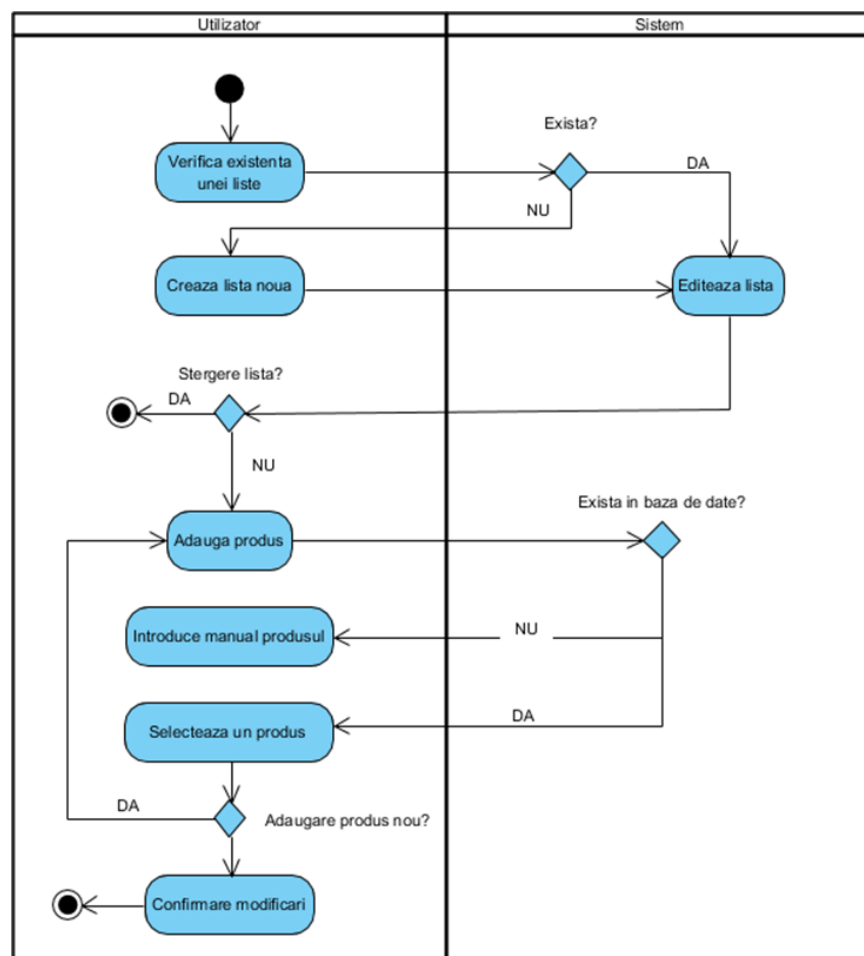
### Diagrame de activitate

Acestea ajută la reprezentarea vizuală a acțiunilor prin care se dorește obținerea unui rezultat și descrie fluxul procedural de lucru de la un nod de plecare până la un nod final venind cu detalii referitoare la căile de decizie pe care le oferă. <sup>[11]</sup>



**Figura 2.2.1** Diagrama de activitate pentru gestiunea mărfurilor

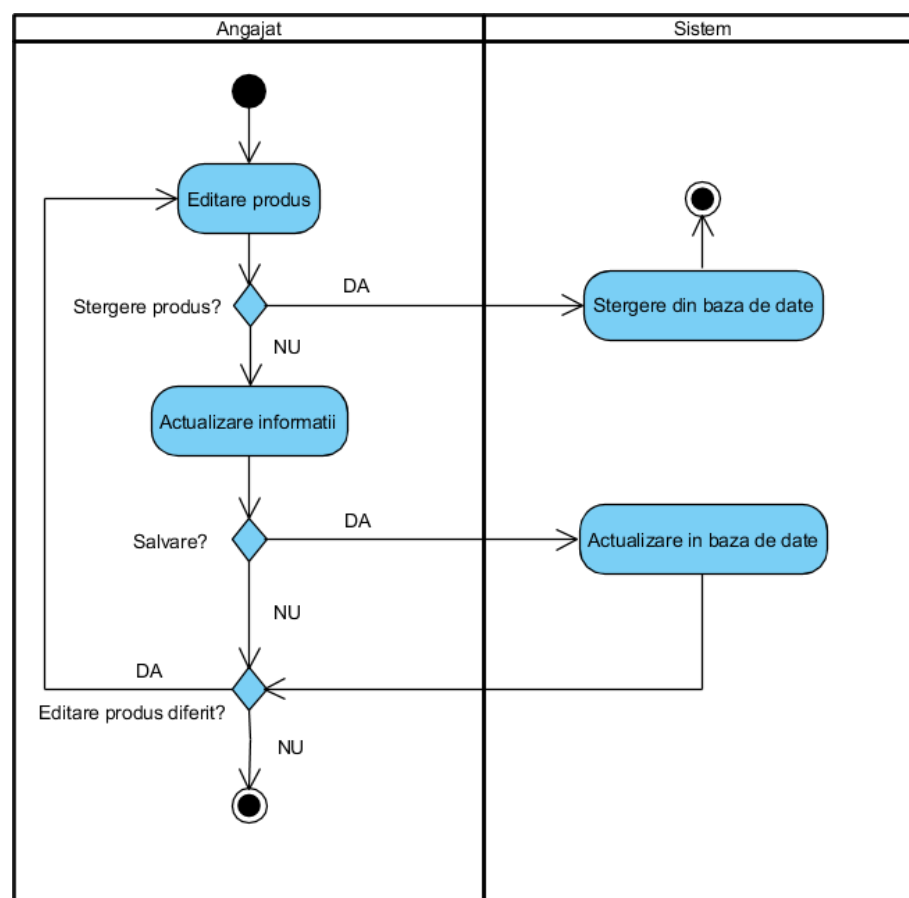
În figura precedentă este detaliată activitatea și schimbul de date ce se produce între cele două partiții principale, angajatul și sistemul, în momentul primirii unei livrări de produse. Angajatul este nevoit să verifice existența în sistem a fiecărui produs în parte, iar pe baza răspunsului la această întrebare fluxul continuă cu un nod decizional. În cazul în care există tot ce are de făcut în continuare este să actualizeze stocul din sistem, iar în caz contrar trebuie să mai întreprindă câteva activități în plus, ce constau în adăugarea produsului în baza de date, alocarea unei locații în depozit și completarea tuturor detaliilor. În final, procesul se încheie prin așezarea produsului la raft.



**Figura 2.2.2** Diagrama de activitate pentru editarea unei liste de cumpărături

Figura 2.2.2 descrie de această dată interacțiunea dintre utilizator și sistem, în vederea editării sau actualizării unei liste de cumpărături. Procesul începe prin verificarea existenței unei liste și la fel ca în cazul anterior, intervine un nod de decizie, pe baza căruia fie este direct editată

lista, în cazul existenței, fie este mai întâi creată și mai apoi editată, în caz contrar. De asemenea, clientul are și posibilitatea ștergerii listei în acest punct, situație în care intervine un nod final și fluxul se încheie. Altfel, acesta continuă prin adăugarea de noi produse, fie introduse manual, sub formă textuală, fie selectate din baza de date. Acesta este un proces repetitiv și se încheie prin confirmarea modificărilor, după adăugarea tuturor produselor dorite.

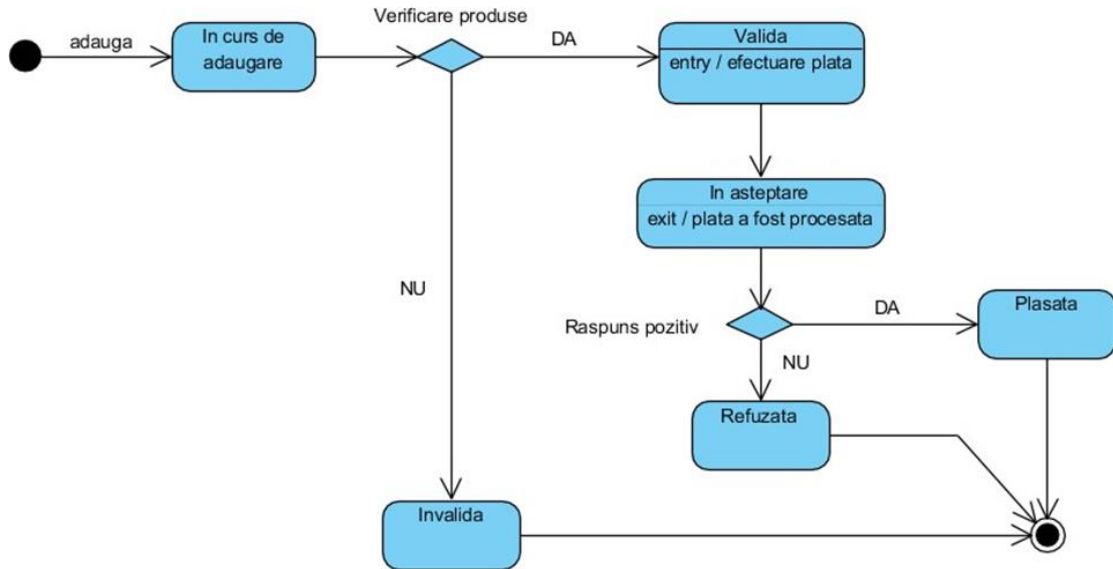


**Figura 2.2.3** Diagrama de activitate pentru actualizarea informațiilor unui produs

Ultima activitate descrisă îl readuce în prim plan pe angajat și ilustrează secvența de acțiuni necesare la actualizarea informațiilor pentru un produs. Procesul este unul simplu și liniar, bazat mai mult pe noduri decizionale, actualizarea informațiilor în sistem având loc doar în cazul în care produsul nu este șters anterior, evident, iar angajatul salvează toată modificările pe care le face asupra unui produs. La final, dacă nu mai există noduri diferite ce trebuie actualizate, procesul se încheie.

