

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

Organizator comenzi cantină

propusă de

Lazăr Vasile

Sesiunea: *iulie, 2020*

Coordonator științific

Lect. Dr. Condurache Rodica

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Organizator comenzi cantină

Lazăr Vasile

Sesiunea: *iulie, 2020*

Coordonator științific

Lect. Dr. Condurache Rodica

Avizat,
Îndrumător Lucrare de Licență
Lect. Dr. Condurache Rodica

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul(a) **Lazăr Vasile** domiciliat în **România, jud. Galați, com. Umbrărești, str. Alexandru Ioan Cuza, nr. 816**, născut(ă) la data de **21 septembrie 1998**, identificat prin CNP **1980921171690**, absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, **Facultatea de Informatică** specializarea **informatică**, promoția **2020**, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Organizator comenzi cantină** elaborată sub îndrumarea doamnei **Lect. Dr. Condurache Rodica**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data,

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „**Organizator comenzi cantină**”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent **Lazăr Vasile**

Cuprins

Introducere.....	6
Motivație.....	6
Obiectiv	6
Contribuții.....	7
Cerințe funcționale	8
1. CAPITOLUL I. Tehnologii folosite:	10
1.1. Flutter framework	10
1.1.1. Sistemul de operare Android	10
1.1.2. Sistemul de operare iOS	12
1.2. Web API ASP.NET CORE.....	14
1.3. Angular	15
1.4. MongoDB	15
2. CAPITOLUL II. Arhitectura aplicației	17
2.1. Structura.....	17
2.2. Modulele aplicației	18
2.2.1. Modulul Interfață Web	18
2.2.2. Modulul interfața Android și iOS.....	27
2.3. Modulul bază de date	30
2.4. Modulul API Gateway	32
3. CAPITOLUL III. Detalii de implementare	33
3.1. Modulul API Gateway	33
3.2. Modulul interfață web.....	39
3.3. Modulul interfață mobile	42
Concluzii.....	44
Bibliografie.....	45

Introducere

Motivație

În această lucrare de licență principala problemă abordată este cea a timpului, a așteptării continue și a statului la coadă, precum și cea a incertitudinii în legătură cu ceea ce am prefera în momentul în care apare pofta de mâncare și am decis să luăm masa la cantină sau restaurant.

Principalul scop al aplicației pentru utilizatori precum studenți și profesori este cel de a reduce timpul de așteptare și de a realiza comenzile de produse într-un ritm cât mai alert pentru cei înfometați prin generarea unui cod de 5 cifre după ce utilizatorul a ales ce produse dorește să cumpere, și de a pune la dispoziție utilizatorilor din afara personalului, cei ce frecventează această cantină, un sistem de recomandare produse pe baza celor cumpărate anterior.

Scopul aplicației pentru administratorii în cadrul componentei interfeței, adică cei ce fac parte din cadrul personalului este cel de a le ușura munca și de a le oferi un sistem de gestionare a produselor cantinei. Principalele funcționalități pentru acest tip de utilizator sunt următoarele:

- magazinul virtual propriu zis în care se selectează produsele ce doresc a fi cumpărate și generarea unei chitanțe
- generarea unei chitanțe pe baza unui cod de 5 cifre generat automat în momentul în care utilizatorii au ales anumite produse ce doresc a fi cumpărate
- adăugarea de produse noi
- adăugarea de utilizatori noi
- crearea unui meniu pentru ziua curentă

Obiectiv

Soluția pentru această problemă, descrisă în această lucrare a fost dezvoltarea unei aplicații Web și mobile pentru sistemele de operare Android, respectiv iOS care este ușor de utilizat și oferă funcționalități similare unui magazin virtual, aplicația fiind dezvoltată pentru două categorii de utilizatori. Prima categorie este cea a administratorilor în cadrul

componentei interfețe, persoane ce fac parte din personalul cantinei iar cea de-a doua categorie este cea a utilizatorilor, adică a studenților și a profesorilor și a altor categorii de clienți care frecventează această cantină/restaurant.

Un magazin virtual/magazin online este un website destinat vânzării de produse și de servicii. În cele mai multe dintre cazuri, magazinul online reprezintă o platformă pe care sunt adăugate produse. În general, magazinele online permit cumpărătorilor să utilizeze funcții de căutare pentru a găsi modele, mărci sau elemente specifice. Clienții online trebuie să aibă acces la Internet.

Una dintre cele mai vechi forme de tranzacționare desfășurate online a fost procesarea tranzacțiilor online a IBM dezvoltată în anii 1960 și a permis prelucrarea tranzacțiilor financiare în timp real. Apariția de cumpărături on-line, după cum știm astăzi s-a dezvoltat odată cu apariția Internetului. Inițial, această platformă funcționa doar ca un instrument publicitar pentru companii, oferind informații despre produsele sale. Aceasta a avansat rapid de la acest utilitar simplu la tranzacția reală de cumpărături online datorită dezvoltării de pagini Web interactive și de transmisii securizate.

Magazinul virtual are ca scop prezentarea și vinderea unor produse. Majoritatea comercianților mari, corporațiile sau brandurile, își dezvoltă propriile departamente web și își creează magazine virtuale.

Funcționalitățile dezvoltate în această aplicație sunt similare cu cele ale unor platforme precum „FoodPanda” ce conține informații sigure și a fost dezvoltată cu scopul de a informa utilizatorii despre anumite produse, pentru a satisface poftele utilizatorilor.

Contribuții

Scopul principal al acestei aplicații este cel de a pune la dispoziție, atât utilizatorilor cât și administratorilor din cadrul interfeței un sistem și anumite funcționalități ce fac posibile reducerea timpului de așteptare și realizarea comenzilor de anumite produse într-un ritm mai alert.

Pentru utilizatori, aplicația pune la dispoziție o interfață web atractivă, dar și o interfață mobile pentru două din cele mai cunoscute sisteme de operare din lume (Android și iOS) și oferă diverse funcționalități precum: afișarea meniului din ziua curentă ce conține informații despre fiecare produs în parte ce este pus în vânzare, precum și un sistem de recomandare

produse pe baza celor cumpărate anterior. De asemenea utilizatorul își poate alege produsele preferate, iar apoi, prin acționarea unui buton, acestuia îi este generat un cod de 5 cifre, care poate fi folosit ulterior atunci când acestuia îi vine rândul la casă, astfel se reduce timpul de așteptare și comenzile se realizează într-un ritm mai alert.

Pe de altă parte, pentru administratorii din cadrul interfeței, aplicația oferă un sistem de selecție a produselor și de generare a unui bon pe baza produselor selectate din cadrul meniului din ziua curentă, dar și un sistem prin care, introducându-se codul de 5 cifre specificat mai sus, se generează un bon automat, fără a mai selecta manual produsele dorite. O altă funcționalitate importantă este cea de creare a meniului pentru ziua curentă, în care, pentru fiecare categorie de produse, se alege din lista de produse disponibile și se adaugă în meniul respectiv. De asemenea, administratorii pot vizualiza care produse mai sunt disponibile, și pentru fiecare produs, cantitatea care mai este disponibilă. Tot aceștia pot adăuga produse noi, pot schimba anumite informații asupra unor anumite produse, pot vedea istoricul vânzărilor și pot căuta informații despre anumite produse.