## Diagrame de stare

Acest tip de diagrame este folosit pentru specificarea posibilelor stărilor prin care poate trece un element, un obiect, punând accent totodată și pe modul în care se face această trecere. <sup>[11]</sup>

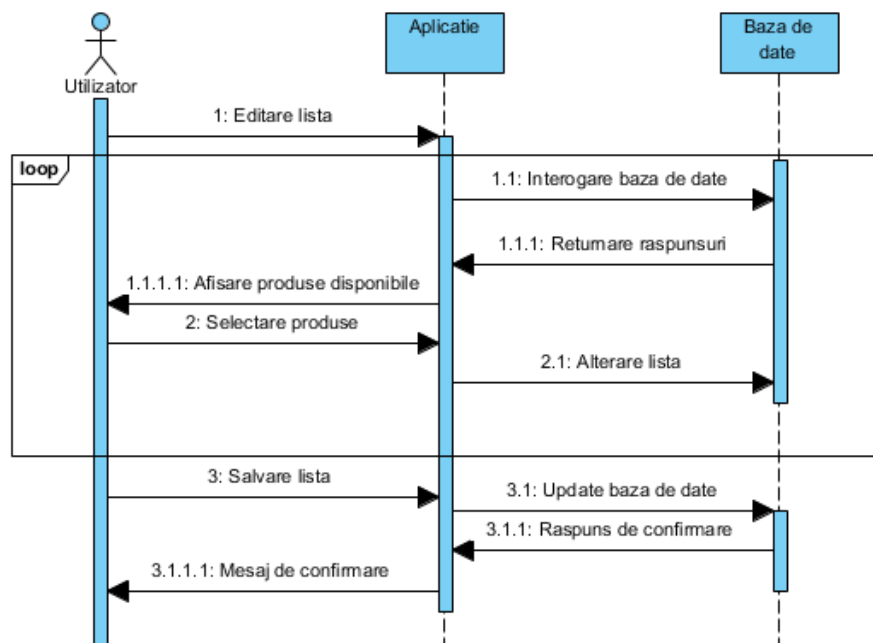


**Figura 2.2.4** Diagrama de stare pentru operațiunile efectuate la plasarea unei comenzi

Din diagrama anterioară (fig 2.2.4) putem extrage mai multe stări prin care poate trece o comandă plasată de utilizator, începând cu "În curs de adăugare", mai apoi "Validă" sau "Invalidă" în funcție de verificarea produselor conținute, urmată de "În așteptare", după efectuarea plății, ca în final această să se regăsească fie în starea "Plasată" fie "Refuzată", în funcție de răspunsul primit în urma procesării plății, aprobată sau nu.

## Diagrame de interacțiune

Rolul acestora este reprezentat de modelarea aspectelor dinamice ce intervin într-un sistem, fiind construite pe baza unui set de obiecte, al relațiilor dintre ele și al mesajelor ce se transmit dintr-un sens în altul. Diagrama de secvență este un subtip al diagramei de interacțiune, ce evidențiază ordinea mesajelor, luând în calcul dimensiunea temporală și le afișează în mod cronologic, pe verticală. <sup>[11]</sup>

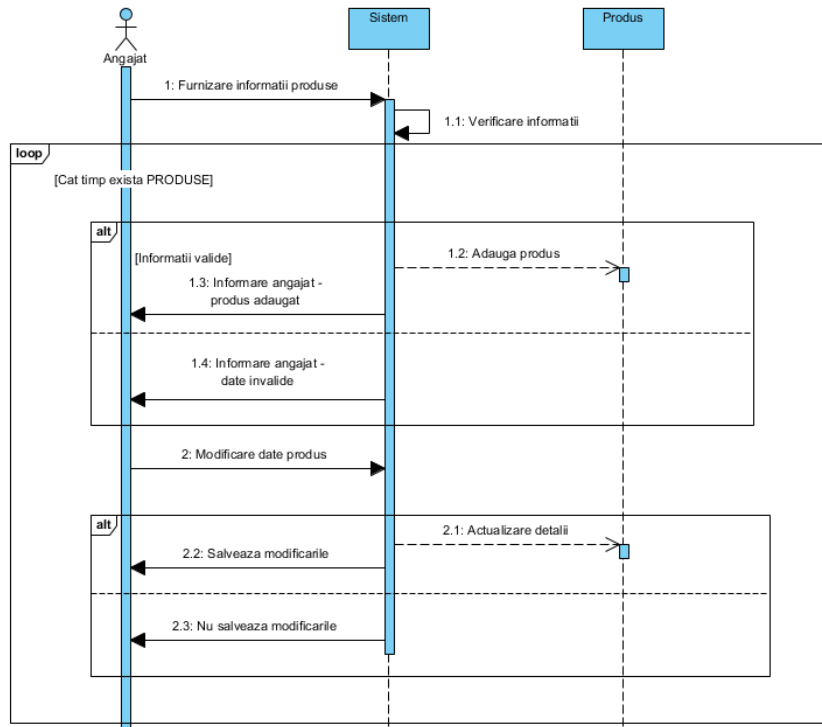


**Figura 2.2.5** Diagrama de secvență pentru scenariul “Editare listă”

Obiectele ce intră în calcul din figura de mai sus sunt Utilizatorul, care este în același timp și actorul principal, Aplicația și Baza de date. Fluxul începe cu un container de tip loop, care pentru fiecare execuție trimite cereri de la utilizator către aplicație, aceasta interogând mai departe baza de date cu toate produsele. Pe baza răspunsului returnat utilizatorul poate selecta produsele afișate, alterând astfel lista sau poate repeta procesul până la încheierea secvenței repetitive. Fluxul se încheie cu decizia utilizatorului de a salva lista, moment în care se face un update în sistem.

În ceea ce privește scenariul “Recepție marfă”, din figura de mai jos, actorul este acum reprezentat de către angajat, iar celelalte obiecte sunt sistemul și produsul ce trebuie adăugat. Inițial, sunt furnizate către sistem informațiile despre produse, iar după verificarea acestora se intră într-o secvență repetitivă, care se execută cât timp există produse. Mai apoi am folosit un fragment alternativ, ce are la bază o condiție de validitate a informațiilor despre produs, adăugând produsul și informând angajatul cu privire la acest lucru în cazul în care condiția este îndeplinită sau notificându-l în mod direct cu privire la invaliditatea datelor, în caz contrar. Diagrama se încheie cu un alt fragment combinat, tot de tip alternativ, care actualizează detaliile despre produs, dacă angajatul salvează modificările făcute sau altfel iese direct din sistem, fără salvarea modificărilor.

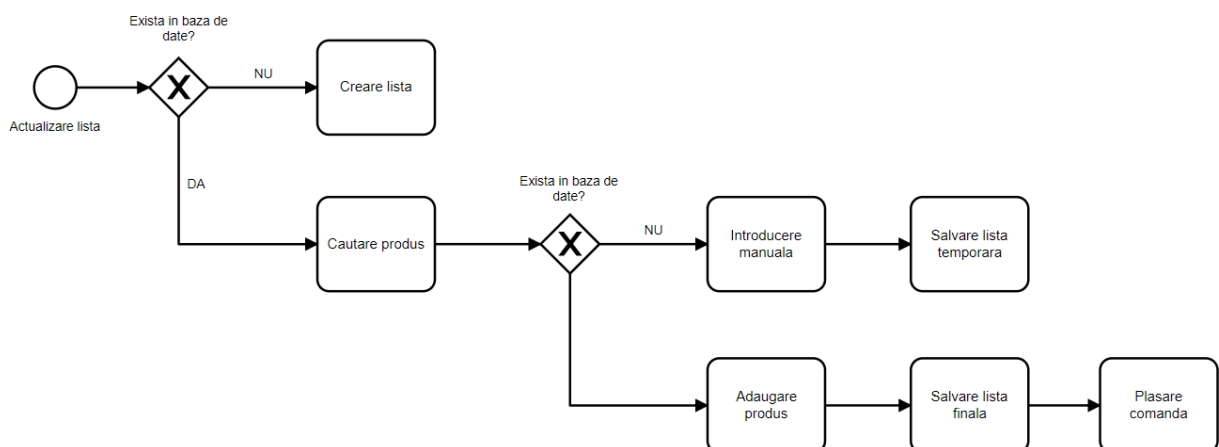




**Figura 2.2.6** Diagrama de secvență pentru scenariul “Recepție marfă”

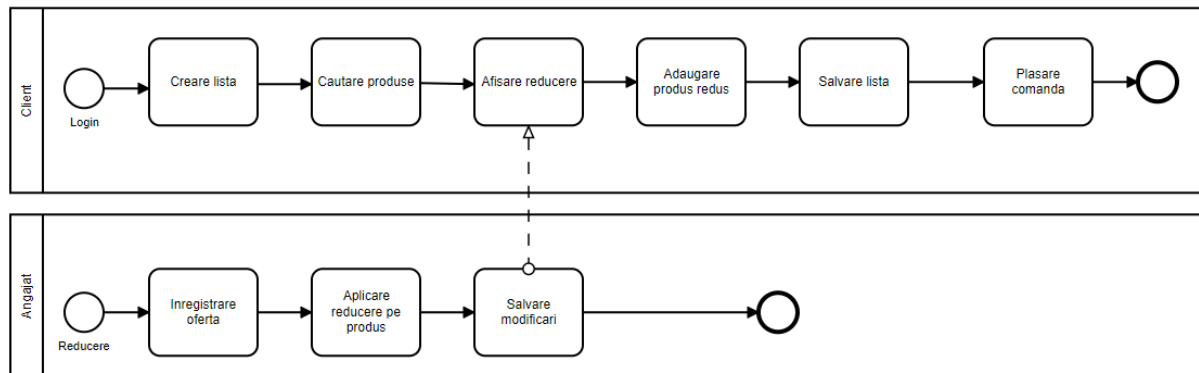
## Diagrame BPMN

Diagramele de procese de afaceri (**B**usiness **P**rocess **M**odel and **N**otation) oferă o notatie grafică bazată pe o tehnică asemănătoare diagramelor de activitate dar au ca obiectiv managementul proceselor de business atât pentru utilizatorii tehnici, cât și pentru utilizatorii sau partenerii de afaceri, oferind o notatie intuitivă și în același timp capabilă să reprezinte o semantică complexă a proceselor. <sup>[11]</sup>



**Figura 2.2.7** Diagrama de proces pentru actualizarea unei liste

În diagrama de proces de mai sus este descris fluxul de activități necesare la actualizarea unei liste, complet simplificată pentru a fi ușor de înțeles, folosind porți exclusive, corespondentul nodurilor decizionale din UML, prin care sunt create fluxuri alternative. Dintre acestea, fluxul de secvențe alege doar unul care se execută. În cazul nostru, porțile exclusive intervin la verificarea existenței listelor sau produselor în baza de date.



**Figura 2.2.8** Diagrama de colaborare în aplicarea unei reduceri

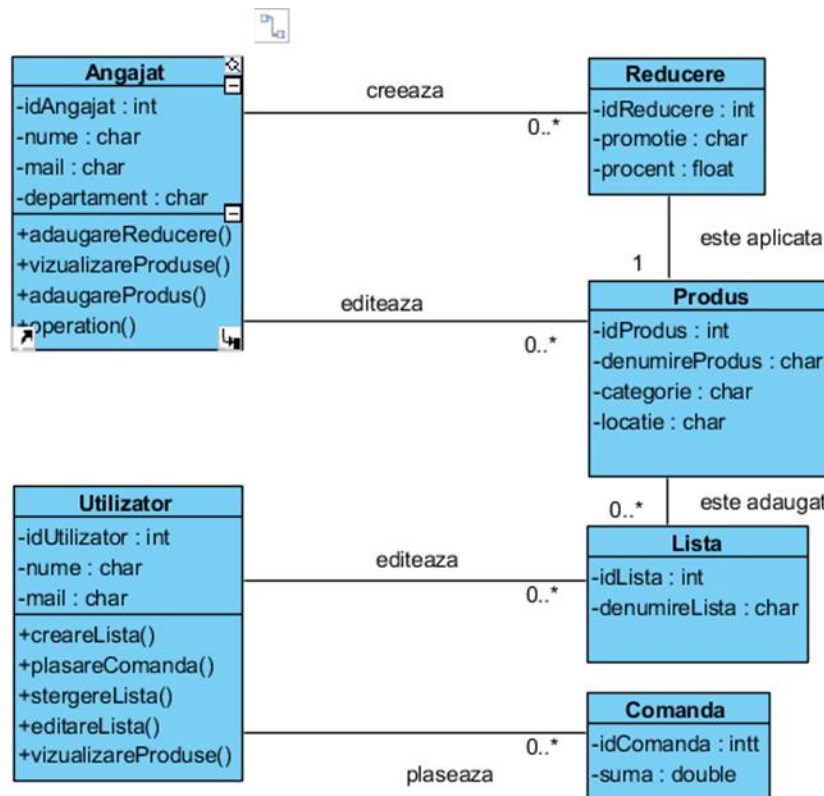
Celălalt tip de diagramă BPMN, cea de colaborare denotă interacțiunea dintre două sau mai multe entități organizaționale, de obicei conținând două sau mai multe containere, conectate prin fluxuri de mesaj. În cazul nostru este analizată percepția din punct de vedere al clientului și al angajatului referitoare la aplicarea reducerilor pentru produsele din ofertă.

În timp ce angajatul desfășoară tot procesul necesar aplicării acestora, clientul nu resimte efectul până când angajatul nu termină activitatea de salvare a modificărilor. Abia după, fluxul de mesaj afișează reducerea sau reducerile în urma procesului de căutare de produse din partea clientului. Mai departe, fluxul continuă doar în partiția acestuia, care poate opta pentru adăugarea unor astfel de produse în lista sa, până la activitatea finală, reprezentată de plasarea comenzii.

Atât pentru diagrama de process, cât și pentru cea de colaborare, am folosit funcționalitățile puse la dispoziție de bpmn.io<sup>[9]</sup>

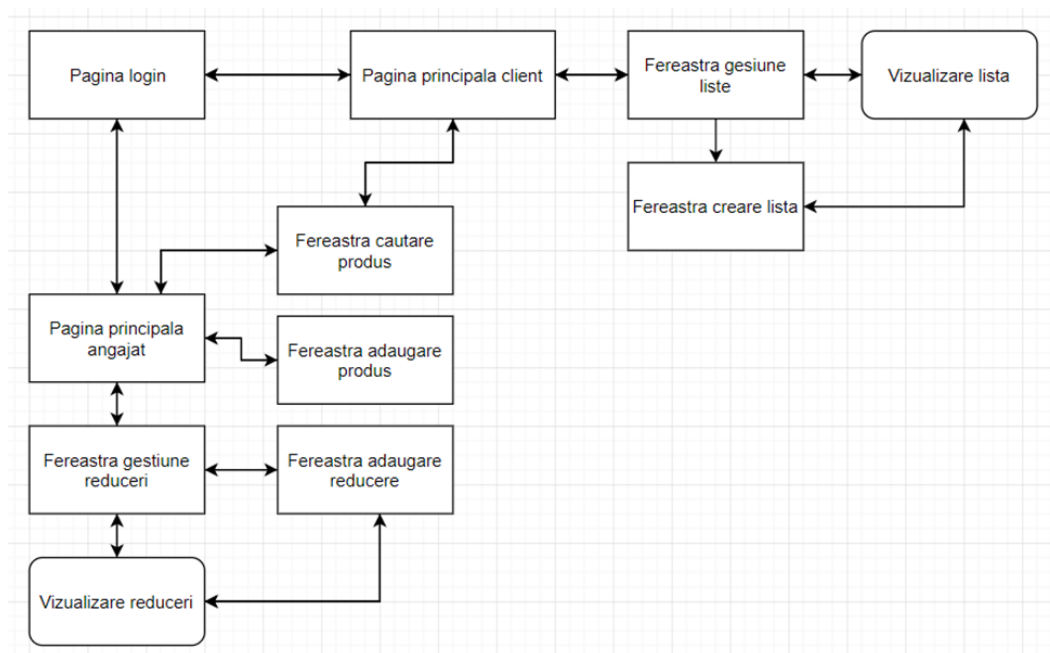
### 3. Proiectarea sistemului informatic

#### 3.1. Proiectarea noului sistem



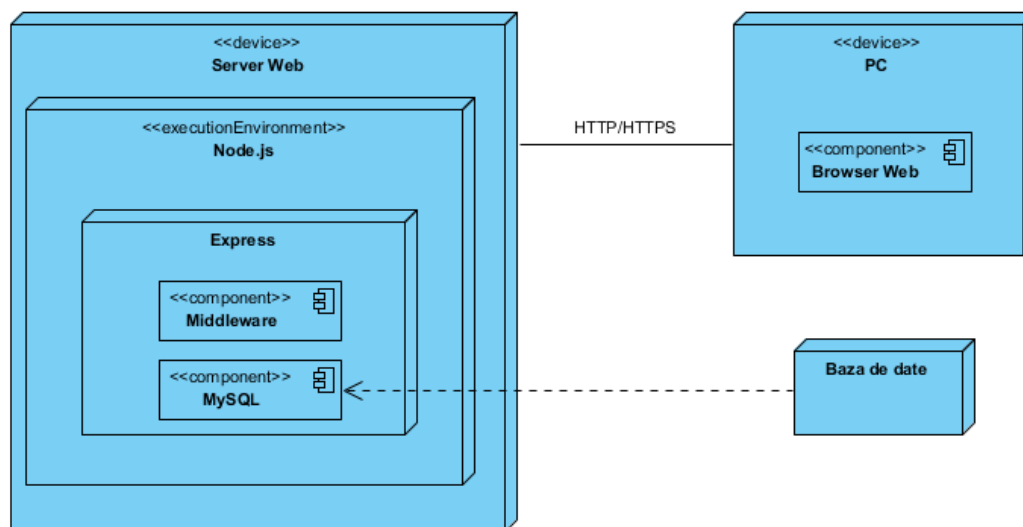
**Figura 3.1.1** Diagrama de clase detaliată