Cerințe funcționale

În momentul în care un utilizator va accesa site-ul Web sau își va descărca și instala aplicația, va apărea pagina de *Login*. Utilizatorul se va putea conecta folosind această pagină în cazul în care are deja făcut un cont introducând adresa de email, respectiv parola.

În cazul în care utilizatorul și-a uitat parola, acesta poate acționa atât în interfața mobile cât și în cea Web butonul *Ai uitat parola / Forgot password* care va duce către pagina de recuperare parolă.

În cadrul acestei pagini utilizatorul va introduce adresa de email către care va fi trimis un link de resetare parolă. Aici utilizatorul va trebui să introducă de două ori aceeași parolă, astfel încât aceasta să poate să fie modificată. După ce utilizatorul a introdus aceeași parolă de două ori, acesta va fi redirecționat către pagina de *Login* pentru a se putea autentifica.

Atât pentru interfața mobile cât și pentru cea Web, în cazul în care utilizatorul nu are un cont existent se poate naviga către pagina de *Register*, folosind butonul de acțiune *Înregistrare / Sign up* – care va duce către pagina de înregistrare, unde utilizatorul își va crea propriul cont în baza de date. Contul va fi completat în funcție de informațiile despre fiecare utilizator.

Odată conectați, utilizatorii vor avea un meniu cu produsele disponibile din ziua curentă, ce conține informații precum: *nume, descriere, preț, gramaj*. Utilizatorul poate arunca o privire asupra meniului, urmând ca la casă să comande ceea ce dorește, sau prin intermediul unui buton ar putea să-și selecteze produsele dorite, iar la final să genereze un cod de 5 cifre, urmând apoi când îi vine rândul să comunice codul cu doamnele de la casă și va fi generată automat un bon. De asemenea utilizatorul dispune și de un buton de recomandare de produse, ce are în spate un algoritm, care pe baza produselor cumpărate anterior, va afișa produsele similare cu cele anterioare.

Pe de altă parte, pentru administratorii din cadrul interfeței, după ce s-au conectat, aceștia vor dispune de magazinul virtual propriu-zis în care vor putea selecta produsele ce sunt dorite și urmează a fi cumpărate de utilizatorii precum profesori și studenți, sau așa cum am zis mai sus vor putea introduce codul de 5 cifre comunicat de către ceilalți utilizatori și se va genera o chitanță automat. În cadrul acestei pagini vor fi afișate doar produsele care mai sunt disponibile.

A doua funcționalitate de care dispune această categorie de utilizatori este aceea de a introduce un produs nou în baza de date, prin completarea unor câmpuri în care sunt adăugate informații despre produsul respectiv. Printre alte funcționalități se enumeră cea de adăugare personal nou în baza de date, în care sunt completate informații despre persoana ce urmează să i se atribue această funcție și crearea de meniu pentru ziua curentă în care pentru fiecare categorie de produse putem selecta din produsele disponibile din ziua curentă și care fac parte din această categorie de produse. După ce meniul a fost creat, acesta va putea fi vizualizat de către utilizatori. Printre alte funcționalități ale aplicației se enumeră cea de vizualizare a informațiilor despre produs, cea de schimbare a informațiilor unui produs în cazul în care se produc modificări, precum și cea de vizualizare a istoricului vânzărilor dintr-o anumită zi, ce conține informații precum: numărul chitanței, numele produselor, precum și cantitățile produselor vândute din acea zi și prețul total a unei comenzi.

1. CAPITOLUL I. Tehnologii folosite:

Pentru dezvoltarea aplicației „Cantina Gaudeamus” au fost folosite următoarele tehnologii:

- Flutter
 - Sistemul de operare Android
 - Sistemul de operare iOS
- Web API ASP.NET Core
- Angular
- MongoDB

1.1. Flutter framework

Flutter framework este un dezvoltator SDK mobil „open-source”, dezvoltat de Google, care este folosit împreună cu limbajul de programare *Dart*, limbaj orientat-obiect. Este o tehnologie relativ nouă și poate fi folosit pentru a dezvolta aplicații de înaltă performanță pentru Android și iOS cu aspect nativ dintr-un singur cod sursă de bază. Scopul este de a permite dezvoltatorilor să furnizeze aplicații performante pe diferite platforme.

Unul dintre motivele principale pentru care am folosit această tehnologie este portabilitatea, ceea ce înseamnă că aplicația poate fi dezvoltată folosind un singur cod sursă de bază pentru Android, cât și pentru iOS, al doilea motiv ar fi că flutter ne ajută să dezvoltăm aplicații de înaltă performanță pe diferite platforme (Android, iOS, Desktop), iar al treilea motiv, dar nu cel din urmă ar fi gradul ridicat de integrare cu alte limbaje de programare.

1.1.1. Sistemul de operare Android

Android este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance.

Android permite dezvoltatorilor să scrie un cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google. Aplicațiile scrise în C și în alte limbaje pot fi compilate în cod mașină ARM și executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google.

Lansarea platformei Android la 5 noiembrie 2007 a fost anunțată prin fondarea Open Handset Alliance. un consorțiu de 48 de companii de hardware, software și de telecomunicații, consacrat dezvoltării de standarde deschise pentru dispozitive mobile. Google a lansat cea mai mare parte a codului Android sub licența Apache. Google a dezvoltat și alte sisteme de operare bazate pe Android: Wear OS pentru ceasuri inteligente, Android TV pentru SmartTV și Android Auto pentru autoturisme.

Începând cu 21 octombrie 2008, Android a fost disponibil ca Open Source. Google a deschis întregul cod sursă, care anterior era indisponibil, sub licența Apache. Sub licența Apache producătorii sunt liberi să adauge extensii proprietare, fără a le face disponibile comunității open source. În timp ce contribuțiile Google la această platformă se așteaptă să rămână open source, numărul versiunilor derivate ar putea exploda, folosind o varietate de licențe. De asemenea, de-a lungul anilor, au fost dezvoltate numeroase versiuni de android, așa cum se poate vedea în Fig. 1.

Versiuni Android:

Versiune		Nume de cod	Data lansării	API level	Distribuție
10		Queen's Cake	03 septembrie 2019	29	1,2%
9		Pie	06 august 2018	28	35,6%
8.0-8.1		Oreo	aug 2017 (API 26, Android 8.0), dec 2017 (API 27, Android 8.1)	26, 27	22,8%
7.0-7.1.2		Nougat	Google I/O 2016	24, 25	13,8%
6.0-6.0.1		Marshmallow	Google I/O 2015	23	11,5%
5.0-5.1.1		Lollipop	Google I/O 2014	21, 22	9,1%
4.4-4.4.4		KitKat	31 oct 2013 (Android 4.4), 5 dec 2013 - 19 iun 2014 (Android 4.4.1-4.4.4)	19	4,0%
4.1-4.3		Jelly Bean (Patch de Securitate NESUPORTAT)	9 iul 2012 - 9 oct 2012 (API 16, Android 4.1-4.1.2), 13 nov 2012 - 11 feb 2012 (API 17, Android 4.2-4.2.2), 24 iul 2013 - 3 oct 2013 (API 18, Android 4.3-4.3.1)	16,17,18	1,5%
4.0-4.0.4		Ice Cream Sandwich (Patch de Securitate NESUPORTAT)	18 oct 2011 - 28 nov 2011 (API 14, Android 4.0-4.0.2), 16 dec 2011 - 29 mar 2012 (API 15, Android 4.0.3-4.0.4)	14, 15	0.2%
2.3-2.3.7		Gingerbread (Patch de Securitate NESUPORTAT)	6 dec 2010 - ian 2011 (API 9, Android 2.3-2.3.2), 9 feb 2011 - 21 sep 2011 (API 10, Android 2.3.3-2.3.7)	09, 10	0.2%

Fig. 1 Versiuni Android

Sistemul de operare Android cuprinde o gamă largă de caracteristici și funcționalități printre care se enumeră următoarele:

- o interfață ce atrage utilizatorul;
- *stocare*, ce folosește SQLite, o bază de date relațională ce permite utilizarea eficientă a resurselor;
- *conectivitatea* prin diverse modalități, printre care se enumeră: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, 4G, NFC, WiMAX;
- *mesagerie* SMS și MMS;
- *navigarea pe Internet* bazată pe motorul open source pentru navigare WebKit împreună cu motorul JavaScript de la Chrome V8 suportând HTML5 și CSS3.
- *multi-touch*, care suportă posibilitatea de contact în mai multe puncte concomitent;
- *multi-tasking*; GCM(Google Cloud Messaging) permițând dezvoltatorilor expedierea de date de dimensiuni reduse, în lipsa unei soluții de sincronizare proprietară.
- *Android Beam*, prin care utilizatorii partajează conținut instant prin apropierea dispozitivelor respective.
- *WiFi direct*, care permite interconectarea între diverse dispozitive, de la o distanță relativ mare, având de asemenea o lățime de bandă mare;
- *multi-tasking*; GCM(Google Cloud Messaging) permițând dezvoltatorilor expedierea de date de dimensiuni reduse, în lipsa unei soluții de sincronizare proprietară.

1.1.2. Sistemul de operare iOS

iOS este un sistem de operare pentru dispozitive mobile creat și dezvoltat de Apple Inc. Este sistemul de operare care alimentează în prezent multe dispozitive ale companiilor mobile, inclusiv iPhone și iPod Touch. Este al doilea cel mai popular sistem de operare mobil la nivel mondial după Android. Este baza altor sisteme de operare realizate de Apple Inc, cum ar fi iPadOS, tvOS și watchOS.

Dezvoltat inițial în 2007 pentru iPhone de primă generație, iOS a fost extins de atunci pentru a sprijini alte dispozitive Apple, cum ar fi iPod Touch și iPad. Începând cu luna martie 2018, App Store Apple conține mai mult de 2.1 milioane de aplicații iOS, dintre care 1 milion sunt native pentru iPad. Aceste aplicații mobile au fost descărcate de peste 130 de miliarde de ori.

Interfața de utilizator iOS se bazează pe o manipulare directă, folosind gesturi *multi-touch*. Elementele de control ale interfeței constau în glisiere, întrerupătoare și butoane. Interacțiunea cu sistemul de operare include gesturi precum *swipe*, *tap*, *pinch* și *pinch invers*, toate având definiții specifice în contextul sistemului de operare iOS și al interfeței sale *multi-touch*. Apple a fost lăudat în mod semnificativ pentru încorporarea funcțiilor de accesibilitate completă în iOS, permițând utilizatorilor cu dizabilități de vedere și auz să-și folosească corect produsele.

De asemenea, sistemul de operare iOS are numeroase caracteristici și funcționalități printre care se enumeră următoarele:

- *ecranul de pornire*, redat de SpringBoard, afișează pictogramele aplicației în partea de jos în care utilizatorul își poate fixa aplicațiile utilizate cel mai des.
- *centru de notificare*, care permite utilizatorilor să vizualizeze un istoric al notificărilor
- *accesibilitate*; iOS oferă diverse funcții de accesibilitate pentru a ajuta utilizatorii cu dizabilități de vedere și auz. O caracteristică principală este VoiceOver, ce oferă informații despre citirea vocii pe ecran, inclusiv butoane contextuale, pictograme și alte elemente de interfață pentru utilizator și permite acestuia să navigheze în sistemul de operare prin gesturi
- *multi-tasking*, *comutarea aplicațiilor*, *finalizarea sarcinilor*
- *Siri* care este un asistent personal inteligent integrat în iOS. Asistentul folosește interogări vocale și o interfață de utilizare a limbajului natural pentru a răspunde la întrebări, a face recomandări și a efectua acțiuni prin delegarea cererilor către un set de servicii Internet. Software-ul se adaptează la utilizările, căutările și preferințele limbajului individual al utilizatorilor, cu utilizare continuă. Rezultatele returnate sunt individualizate. *Siri* acceptă o gamă largă de comenzi ale utilizatorului, inclusiv efectuarea acțiunilor telefonice, verificarea informațiilor de bază, programarea evenimentelor, gestionarea setărilor dispozitivului, căutarea pe internet, navigarea în zone și găsirea informațiilor despre divertisment.
- *rețea de jocuri multiplayer* care permite utilizatorilor să invite prietenii să joace un joc, să înceapă un joc prin realizarea meciurilor, să urmărească realizările lor și să compare scorurile lor înalte pe un clasament.
- *platformă hardware* care are arhitectura ARM. Versiunile iOS 7 pot fi rulate numai pe dispozitive iOS cu procesare ARM pe 32 de biți.

1.2. Web API ASP.NET CORE

Serviciile web sunt servere web create special pentru a îndeplini nevoile unui site sau a unei aplicații. Programele client folosesc interfața de programare a aplicațiilor (Application Programming Interface – API) pentru a comunica cu serviciile web. General vorbind, un API dispune de un set de date și de acțiuni pentru a facilita interacțiunea dintre anumite programe ale calculatorului și le permite să realizeze schimb de informații între ele.

API reprezintă un set de definiții de sub-programe, protocoale și unele pentru programarea de aplicații software. Un API poate fi pentru un sistem web, sistem de operare, sistem de baze de date, Hardware sau biblioteci software.

REST (Representational state transfer) este o arhitectură software care definește un set de constrângeri care trebuie utilizate pentru crearea serviciilor Web. Serviciile web care se conformează stilului arhitectural REST, numit “*RESTful Web services*”, asigură interoperabilitatea între sistemele informatice de pe Internet. Serviciile web *RESTful* permit sistemelor solicitante să acceseze și să manipuleze reprezentările textuale ale resurselor Web, folosind un set uniform și predefinit de operații.

Un *RESTful API* este o interfață de programare a aplicațiilor care utilizează cereri *HTTP* pentru a obține, a actualiza, a șterge sau a face anumite operații cu anumite date. Un *API* pentru un site web este cod ce permite ca două programe software să comunice între ele.