În diagrama din figura 3.1.1 sunt prezentate clasele principale ce intră în procesul de lucru al aplicației, alături de modul în care relaționează, atributele acestora și funcțiile pe care le implementează. Practic, descrie comportamentul claselor ce intră în componența sistemului informatic și ușurează înțelegerea modului în care legăturile și tot sistemul funcționează. Această diagramă trebuie să reflecte și să respecte atât fluxurile din cadrul diagramelor de secvențe, cât și a celor de activitate. <sup>[11]</sup>



**Figura 3.1.2** Diagrama interfețelor utilizator

În cadrul proiectării interfețelor sunt prezentate pagini, ferestre sau interfețe din cadrul aplicației, cu care utilizatorul ia contact și pe care le poate accesa. Funcționează după aceleași principii ca o hartă pentru un site (sitemap) și are ca scop principal înțelegerea și satisfacerea nevoilor utilizatorilor finali. Proiectarea a fost realizată folosind soft-ul online draw.io<sup>[10]</sup>



**Figura 3.1.3** Diagrama de desfășurare

Diagrama de desfășurare din figura anterioară prezintă arhitectura fizică pe baza căruia va fi implementat sistemul, împreună cu nodurile sale – calculatoare, device-uri. Componentele și obiectele sunt distribuite în interiorul nodurilor, permițând astfel o vizualizare a unităților executabile de pe fiecare nod. Scopul principal al acestei diagrame este cel de vizualizare a topologiei hardware pentru un sistem și descriere a componentelor hardware necesare implementării componentelor software.

## 3.2. Proiectarea schemei bazei de date

Plecând de la diagrama claselor detaliată am identificat cerințele sistemului din punct de vedere al datelor folosite, entităților necesare, asocierile dintre acestea și structurarea lor într-un mod cât mai omogen.

Astfel, plecând de la cele două entități principale, Clienții și Angajații, iar în completare venind cu entitatea comună reprezentată de Produse, am ajuns la următoarea versiune pentru schema bazei de date:

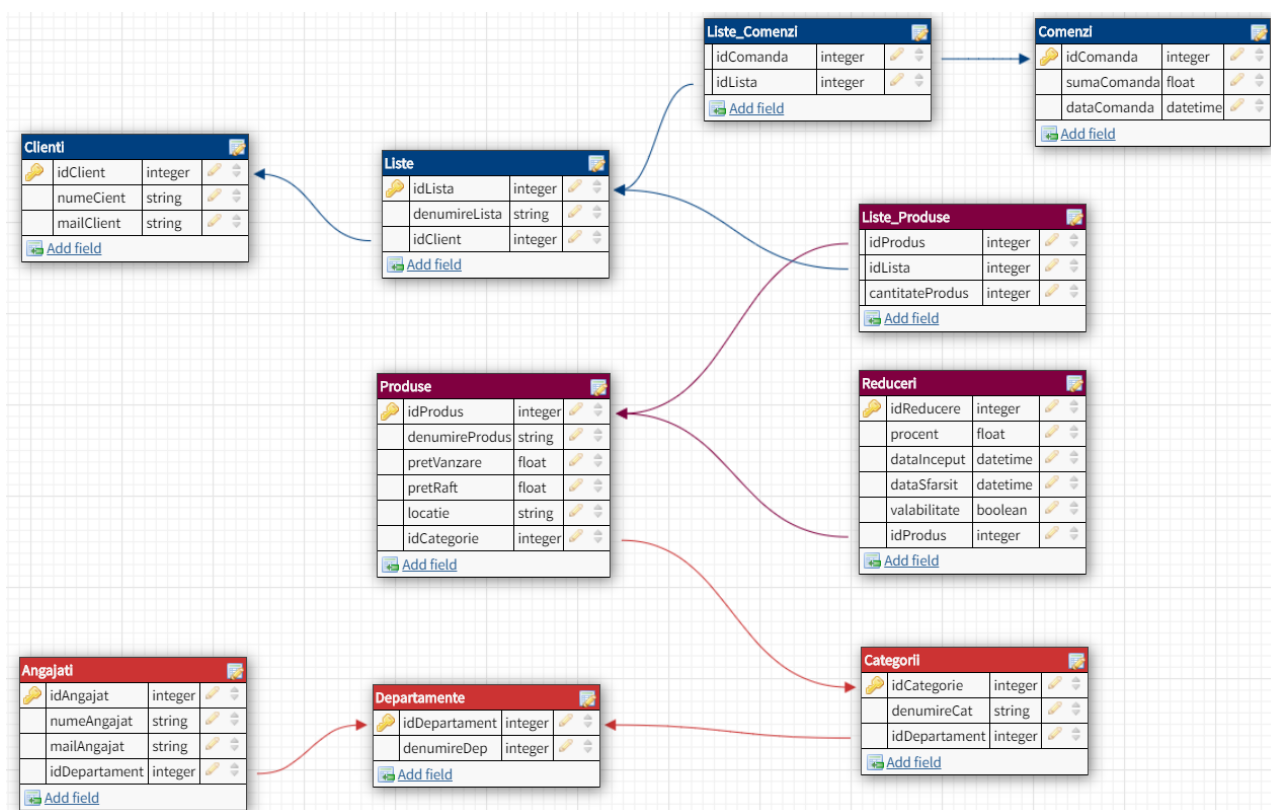


Figura 3.2.1 Schema bazei de date

După cum se poate observa și din schema de mai sus, am încercat să pun cât mai mult accent pe modularitatea bazei de date și libera posibilitate a dezvoltării ulterioare, evitând pe cât posibil hard-codarea anumitor câmpuri sau funcționalități și construind legături cât mai omogene între tabele. Majoritatea relațiilor dintre acestea sunt unu la mai multe (one-to-many), iar în cazurile în care legătura era de tip mai multe la mai multe (many-to-many) am creat tabele de legătură, ce înglobează chei secundare cu referință la cheile primare din cele două tabele implicate.

Generarea schemei am făcut-o folosind platforma dbdesigner.net<sup>[12]</sup>

În ceea ce privește schema generală, am creat un număr de 10 tabele, fiecare având următoarele roluri:

- **Cienti** – gestionează conturile utilizatorilor și păstrează detaliile personale ale acestora cu privire la nume și adresa de mail. Este bazată pe cheia primară idClient, de tip număr întreg și ar putea fi populată pe viitor și cu alte detalii, precum numărul de telefon, adresa, data înregistrării etc.
- **Liste** – conține datele generale cu privire la toate listele create la un moment dat. Este bazată pe cheia primară idLista, de tip număr întreg și alte date pe care le conține sunt denumirea listei dată de către utilizator și cheia secundară idClient, pentru a putea fi făcută legătura cu tabela Cienti. Relația între tablele Cienti și Liste este de tip one-to-many, adică un client poate avea mai multe liste.
- **Comenzi** – are în componență datele generale cu privire la comenzile plasate. Este bazată pe cheia primară idComanda, de tip număr întreg și celelalte date fac referire la suma comenzii și data la care a fost plasată.
- **Liste\_Comenzi** – gestionează legătura dintre cele două tabele prezentate mai sus, Liste și Comenzi, prin folosirea a două câmpuri de chei secundare, unul pentru idLista și unul pentru idComanda. Relația fiecărei tabele din cele enumerate cu Lista\_Comenzi este de tipul mai multe la unu (many-to-one).
- **Liste\_Produse** – similar ca tabela anterioară, aceasta gestionează legătura dintre cele tabelele Liste și Produse, prin folosirea cheilor secundare idLista și idProdus. Totuși, are în plus față de aceasta un câmp de tip număr întreg pentru cantitatea produsului selectat.
- **Produse** – conține datele generale cu privire la toate produsele existente în ofertă. Are la bază cheia primară idProdus, de tip număr întreg, iar celelalte câmpuri din

componenta sa fac referire la denumirea produsului, prețul întreg de vânzare, prețul la raft, care este același cu cel de vânzare în caz că nu există nicio reducere activă, locația din magazin și cheia secundară idCategorie, pentru a putea face legătura dintre un produs și categoria din care face parte.

- **Reduceri** – este tabela ce conține informații cu privire la toate reducerile, fie ele active sau inactive. Este bazată pe cheia primară idReducere, de tip număr întreg și conține procentul reducerii, câmp de tip număr real(float), cele două date, de început și de sfârșit între care este valid, un steag(flag) de tip boolean, pentru ușurință în verificarea valabilității și cheia secundară idProdus, pentru a determina produsul pe care este aplicată reducerea.
- **Categorii** – aici sunt gestionate toate categoriile de produse existente în oferta magazinului sau supermarket-ului. Conține cheia primară idCategorie, denumirea categoriei în cauză și cheia secundară idDepartament, pentru a identifica departamentul ce are în sarcină acea categorie de produse. De asemenea, are și o legătură de tip one-to-many cu tabela Produse, în cadrul unei categorii fiind evident incluse mai multe produse.
- **Departamente** – conține toate departamentele companiei, împreună cu denumirea lor și face legătura atât cu tabela de mai devreme, Categorii, cât și cu cea de Angajati, ambele fiind de tipul one-to-many (“Un departament are în gestiune mai multe categorii, respectiv are în componență mai mulți angajați”).
- **Angajati** – este ultima tabela din lista, dar funcționează la fel ca prima, Client, ca punct de plecare. În cadrul său se regăsesc conturile angajaților, păstrând detaliile personale ale acestora cu privire la nume și adresa de mail și departamentul în care lucrează, prin cheia secundară idDepartament. Și aceasta ar putea fi dezvoltată ulterior, prin adăugarea de detalii de tip salariu sau rol în cadrul companiei.

## 4. Implementarea sistemului informatic

### 4.1. Tehnologii informatice utilizate

În dezvoltarea sistemului informatic am folosit mai multe tehnologii, programe și alte utilitare, printre care cele mai importante ar fi:

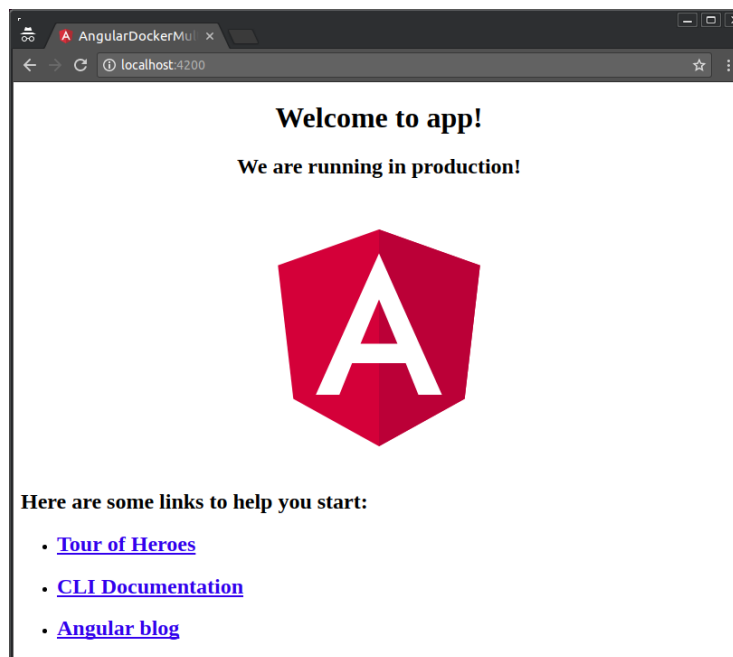
- **Angular 5** – framework dedicat aplicațiilor web
- **Node.js** – mediu folosit pentru partea de server JavaScript
- **MySQL** – pentru dezvoltarea și mentenanța bazei de date
- **SCSS** – extensie a sintaxei CSS
- **Lazy Loading** – design pattern pentru eficientizarea timpului de încărcare
- **OAuth** – protocol de autorizare pentru partea de autentificare
- **Github** – platformă ce gestionează și versionează codul sursă
- **Visual Studio Code** – mediu de dezvoltare și editor de cod
- **Visual Paradigm** – instrument folosit la proiectarea UML/BPMN

#### Angular 5

Lansat în noiembrie 2017, Angular 5 este un framework open-source de aplicații web, ce are la bază TypeScript (un superset strict sintactic al limbajului JavaScript). Îmbunătățirile majore în această versiune, față de versiunile trecute, includ suport pentru aplicațiile web progressive (PWA), optimizarea Build-urilor și îmbunătățiri legate de Material Design. <sup>[13]</sup>

Angular este folosit în principal pentru proiecte bazate pe componente, în cadrul cărora e nevoie de dezvoltarea unor tablouri de analiză (dashboards) și în care conținutul se reîncarcă automat fără a fi nevoie de reîncărcarea întregii pagini.





**Figura 4.1.1** Interfața implicită la configurarea unei aplicații Angular

Punctele forte de care dispune Angular 5, în cea mai recentă versiune a lui, sunt:

- **Dezvoltare multiplatformă** - cu Angular poți învăța să construiești o singură aplicație, iar mai apoi doar să reutilizezi codul și să îți folosești abilitățile în dezvoltarea de aplicații pentru orice platformă: web, web mobil, mobil native și desktop nativ. <sup>[14]</sup>
- **Viteză și performanță** - poți obține viteza maximă pe platforma web și să o duci chiar mai departe prin web workers (script-uri ce rulează în fundal) și rendering pe partea de server. Angular oferă control asupra scalabilității și poate îndeplini cerințe foarte mari de date pe baza modelelor RxJS (Reactive Extensions for JavaScript) sau altor modele similare. <sup>[14]</sup>
- **Unelte incredibile** – poți construi rapid funcțiile cu șabloane simple, declarative, iar mai apoi să le extinzi cu propriile componente sau folosind gama largă de componente deja existente. În plus, cu aproape fiecare IDE și editor poți obține ajutor și feedback, astfel încât focusul să fie pe construirea aplicației în sine, nu pe problemele de cod și sintaxă. <sup>[14]</sup>

## Node.js

Node.js este un mediu de rulare JavaScript care procesează cererile de intrare într-o buclă, numită bucla eveniment.

- **Threading** - Node.js funcționează pe o singură buclă de evenimente pentru fire (thread-uri), folosind apeluri intrare/ieșire fără blocare. Astfel poate suporta zeci de mii de conexiuni concurente fără a mai comuta între fire. Un dezavantaj al acestei abordări cu un singur fir de execuție este că Node.js nu permite scalarea verticală prin creșterea numărului de nuclee CPU ale mașinii pe care rulează, fără a utiliza un modul suplimentar. <sup>[15]</sup>
- **V8** - V8 este motorul de executare JavaScript, care fusese inițial conceput pentru Google Chrome. Acesta devenit open-source în 2008, iar scopul lui este să compileze codul sursă JavaScript la codul mașinii native în timpul executării, în loc să îl interpreteze înainte. Node.js utilizează biblioteca libuv pentru a gestiona evenimente asincrone. Acesta este un strat de abstractizare pentru funcționalitatea sistemului de rețea și a fișierelor. <sup>[15]</sup>
- **Gestiunea pachetelor** - npm este un manager de pachete preinstalat pentru platforma server Node.js. Acesta instalează programele Node.js din registrul npm, gestionând programele Node.js venite de la terți (3rd party). Pachetele din registrul npm pot varia de la simple biblioteci de ajutor la un număr foarte mare de componente ce pot încărca un proiect fără a fi folosite în scop util.

```
"devDependencies": {
  "@angular/compiler-cli": "^6.0.3",
  "@angular-devkit/build-angular": "~0.6.8",
  "typescript": "~2.7.2",
  "@angular/cli": "~6.0.8",
  "@angular/language-service": "~6.0.3",
  "@types/jasmine": "~2.8.6",
  "@types/jasminewd2": "~2.0.3",
  "@types/node": "~8.9.4",
  "codemlizer": "~4.2.1",
  "jasmine-core": "~2.99.1",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~1.7.1",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "~2.0.0",
  "karma-jasmine": "~1.1.1",
  "karma-jasmine-html-reporter": "~0.2.2",
  "protractor": "~5.3.0",
  "ts-node": "~5.0.1",
  "tslint": "~5.9.1"
}
```

Figura 4.1.2 Exemplu pentru o listă de pachete

## MySQL

MySQL este un sistem ce are ca scop gestionarea bazelor de date relaționale, ce funcționează sub egida Oracle și este bazat pe SQL (Structured Query Language). Rulează toate platformele mari, Linux, UNIX și Windows și este cel mai des utilizat pentru dezvoltarea de aplicații web sau publicare în mediul online.<sup>[13]</sup>

MySQL se bazează pe un model client-server, care gestionează toate instrucțiunile și comenzile aplicate asupra bazei de date. Serverul MySQL este disponibil ca program separat pentru utilizarea într-un mediu de rețea client-server și ca o bibliotecă ce poate fi integrată în aplicații de sine stătătoare.<sup>[16]</sup>

Ca și caracteristici de bază, MySQL permite stocarea și accesarea datelor prin cadrul mai multor motoare de stocare și este capabil să replice tabelele de date și partiții cu o performanță și o durabilitate ridicată. Pentru utilizarea sa nu este nevoie de un limbaj nou de programare sau folosirea altor comenzi decât cele standard SQL.

De asemenea, MySQL lasă la îndemâna utilizatorilor alegerea motorului de stocare potrivit pentru orice tabel dat, programul poate folosi mai multe motoare de stocare pentru tabele individuale. Unul dintre motoarele MySQL este InnoDB, motor proiectat pentru disponibilitate ridicată.<sup>[16]</sup>

În plus, MySQL oferă și o gamă largă de instrumente de analiză și raportare a datelor, existând o diversitate de astfel de instrumente de analiză disponibile de la companii de software, cum ar fi QlikView sau BIRT (Business Intelligence Reporting Tools).

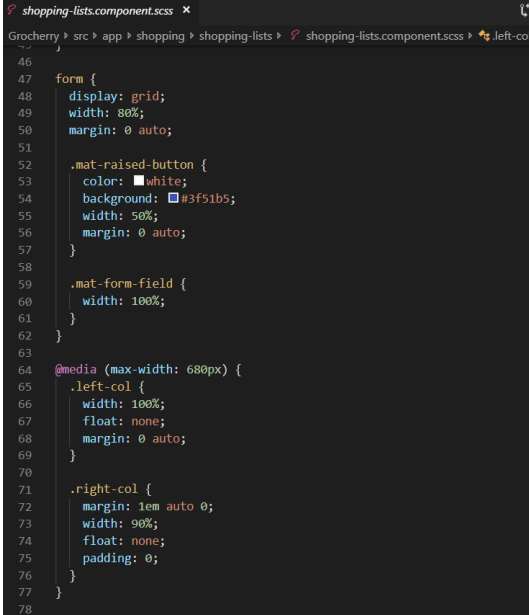
În trecut, principala diferență dintre MySQL și SQL era conferită de varietatea platformelor ce pot fi utilizate, MySQL fiind în avantaj, întrucât SQL putea fi utilizată doar pe Windows. Totuși, Microsoft a extins acum 2 ani suportul SQL și pentru Linux, reducând astfel drastic diferențele dintre cele două și lăsând la decizia utilizatorului și a preferințelor sale ce soluție să folosească.