Principalele motive pentru care am folosit Web API.NET CORE sunt următoarele:

- dezvoltatorul deține controlul asupra transmiterii și răspunsurilor la mesajele protocolului HTTP
- API-urile ASP.NET oferă un nivel ridicat de abstractizare cu ajutorul căruia dezvoltatorii pot crea API-uri web care pot încapsula *HttpMessageHandler*
- arhitectura API-urilor web este foarte ușoară, ceea ce le face o alternativă perfectă pentru dezvoltatori atunci când doresc să construiască aplicații pentru dispozitive cu lățime de bandă limitată
- este simplu, scalabil și robust, deoarece acceptă toate funcțiile MVC, cum ar fi rutarea, controlerele, rezultatele acțiunii, filtrul sau injecția de dependență

1.3. Angular

Angular este o platformă pentru construirea aplicațiilor client utilizând HTML și TypeScript. Angular este scris cu TypeScript și implementează funcționalități de bază și opționale ca un set de biblioteci pe care le importăm din aplicații.

Arhitectura unei aplicații Angular se bazează pe concepte fundamentale. Blocurile de bază sunt NgModules, care oferă un context de compilare pentru componente. NgModules colectează codul aferent de seturi funcționale. O aplicație Angular este definită de un set de NgModule. O aplicație are întotdeauna cel puțin un modul rădăcină.

Componentele definesc pagini web (vizualizări), care oferă funcționalități specifice care nu sunt legate direct de alte pagini web. Furnizorii de servicii pot fi integrați în componente ca și dependențe, ceea ce face codul sursă să fie modular, reutilizabil și eficient. Atât componentele, cât și serviciile sunt pur și simplu clase, cu decoratori care marchează tipul lor și oferă *metadata* care îi spun platformei cum să le folosească.

Componentele unei aplicații definesc de obicei mai multe pagini web, aranjate ierarhic. Angular oferă serviciul de rutare (*Router*) pentru a ne ajuta să definim căile de navigare între paginile web. Rutarea oferă funcții sofisticate de navigare în *browser*.

1.4. MongoDB

MongoDB este o bază de date ce stochează date *NoSql* (Not only sql), adică o bază de date ne-relațională în care informațiile sunt stocate în documente/colecții, ci nu în tabele. O astfel de bază de date permite stocarea unor volume foarte mari de informații într-un timp scurt, față de o bază de date *SQL*. O colecție de date MongoDB este similară ca și format cu o dată de tip dicționar ce se folosește în cadrul fișierelor .json.

Avantajul folosirii colecțiilor:

- cea mai naturală și productivă metodă de a lucra cu date
- suporta *array* și obiecte ca și valori
- permite scheme flexibile și dinamice

Exemplu de colecție *MongoDB*:

```
{
  "_id" : " ",
  "firstName" : "Vasile",
  "lastName" : "Lazar",
  "address" : {
    "street" : "Mihai Eminescu",
    "city" : "Galati",
    "number" : "361",
  },
  "hobbies":["tennis","football"]
}
```

Principalele motive pentru care am folosit această bază de date:

- este un limbaj de interogare bogat și expresiv, care ne permite să filtrăm și să sortăm după orice criteriu
- suport pentru agregări și alte cazuri moderne de utilizare, cum ar fi căutare în grafic și căutare de text
- întrebările sunt și ele la rândul lor sub format JSON, deci sunt ușor de compus
- asistență pentru *join* la interogări
- două tipuri de relații în loc de una: de referință și incorporare
- permite stocarea datelor în colecții sub mai multe forme (Array, Object)
- deoarece datele sunt stocate în colecții ce au un format asemănător unei date de tip dicționar ceea ce le face mult mai ușor de utilizat și datorită complexității timp și spațiu ale obținerii de soluții.
- fiecare document are o cheie primară cu ajutorul căreia este identificat unic în colecție
- timpul de interogare a unei baze de date NoSQL este scăzut comparativ cu cel al unei baze de date relaționale în momentul în care între colecții nu există relații
- are suport pentru o gamă largă de limbaje de programare

2. CAPITOLUL II. Arhitectura aplicației

În acest capitol vom pune în evidență arhitectura aplicației, principale module aplicației, structura și modul de comunicare între modulele aplicației.

Aplicația a fost dezvoltată astfel încât să ofere o soluție cât mai eficientă și ușor de utilizat de către cei ce o folosesc. Arhitectura este construită astfel încât să utilizeze resurse de memorie cât mai puține și care să ofere un timp de execuție, respectiv de încărcare cât mai scurt.

Aplicația a fost dezvoltată pentru partea de web și pentru cele mai cunoscute sisteme de operare *mobile* de la ora actuală (Android, iOS). Pentru a putea folosi această aplicație este nevoie de conexiune la Internet.

2.1. Structura

Așa cum se poate observa în Fig. 2 Arhitectura aplicației este una clasică, în care avem o interfață mobile pentru Android, respectiv iOS și web și o „poartă de acces” care face legătura între interfață și baza de date.

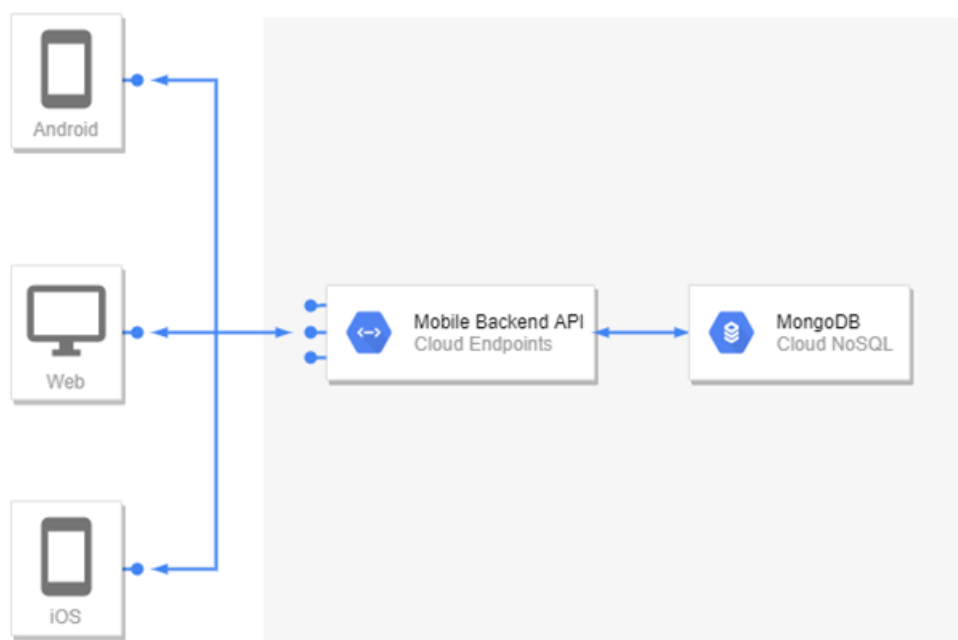


Fig. 2 Arhitectura aplicației

2.2. Modulele aplicației

În continuare vă voi prezenta modulele aplicației detaliat și ce conțin acestea, respectiv cum funcționează .

2.2.1. Modulul Interfață Web

Unul dintre cele trei module de interfață este cel pentru Web.