## SCSS

Dacă CSS este limbajul de stil pe care orice browser îl folosește pentru partea de stilizare, SCSS este un tip special de fișier pentru SASS, cu o mulțime de funcționalități suplimentare CSS-ului, cum ar fi variabile, operatori sau procese de nesting asupra selectorilor din CSS, făcând-ul mai ușor și mai rapid decât acesta. <sup>[13]</sup>

Cu ajutorul preprocesorului SCSS poate fi redus considerabil numărul de repetări făcute în cod, făcând-ul mai ușor de înțeles și de menținut și ajutând la respectarea principiului DRY (Don't repeat yourself). <sup>[17]</sup>

În tot acest timp, SCSS este compatibil în totalitate cu sintaxa CSS, fiind de fapt o extensie a acesteia. Mai exact, asta înseamnă că orice foaie de stil CSS validă este un fișier SCSS valid, folosit în același scop.



```
shopping-lists.component.scss
Grocherry > src > app > shopping > shopping-lists > shopping-lists.component.scss > left-col
46
47 form {
48   display: grid;
49   width: 80%;
50   margin: 0 auto;
51
52   .mat-raised-button {
53     color: white;
54     background-color: #3f51b5;
55     width: 50%;
56     margin: 0 auto;
57   }
58
59   .mat-form-field {
60     width: 100%;
61   }
62 }
63
64 @media (max-width: 680px) {
65   .left-col {
66     width: 100%;
67     float: none;
68     margin: 0 auto;
69   }
70
71   .right-col {
72     margin: 1em auto 0;
73     width: 90%;
74     float: none;
75     padding: 0;
76   }
77 }
78
```

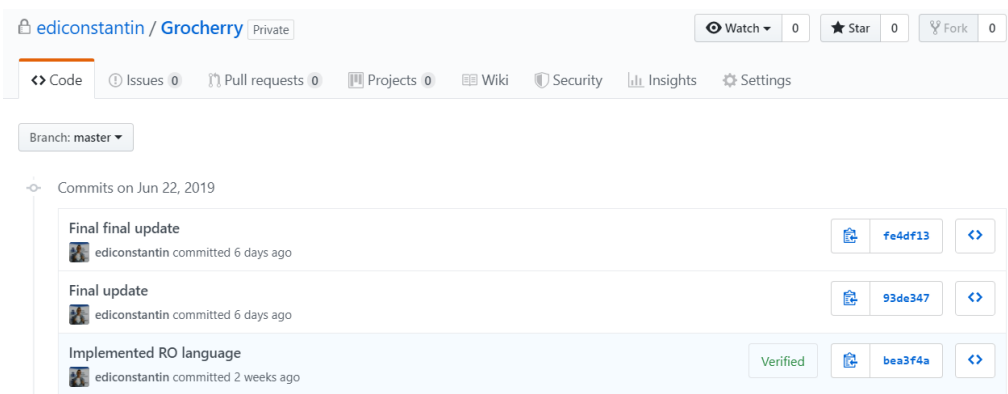
**Figura 4.1.3** Exemplu de fișier de tipul SCSS

## GitHub

Pentru a înțelege GitHub, trebuie să înțelegem mai întâi ce înseamnă Git. Git este un sistem de versionare a codului sursă, versiuni ce apar atunci când dezvoltatorii creează ceva, fie un site, o

aplicație sau un document de orice fel, iar apoi vin cu schimbări constante asupra codului sau a fișierelor, până ajung la o versiune finală, care este de obicei și de lansare.<sup>[18]</sup>

Controlul versiunilor presupune stocarea într-un mod centralizat a tuturor acestor revizii, într-un depozit numit “repository”, spațiu în care le este permis dezvoltatorilor să colaboreze cu ușurință, deoarece pot descărca în orice moment ultima versiune, pot efectua modificări și pot contribui la rândul lor cu o versiune și mai nouă. Tot istoricul ce conține aceste schimbări poate fi accesat și urmărit de oricine, în cazul în care proiectul este unul public sau doar de membrii proiectului respectiv, în cazul în care este privat.



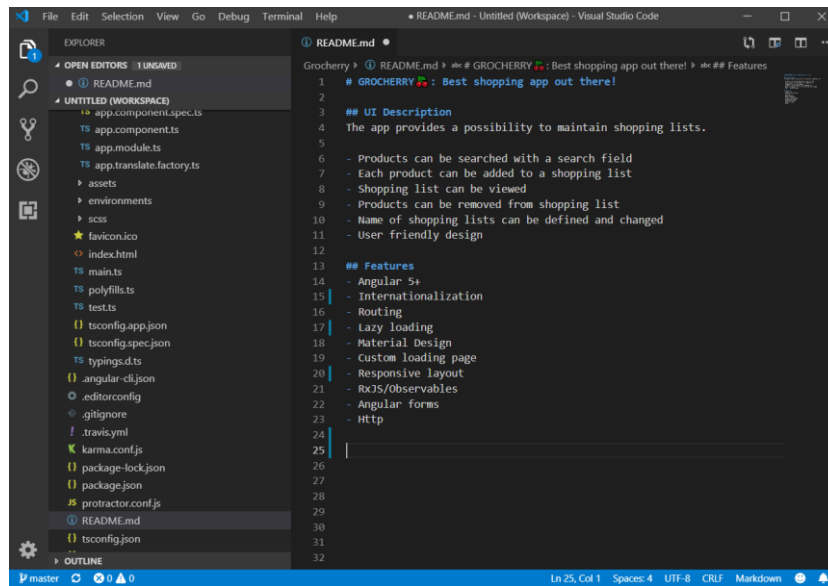
**Figura 4.1.4** Interfața ce conține log-urile de commit în Github

## Visual Studio Code

Visual Studio Code combină simplitatea unui editor de cod sursă cu IntelliSense și multe alte utilități și instrumente puternice de dezvoltare. Este un editor extrem de rapid, ce are la bază ciclul editare-construire-depanare (edit-build-debug), ce oferă support pentru un număr mare de limbi diferite și oferă funcționalități ce cresc considerabil productivitatea, cum ar fi evidențierea sintaxelor, auto-indentarea, fragmente și multe altele.<sup>[19]</sup>

VS Code include un program pentru depanare interactivă (interactive debugging), astfel încât în timp ce este parcurs codul sursă, utilizatorul poate examina variabilele, vizualiza stiva de apela sau chiar să execute comenzi în consolă.

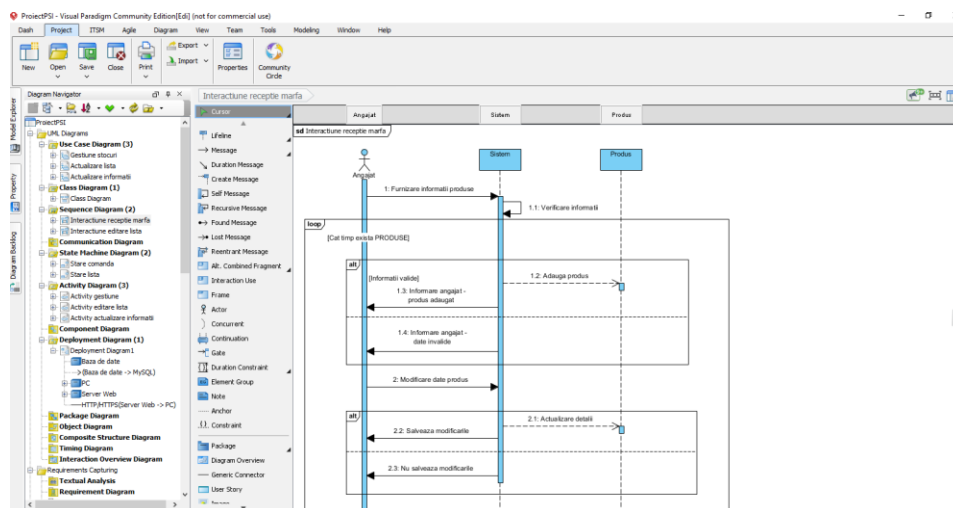
În plus, oferă suport pentru mediul de dezvoltare Node.js cu JavaScript și TypeScript, dar și pentru tehnologii web cum ar fi React, HTML, CSS, SCSS sau JSON.



**Figura 4.1.5** Exemplu de fișier deschis în Visual Studio Code

## Visual Paradigm

Visual Paradigm este un instrument de proiectare și gestionare ușor de utilizat și destinat sistemelor informatice. Acesta pune la dispoziția dezvoltatorilor de software o platformă ce are capacitatea de a sta la baza construirii unor aplicații de calitate ridicată. În același timp, facilitează o interoperabilitate excelentă cu alte instrumente CASE, dar și cu cele mai multe medii de dezvoltare. Suportă notarea UML 2, SysML și BPMN, în plus oferind capacitatea de generare de rapoarte și de codificare, inclusiv generarea de coduri.<sup>[20]</sup>



**Figura 4.1.6** Interfața aplicației Visual Paradigm

## 4.2. Prezentarea aplicației

Luând în calcul toate aspectele menționate mai sus, am ajuns la implementarea funcționalităților aplicației, al cărei scop final a fost și este transformarea procesului de cumpărături într-unul cât mai simplu și mai plăcut, ajutând în același timp la o gestionare a stocurilor cât mai eficientă și mai rapidă din partea angajaților și a reprezentanților de magazin.