Culorile alese în cadrul acestei interfețe sunt albastru, alb și mov, deoarece albastrul este o culoare rece, movul este o culoare care provoacă senzația de calmare, iar albul simbolizează puritatea, liniștea, toate acestea la un loc tind să fie mai plăcute și mai relaxante.

La prima utilizare, utilizatorul este întâmpinat de pagina de *Login*, în care utilizatorul trebuie să introducă anumite date, acestea fiind adresa de email și parola. De precizat faptul că în cadrul acestei aplicații exista două categorii de utilizatori, aceștia fiind: utilizatori simpli (studenți, profesori, alte persoane ce frecventează cantina) și administratorul în cadrul componentei de interfață (casierile, persoane ce fac parte din personalul cantinei). Dacă datele introduse sunt corecte atunci poate se poate începe utilizarea aplicației.

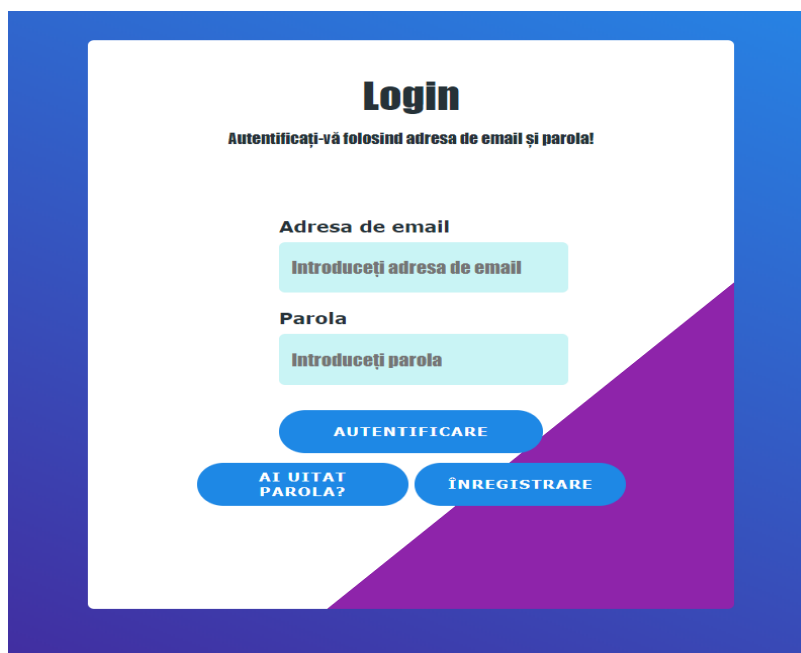


Fig. 3 Pagina de autentificare - web

În cazul în care utilizatorul și-a uitat parola, acesta poate acționa atât în interfața mobile cât și în cea Web butonul *Ai uitat parola / Forgot password* care va duce către pagina de

recuperare parolă, unde se introduce adresa de email, iar utilizatorul va primi ulterior pe adresa introdusă un mail cu parola.

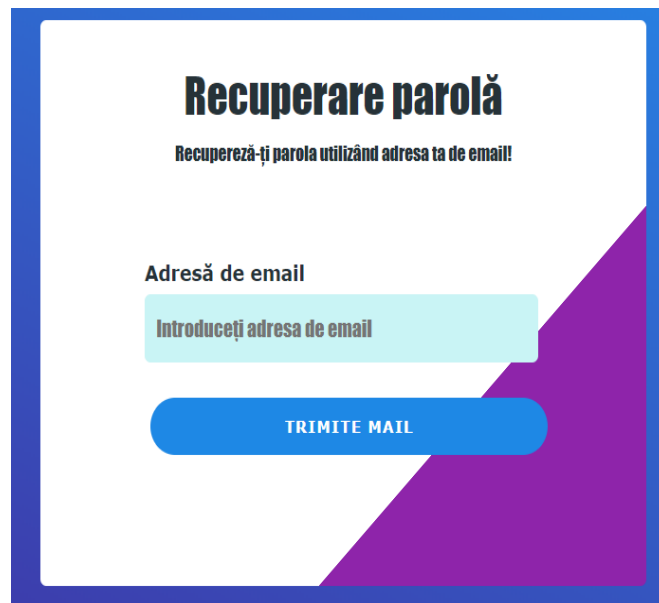


Fig. 4 Pagina de recuperare parolă - web

În Fig. 5 se poate observa pagina de resetare parolă. Aici utilizatorul va trebui să introducă de două ori aceeași parolă, astfel încât aceasta să poată să fie modificată. După ce utilizatorul a introdus aceeași parolă de două ori, acesta va fi redirecționat către pagina de *Login* pentru a se putea autentifica.



Fig. 5 Pagina de resetare parolă

În cazul în care utilizatorul nu are deja un cont, în partea de jos a paginii *Login* există un buton de acțiune ce redirecționează către pagina de *Register*. Principala funcționalitate în cadrul acestei pagini este aceea de înregistrare a unui utilizator nou prin completarea a trei câmpuri: *email*, *parola* și *confirmare parolă*. De precizat faptul că *parola* și *confirmare parolă* trebuie să coincidă și să fie identice.



Fig. 6 Pagina de înregistrare - web

Pentru utilizator, după ce acesta se conectează la aplicație, va fi întâmpinat de un ecran asemenea unui meniu, ce conține informații despre produsele disponibile din ziua curentă, informații precum: nume, descriere/ingrediente, preț. De asemenea fiecare produs este însoțit de două butoane. Primul buton este cel de adăugare produs, iar al doilea este cel de ștergere produs. În cadrul acestei pagini utilizatorul poate pur și simplu doar să se informeze să vadă ce produse sunt disponibile în ziua curentă, sau poate să aleagă din lista de produse disponibile.

În partea de jos a paginii se află un container ce conține informații despre prețul total al produselor ce au fost selectate și un buton *Generare cod*, care după cum spune, după ce utilizatorul a finalizat de selectat produsele ce dorește să le cumpere, acționând acest buton va fi generat un cod de 5 cifre care va apărea în partea de sus a paginii, acest cod poate fi folosit

ulterior în momentul în care utilizatorul ajunge la casă și nu mai trebuie să comunice casieritei ce produse dorește să cumpere, ci va comunica doar codul generat, care va genera un bond automat așa cum se descrie în partea ce urmează.

Codul dumneavoastră este: 9957

Meniu 17-06-2020

RECOMANDARE PRODUSE

Ciorbe si supe / Soups

Ciorbă de perișoare din carne de vită	Conține smântână, țelină, produs decongelat	3.8 lei	-	1	+
Cremă de legume cu broccoli și crutoane	Conține glute, țelină, produs decongelat	3.1 lei	-	2	+

Garnituri / Side dishes

Cartofi prăjiți cu mozzarella	Conține mozzarella	3.1 lei	-	0	+
Cartofi prăjiți	Conține produs decongelat	2.3 lei	-	1	+
Pilaf sărbesc	Conține produs decongelat	1.1 lei	-	1	+

Felul II

Varză cu carne de porc și mămăligă	Conține produs decongelat	4.8 lei	-	0	+
------------------------------------	---------------------------	---------	---	---	---

Preț total

13.4 lei

CUMPARĂ

Fig. 7 Meniu utilizator 1

Pulpe de pui umplute și piure de cartofi	Conține unt, produs decongelat	7.1 lei	-	0	+
Muşchiuleț de porc la grătar	Conține produs decongelat	8.9 lei	-	0	+
Bulz cu brânză și ou la cuptor	Conține une, brânză, ou	7.4 lei	-	0	+

Desert / Deserts

Clătite cu ciocolată	Conține gluten	1.4 lei	-	0	+
Plăcintă cu brânză	Conține glute, brânză, ouă	1.9 lei	-	0	+
Boema	Conține gluten, ouă	2.1 lei	-	0	+

Salate / Salads

Salată de varză		1 lei	-	0	+
Salată de varză albă		1 lei	-	0	+

Paine / Bread

Chifle		0.3 lei	-	0	+
--------	--	---------	---	---	---

Preț total

13.4 lei

CUMPARĂ

Fig. 8 Meniu utilizator

Pentru administratorul în cadrul componentei interfeței, după ce acesta se conectează la aplicație, va fi întâmpinat de un ecran ce conține în partea de sus(header) numele aplicației și două butoane: *Logout* și *Users* ca în Fig. 9. Acționarea butonului *Logout* va redirecționa administratorul către pagina de *Login*, iar acționarea butonului *Users* va redirecționa administratorul către pagina utilizatorului.



Fig. 9 Header

De asemenea, utilizatorii din cadrul interfeței sunt de două categorii: cei ce se ocupă cu gestionarea produselor și crearea meniurilor, precum și cei ce fac parte din personalul ce se ocupă cu vânzarea produselor. În partea stânga a paginii avem o bară de navigare ce permite redirecționarea către alte pagini ale aplicației, acestea fiind: *Magazin*, *Creare meniu*, *Adăugare produs*, *Detalii produse*, *Actualizare produse*, *Adăugare admin*, *Istoric vânzări*. Bara de navigare pentru cele două categorii de administratori din cadrul interfeței se poate observa în **Eroare! Fără sursă de referință..**



Fig. 10 – Bara de navigare 1



Fig. 11 – Bara de navigare 2

După ce realizează conectarea, administratorul este întâmpinat de pagina *Magazin* sau de *Creare meniu*, în funcție de tipul acestuia din bara de navigare a aplicației. Această pagina cuprinde o listă cu produsele disponibile în ziua curentă. Fiecare produs are informații despre: nume, preț profesor, preț student, descriere/ingrediente, precum și câte produse de acest fel sunt disponibile la momentul respectiv. De asemenea fiecare produs conține două butoane, unul de adăugare produs și altul de ștergere produs. Acționarea acestora adaugă, respectiv șterge produsul selectat din lista celor ce urmează a fi cumpărate.

Rolul administratorilor interfeței în cadrul acestei pagini este de a selecta produsele ce doresc a fi cumpărate de către utilizatori și de a genera chitanța. Însă pentru a realiza acest lucru am pus la dispoziție o metoda mai simplă și mult mai rapidă, și aceea de a introduce în câmpul de sus al paginii un cod de 5 cifre care a fost generat în pagina utilizatorilor. Acest cod va selecta automat produsele dorite de către utilizator și va genera chitanța în funcție de produsele selectate. Acest lucru se face prin acționarea butonului *Arata meniul*. În partea dreaptă din jos a paginii, se află un container ce conține informații despre prețul total în urma selectării produselor și un buton *Cumpără* ce generează automat chitanța. Mai jos se poate vedea tot ceea ce am descris până aici.

Cod pentru generare meniul automat:

introduceți cod

ARATĂ MENIU

☐ Profesor/alți clienți

Giorbă de perisoare din carne de vită

Pret profesor: 4.5

Pret student: 3.8

Conține smântână, țelină, produs decongelat

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Cremă de legume cu broccoli și crutoane

Pret profesor: 3.7

Pret student: 3.1

Conține glute, țelină, produs decongelat

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Giorba de burta

Pret profesor: 7.6

Pret student: 6.8

Conține smantana

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Cartofi prăjiți cu mozzarella

Pret profesor: 3.7

Pret student: 3.1

Conține mozzarella

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Cartofi prăjiți

Pret profesor: 2.8

Pret student: 2.3

Conține produs decongelat

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Pilaf sârbesc

Pret profesor: 1.3

Pret student: 1.1

Conține produs decongelat

Rămase:11 / Selectate: 1

ADAUGĂ

ȘTERGE

Clătite cu ciocolată

Pret profesor: 1.7

Pret student: 1.4

Conține gluten

Rămase:12 / Selectate: 0

ADAUGĂ

ȘTERGE

Piacintă cu brânză

Pret profesor: 2.3

Pret student: 1.9

Conține glute, brânză, ouă

Rămase:12 / Selectate: 0

ADAUGĂ

ȘTERGE

Boema

Pret profesor: 2.5

Pret student: 2.1

Conține gluten, ouă

Rămase:12 / Selectate: 0

ADAUGĂ

ȘTERGE

Salată de varză

Pret profesor: 1.2

Pret student: 1

Rămase:12 / Selectate: 0

ADAUGĂ

ȘTERGE

Salată de varză albă

Pret profesor: 1.2

Pret student: 1

Rămase:12 / Selectate: 0

ADAUGĂ

ȘTERGE

Chifle

Pret profesor: 0.5

Pret student: 0.3

Rămase:12 / Selectate: 0

Preț Total

20.2 lei

CUMPĂRĂ

Preț Total

20.2 lei

CUMPĂRĂ

Fig. 12 Meniu casierie

Al doilea buton din bara de navigare din partea stângă a paginii este cel de *Creare meniu*, în care după cum spune și numele, administratorul poate crea meniul pentru ziua curentă sau pentru altă zi. Această pagină conține o serie de câmpuri ce sunt necesare a fi completate de către administrator. Primul câmp este cel de introducere a datei în care se dorește să fie afișat meniul. În următoarele câmpuri pentru fiecare categorie de produse, se pot adăuga produsele dorite și numărul de produse disponibile de acest tip. Categoriile de produse sunt: *Ciorbe*, *garnituri*, *Felul II*, *salate*, *apă*, *etc.* De asemenea, pentru fiecare categorie de produse există și un buton de *Adăugare produs*, care în momentul în care este acționat se poate adăuga un nou produs pentru categoria respectivă. La final, după ce au fost selectate produsele, se poate crea meniul, acționând butonul *Creare meniu*, care va crea automat meniul ce va putea fi vizualizat de către utilizatori în ziua curentă. Pagina de creare meniu se poate vedea în Fig. 13.

Fig. 13 Creare meniu

Al treilea buton din bara de navigare din partea stângă a paginii este cel de *Adăugare produs*, care ne va redirecționa către pagina în care vrem să adăugăm un produs nou în lista de produse deja disponibile, care poate fi folosit în momentul în care în cadrul cantinei a apărut un produs nou și se dorește a fi pus la cumpărare. Această pagină conține anumite câmpuri ce

trebuie completate de administrator. Aceste câmpuri sunt: *nume produs*, *categorie produs*, *preț profesor*, *preț student*, *descriere*, *gramaj*. În momentul în care câmpurile au fost completate, produsul va fi adăugat automat în baza de date. Pagina de *Adăugare produs* poate fi văzută în Fig. 14.