Actorul principal al aplicației este utilizatorul, fie că ia forma clientului, care dorește să își organizeze listele de cumpărături și necesarele de zi cu zi, ori a angajatului, care trebuie să se asigure de validitatea datelor cu privire la produsele oferite spre vânzare.

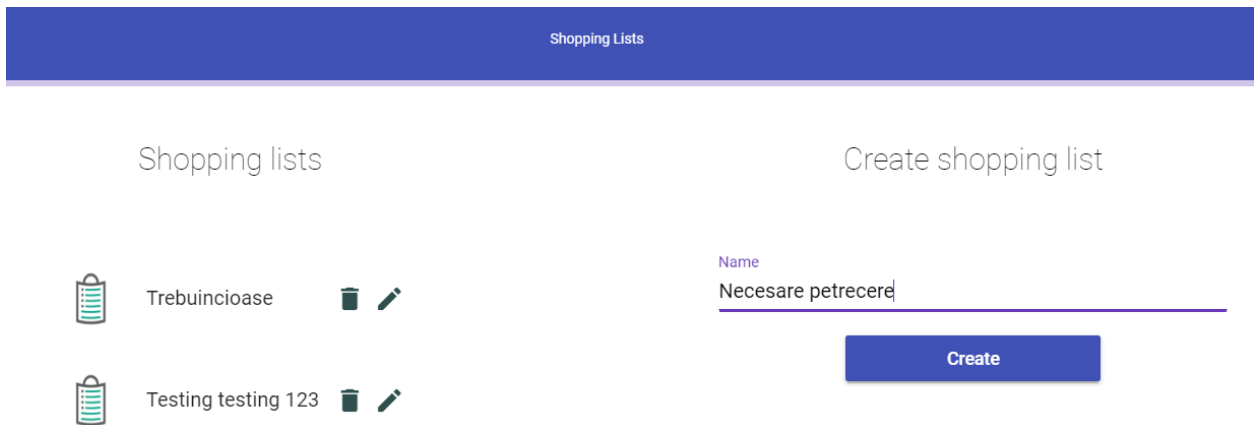
Astfel, consider că cele două module enumerate sunt și cele principale, deoarece tot fluxul de informații și date transmise în cadrul sistemului au la bază acest aspect al utilizatorului și al rolului pe care îl are. Ca importanță, consider că modulul ce integrează procesul de cumpărături este mai important în detrimentul celui administrativ, întrucât impactul pe care îl are se propagă mult mai repede și la un nivel mult mai mare, în timp ce detaliile tehnice și problemele întâmpinate sunt mai ușor de discutat și rezolvat intern, în cadrul unei companii.

Așadar, în cele ce urmează voi prezenta cazul de utilizare ce are în prim-plan clientul și activitățile pe care acesta le poate întreprinde în cadrul aplicației, prezentând succint paginile la care are acces și elementele din cadrul fiecăreia.

Prima fereastră cu ia contact utilizatorul după autentificare este pagina ce conține toate listele sale de cumpărături. Design-ul este unul simplu, minimalist, fără prea multe detalii, dar cu toate funcționalitățile la un singur click distanță. În cadrul acesteia, el are mai multe opțiuni:

- Să creeze o listă nouă, furnizând un nume pentru aceasta
- Să editeze una dintre listele deja existente
- Să șteargă o listă.

Să presupunem că în cazul de față el dorește să adauge o nouă listă de cumpărături, dedicată zilei sale de naștere. Astfel, după introducerea numelui, pe baza unui formular Angular simplu, în baza de date este introdusă o listă nouă, care în mod implicit nu conține niciun produs.

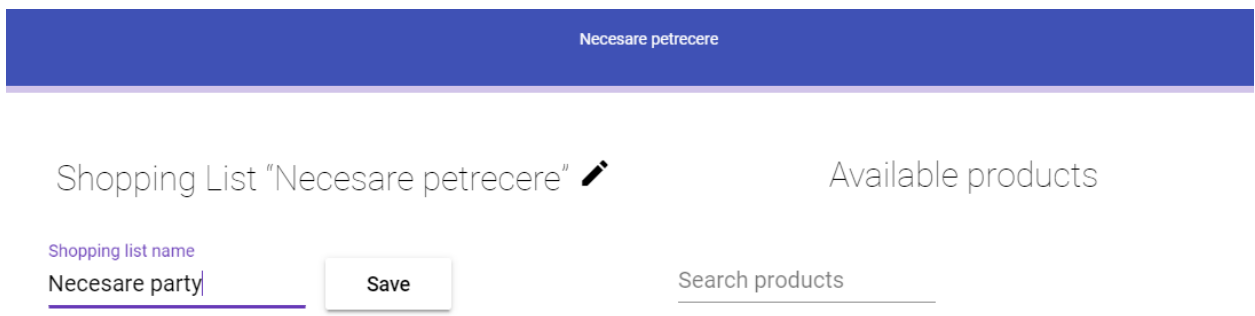


**Figura 4.2.1** *Fereastra de deschidere*

Mai apoi, în mod evident, dorește să editeze noua sa listă de cumpărături și să adauge produsele pe care le consideră necesare. Astfel ajunge în fereastra dedicată listei sale, foarte asemănătoare cu cea precedentă, dar cu funcționalități diferite. Acțiunile pe care le poate face în această pagină sunt:

- Să modifice numele listei selectate
- Să caute produse pe baza unor cuvinte cheie
- Să adauge sau să șteargă produse din listă

Momentan, vom presupune din nou că acesta dorește doar să schimbe numele listei, în cazul din față din “Necesare petrecere” în “Necesare party”. Astfel, la apăsarea butonului **Save**, toate modificările se vor propaga mai departe în pagină.



**Figura 4.2.2** *Editarea unei liste de cumpărături*



Cu modificările făcute, acesta poate continua în aceeași fereastră, fără a mai fi nevoie de actualizarea întregii pagini. Astfel, următorul pas poate fi instant adăugarea de produse, pe coloana din dreapta a ecranului fiind afișate toate rezultatele întoarse pentru căutarea făcută.

Necesare party

Shopping List "Necesare party" ✎

Available products

Shopping list name  
Necesare party Save

Search products  
lumanari

Lumanari aniversare

+ Add to shopping list

Lumanari de biserica

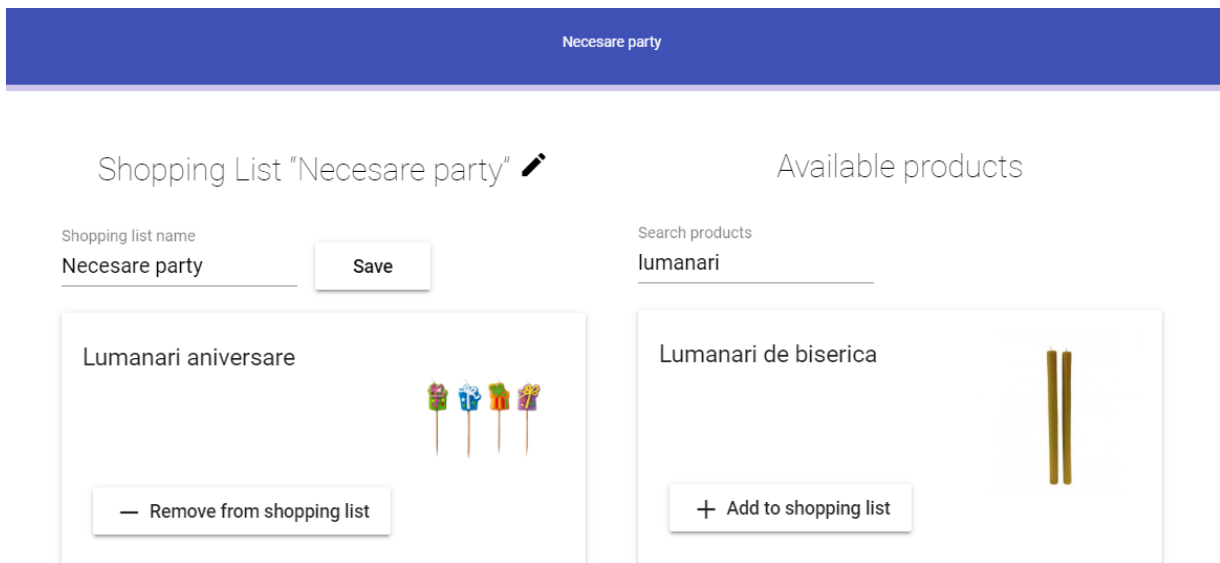
+ Add to shopping list

Saved OK

**Figura 4.2.3** Fereastra de căutare a produselor

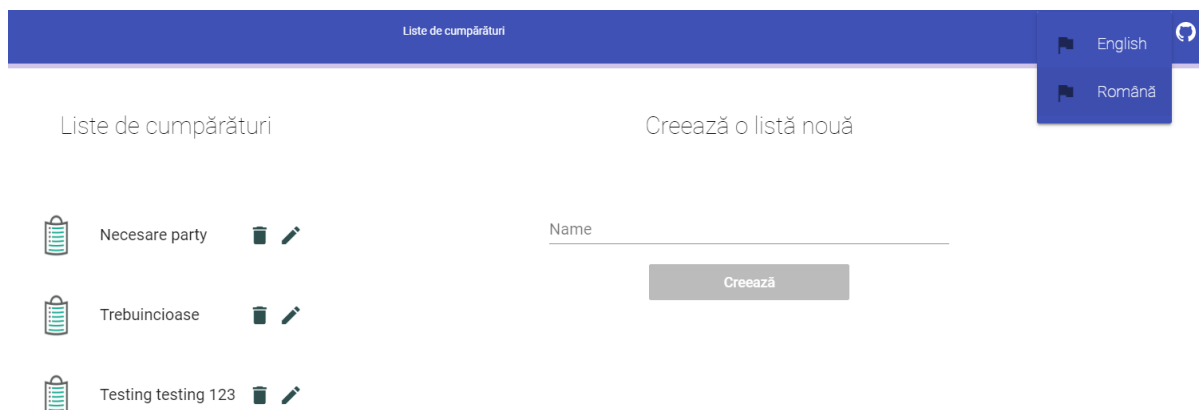
În cazul în care una sau mai multe opțiuni se potrivesc cu necesitățile clientului, aceasta poate adăuga toate acele produse în lista sa, putând la fel de ușor să le și șteargă după, în caz că se răzgândește.

Pentru situația noastră, căutarea a fost făcută pe baza cuvântului cheie “lumanari”, obținându-se două rezultate, așa cum se poate observa și în figura de mai sus. Dintre cele două, doar o opțiune era potrivită cu tematica listei, deci clientul nostru se decide să o adauge în lista sa, după cum se poate vedea și în figura următoare.



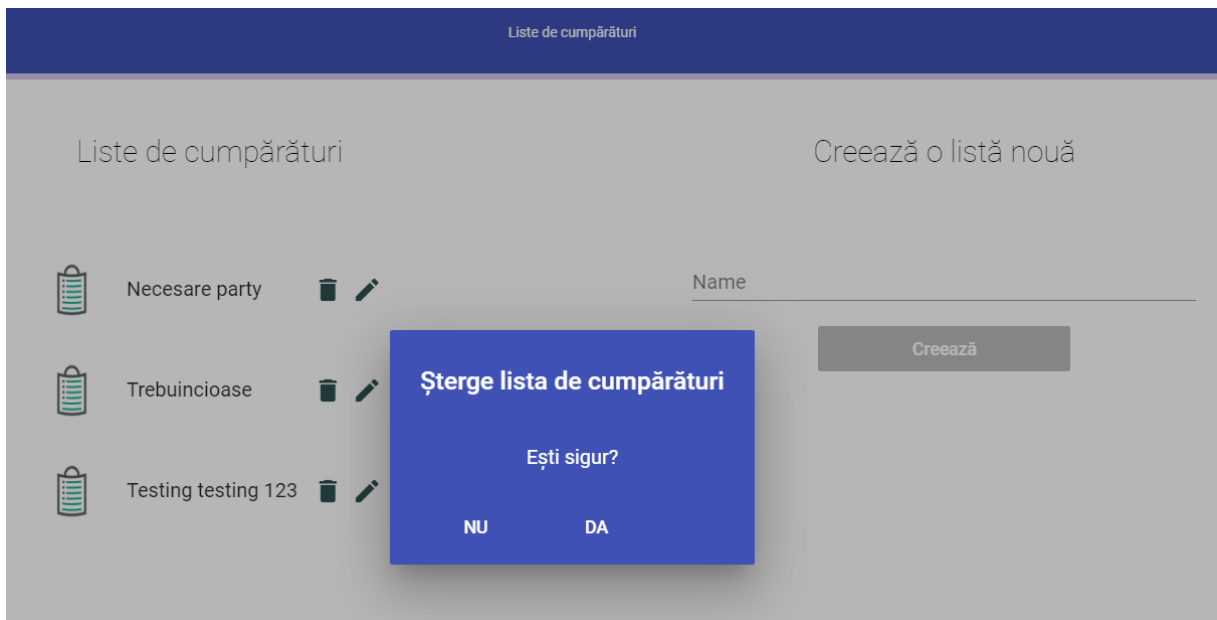
**Figura 4.2.4** Adăugarea și actualizarea listei de cumpărături

După ce finalizează lista propusă, clientul se întoarce în cadrul ferestrei principale și decide să testeze o altă funcționalitate prezentă, și anume schimbarea limbii contextuale din Engleză în Română. Din nou, toate modificările se propagă instant în cadrul întregii pagini și a aplicației, modificând numele titlurilor și textele din cadrul butoanelor sau al meniului.



**Figura 4.2.5** Modificarea limbii contextuale

În final, clientul nostru ajunge la concluzia că nu mai dorește să țină o petrecere și vrea să șteargă lista abia finalizată, devenind practic inutilă. Astfel, apasă pe butonul de ștergere, iar înainte ca acțiunea să devină ireversibilă și să șteargă lista din baza de date, o fereastră de tip pop-up apare pe ecran, pentru confirmare.



**Figura 4.2.6** Fereastra de ștergere a unei liste

# Concluzii

În final, sunt de părere că sistemul informatic dezvoltat ar avea șanse reale să concureze pe piață, dacă luăm în considerare variantele existente la momentul actual și ce pot oferi ele. Consider că un mare plus al aplicației față de restul soluțiilor este simplitatea sa, conferindu-i astfel rapiditate în procesarea datelor, iar un avantaj concurențial ar fi integrarea în cadrul aceleiași platforme a funcționalităților necesare de ambele părți ce contribuie la procesul de vânzare-cumpărare.

Ca grad de finalizare, cred sincer că la momentul de față, sistemul ar trebui considerat un simplu prototip, fiind nevoie de mai mult timp și consum de resurse pentru îndeplinirea tuturor cerințelor ce pot apărea pentru o astfel de aplicație. Totuși, plecând de la stadiul actual și cu o echipă dedicată în spate, sunt de părere că am putea atinge potențialul pe care îl are aplicația și am putea concura în mod serios cu alte soluții similare.

Ca posibile dezvoltări, am putea lua în calcul noi module ce ar merita adăugate, evident după îmbunătățirea celor deja existente, iar ca funcționalități noi ce ar aduce un plus de valoare aplicației, consider că următoarele ar fi cele mai importante:

- Adăugarea de roluri și crearea de ierarhii în cadrul modulului destinat angajaților;
- Adăugarea mai multor statistici și indici de performanță în cadrul aceluiași modul;
- Posibilitatea scanării produselor din partea de client;
- O modularitate mai mare a platformei în sine, pentru a putea fi integrate mai multe magazine și mai mulți parteneri;
- Crearea unei aplicații de tip web progresiv (PWA), pe baza celei web, pentru a putea deservi dispozitivele mobil sub forma unor aplicații native.

În stadiul final, cred că aplicația ar prezenta și mai multe avantaje decât cele enumerate deja, precum:

- Costuri reduse pentru dezvoltarea și mentenanța platformei;
- Procesul facil de trecere de la un tip de device la altul;
- Tehnologii actuale, populare printre dezvoltatori și actualizate constant;
- Interes ridicat din partea colaboratorilor, pentru oferirea unor servicii de administrare ce satisfac atât nevoile lor, cât și ale clienților.

# Bibliografie

- [1] - I. Lungu, A. Bâra, M. Andronie – Administrarea bazelor de date, Editura ASE, 2008
- [2] - <https://interestingengineering.com/9-interesting-failed-inventions-past> - 9 of the Most Interesting Failed Inventions from the Past
- [3] - <https://theculturetrip.com/north-america/usa/articles/10-inventions-no-one-thought-would-be-a-success> - 10 Inventions No One Thought Would Be a Success
- [4] - <https://www.businessinsider.com/inventions-that-were-ridiculed-2016-8> - World-changing inventions that were ridiculed when they came out
- [5] - <http://whoinventedthe.com/who-invented-supermarkets/> - Who Invented Supermarkets
- [6] - <http://www.spellboundinc.com/which-inventions-are-changing-the-grocery-industry/> - Which Inventions are Changing the Grocery Industry
- [7] - <https://rubygarage.org/blog/grocery-apps-guide> - Go-to Strategies For Six Types of Grocery Apps
- [8] - <https://edition.cnn.com/2018/10/03/tech/amazon-go> - I visited Amazon Go and saw the future of retail
- [9] - <https://demo.bpmn.io/> - BPMN Modeler
- [10] - <https://www.draw.io/> - Flowchart Maker and Online Diagram Software
- [11] – Suport de curs și seminar – Proiectarea Sistemelor Informatice – Prof. A. Corbea
- [12] - <https://www.dbdesigner.net/> - Online Database Schema Design and Modeling Tool
- [13] - <https://devdocs.io/> - DevDocs API Documentation
- [14] - <https://www.quora.com/What-is-Angular-5> - Angular 5 Documentation
- [15] - <https://nodejs.org/en/about/> - Node.js Documentation
- [16] - <https://searchoracle.techtarget.com/definition/MySQL> - MySQL Documentation
- [17] - <https://stackoverflow.com/questions/46400443/what-is-the-difference-between-css-and-scss> - SCSS Documentation
- [18] - <https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/> - GitHub Documentation
- [19] - <https://code.visualstudio.com/docs/editor/whyvscode> - Visual Studio Code Documentation
- [20] - <https://www.visual-paradigm.com/features/> - Visual Paradigm Documentation