Nume produs

Categorie produs

Preț profesor

Preț student

Gramaj

Descriere

ADAUGA PRODUS

Fig. 14 Adăugare produs

Cel de-al patrulea buton din bara de navigare este cel de detalii produse, ce oferă un serviciu de căutare a produselor din lista totală de produse și conține o listă cu toate produsele, precum și informațiile fiecărui produs așa cum se poate vedea în Fig. 15.

Nume	Categorie	Pret profesor	Pret student	Greutate	Descriere
Ciorbă de perișoare din carne de vită	Ciorbe si supe / Soups	4.5 lei	3.8 lei	400 gr	Conține smântână, țelină, produs decongelat
Cremă de legume cu broccoli și crutoane	Ciorbe si supe / Soups	3.7 lei	3.1 lei	300 gr	Conține glute, țelină, produs decongelat
Chifteluțe din ciuperci	Felul II	4.6 lei	3.8 lei	130 gr	Conține gluten
Clătite cu ciocolată	Desert / Deserts	1.7 lei	1.4 lei	200 gr	Conține gluten
Plăcintă cu brânză	Desert / Deserts	2.3 lei	1.9 lei	100 gr	Conține glute, brânză, ouă

Fig. 15 Detalii produse

Cel de-al cincilea buton din bara de navigare este cel de *Actualizare produse*. Acesta poate fi folosit, în special, în cazul în care vrem să modificăm informațiile despre un anumit produs. Se completează doar câmpurile ce se doresc a fi modificate. În cazul în care nu va fi modificată nicio informație atunci se va afișa o eroare. Această pagina arată ca în figura ce urmează.

Nume produs
Cartofi prăjiți cu mozzarella

Preț profesor
3.5

Preț student
Preț

Gramaj
Introduceți gramaj

Descriere
Descriere / Ingrediente

ACTUALIZEAZA
PRODUS

Fig. 16 Actualizare produs

Cel de al șaselea buton (Fig. 17) din bara de navigare este cel în care administratorul trebuie să introducă o adresă de email, iar persoana ce are acea adresă de email va primi la rândul ei rolul de administrator în cadrul interfeței.

Adresa de email
example.com@gmail.com

ADAUGĂ ADMIN

Fig. 17 Adăugare admin

Ultimul, dar nu cel din urmă buton din bara de navigare este cel de *Istoric vânzări*. În cadrul acestei pagini utilizator trebuie să aleagă o dată din care dorește să obțină informațiile despre vânzările din ziua respectivă. Informațiile ce vor fi afișate pentru fiecare vânzare din

ziua selectată sunt: codul chitanței, numele produselor ce au fost vândute, precum și numărul de produse vândute și prețul total pentru fiecare vânzare/chitanță în parte. Istoricul arată așa cum puteți vedea în Fig. 18.

Selecți data:

06/17/2020

ARATĂ ISTORIC

Cod chitanță	Nume produs x cantitate	Preț total
54003393	Cartofi prăjiți x 1 Cartofi prăjiți cu mozzarella x 1 Ciorbă de perișoare din carne de vită x 1 Cremă de legume cu broccoli și crutoane x 1 Pilaf sârbesc x 1	13.4 lei
7982965	Bulz cu brânză și ou la cuptor x 1 Chifteluțe din ciuperci x 1 Clătite cu ciocolată x 1 Mușchiuleț de porc la grătar x 1 Pulpe de pui umplute și piure de cartofi x 1 Salată de varză x 1	29.6 lei
91089802	Apă plată borsec x 2 Boema x 1 Cartofi prăjiți cu mozzarella x 1 Clătite cu ciocolată x 1 Salată de varză x 1 Varză cu carne de porc și mămăligă x 1	16.4 lei
84089192	Ciorbă de perișoare din carne de vită x 1 Cremă de legume cu broccoli și crutoane x 1 Mușchiuleț de porc la grătar x 1 Pilaf sârbesc x 1 Pulpe de pui umplute și piure de cartofi x 1	24 lei
53448385	Apă plată borsec x 1 Bulz cu brânză și ou la cuptor x 1 Cartofi prăjiți cu mozzarella x 1 Chifle x 1 Ciorbă de perișoare din carne de vită x 1 Clătite cu ciocolată x 1 Salată de varză x 1 Varză cu carne de porc și mămăligă x 1	23.8 lei

Fig. 18 Istoricul produselor

2.2.2. Modulul interfața Android și iOS

Acest modul a fost dezvoltat doar pentru utilizatori, nu și pentru administratorii în cadrul componentei interfeței. Paginile din această componentă sunt similare cu cele descrise în 2.2.1. Paleta de culori aleasă pentru această componentă au fost movul și albastrul, ce tind să fie mai plăcute și mai relaxante. La fel ca și la modulul de interfață web, aici avem o pagină de *Login*, în care atât utilizatorii cât și administratorii vor introduce numele și parola corecte, după care vor fi redirecționați către pagina unde se află meniul, în cazul în care aceștia au introdus datele lor corecte. În caz contrar, va fi afișat un mesaj de eroare. Pagina de *Login* din cadrul acestui modul poate fi vizualizată în Fig. 19 Pagina de autentificare - mobile de mai jos.

Fig. 19 Pagina de autentificare - mobile

Cum am menționat și în 2.2.1, există și aici un buton în care utilizatorul va fi redirecționat către pagina de *Register*, în cazul în care acesta nu are un cont existent în baza de date. De asemenea există și un buton de *Forgot password* în cazul neplăcut în care utilizatorul și-a uitat parola și dorește să o recupereze. Paginile de *Register*, respectiv *Forgot password* pot fi vizualizate în Fig. 20 și Fig. 21.

Fig. 20 Pagina de înregistrare mobile

Fig. 21 Pagina de recuperare parol mobie

După ce utilizatorul s-a autentificat cu succes pe aplicația din cadrul interfeței mobile, acesta va fi întâmpinat de un meniu, ce conține informații despre produsele disponibile spre vânzare din ziua curentă. În partea de sus a paginii utilizatorul dispune de un buton de deconectare-*Logout* și un text ce conține informații despre data din care este afișat meniu. Informațiile din cadrul meniului sunt: numele produsului, prețul produsului, descriere produs și numărul de produse selectate. De asemenea utilizatorul dispune de 2 butoane; unul de adăgare produs și altul de ștergere produs. Deasupra meniului utilizatorul dispune de un buton de recomandare produse. Rolul acestuia este de produsele ce au mai fost cumpărate anterior de către utilizator. În partea de jos a paginii se află un text cu informații despre prețul total, precum și un buton de *Buy*. Acționarea acestui buton conduce la generarea unui cod de 5 cifre care va fi afișat ulterior deasupra meniului. Tot ce am desris în cadrul acestei pagini se poate vedea în Fig. 22, respectiv Fig. 23.



Fig. 22 Pagina de meniu a utilizatorului



Fig. 23 Pagina de meniu a utilizatorului

2.3. Modulul bază de date

Baza de date are un rol foarte important în cadrul unei aplicații, rolul acesteia fiind cel de stocare a informațiilor și de asigurare a integrității datelor. Pentru acest modul am ales o bază de date *NoSQL MongoDB*, deoarece datele sunt stocate în colecții ce au un format asemănător unei date de tip dicționar ceea ce le face mult mai ușor de utilizat și datorită complexității timp și spațiu ale obținerii de soluții. În cadrul acestei aplicații am folosit platforma cloud MongoDB Atlas, ce permite stocarea bazelor de date MongoDB în suport online.

Prima colecție este cea a utilizatorilor - *userlist*. În cadrul acestei colecții, în fiecare document sunt stocate informații despre fiecare utilizator în parte; pentru fiecare utilizator avem un identificator unic *_id* de tipul *int32*, *email* și *password* care sunt de tip *string*, ce reprezintă adresa de email al fiecărui utilizator în parte, respectiv parola. De menționat faptul că *password* este un hash SHA-256 de lungime fixă de 64 caractere. De asemenea pentru fiecare utilizator mai avem *type* și *role* de tip *string* ce reprezintă tipul utilizatorului, respectiv rolul acestuia, precum și un *token* care este un *string* de dimensiune mare.

Cea de-a doua colecție este cea a produselor - *products*. În cadrul acestei colecții sunt stocate informații despre fiecare produs în parte; de asemenea și pentru acestea avem un identificator unic *_id* de tip *int32*. Fiecare document mai conține și următoarele câmpuri: *name*, *category*, *description* ce sunt de tip *string* și conțin informații despre numele, categoria, respectiv descrierea produsului și *professorPrice*, *studentPrice*, *weight* ce sunt de tip *int*, respectiv *double* și conțin informații despre prețul pentru profesori, cel pentru student și gramajul.

Cea de-a treia colecție este cea a meniurilor - *menus*. Pentru fiecare document din cadrul acestei colecții avem un identificator unic *_id* de tip *int*, *dateMenu* de tip *Date* ce reprezintă data în care a fost creat meniul respectiv și *productsIdAndAmounts* care este de tipul *Object* și conține informații sub forma unui dicționar în format JSON, în care cheia reprezintă id-ul produsului, iar valoarea reprezintă cantitatea disponibilă pentru produsul respectiv în acea zi.

Cea de-a patra colecție este cea a comenzilor – *orders*. La fel ca la colecțiile prezentate mai sus, și pentru această colecție avem un identificator unic *_id* de tip *int*. De asemenea mai avem următoarele: *idUser* de tip *int* ce reprezintă id-ul utilizatorului pentru care a fost generat acest cod; *idProducAndAmount* de tip *Object*, ce reține informații cantitatea ce se dorește a fi cumpărată pentru fiecare produs în parte; *code* de tip *int32* ce reprezintă codul generat

automat care mai târziu poate fi folosit pentru achiziționarea de produse; *date* de tip *Date* ce reține data în care a fost generată comanda respectivă și *totalPrice* de tip *double* ce reprezintă prețul total asociat comenzii.

Ultima colecție este cea a istoricului comenzilor – *history*. Acest document conține un identificator unic *_id* de tip *int32* de 6 cifre, ce reprezintă numărul chitanței. De asemenea acesta reține data în care a fost generată această chitanță – *date*, numele produselor precum și cantitatea acestora într-un obiect sub format JSON – *nameProductsAndAmounts*, precum și prețul total al acestora – *totalPrice*. Schema bazei de date arată după cum se poate vedea în Fig. 24 de mai jos.

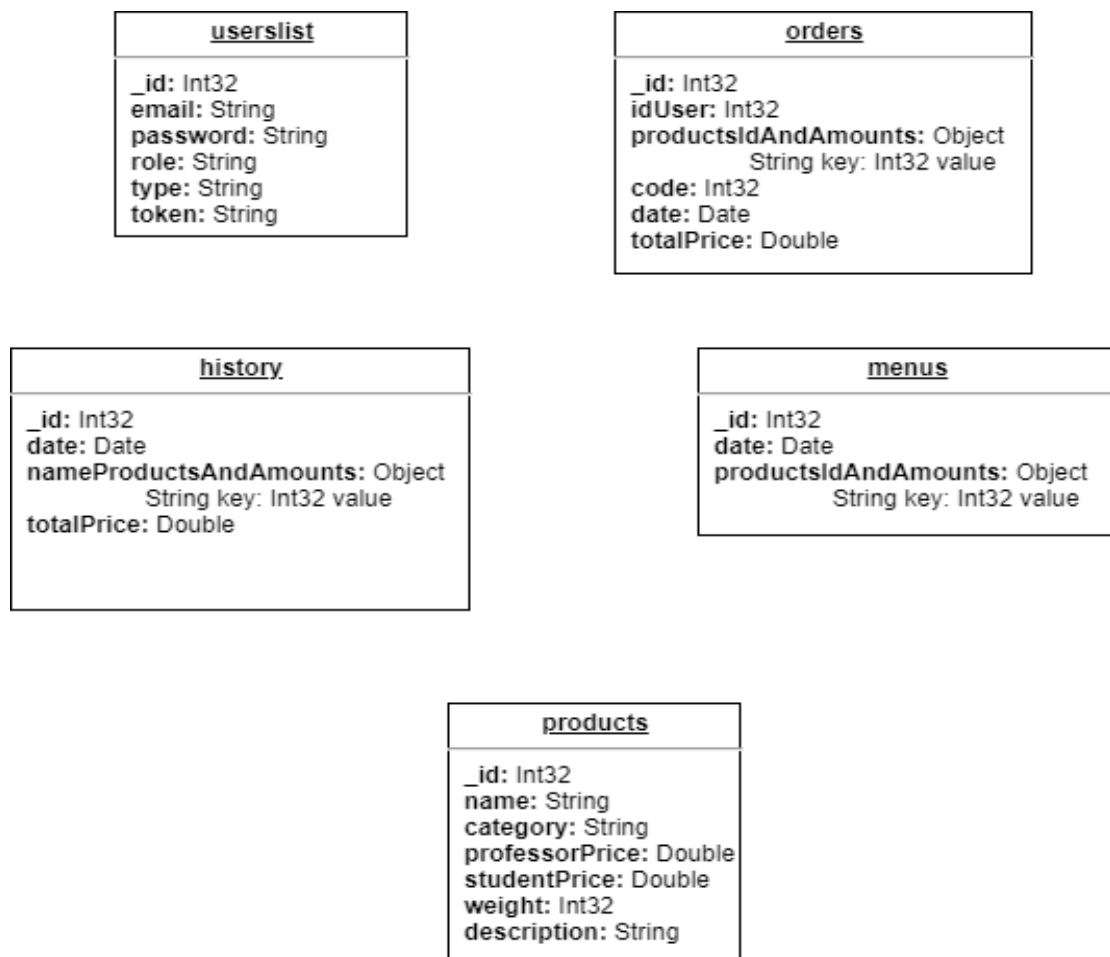


Fig. 24 Schema bazei de date

2.4. Modulul API Gateway

Pentru implementarea acestui modul am folosit modelul de lucru *ASP.NET CORE* împreună cu limbajul de programare C#. Principalul motiv pentru care am ales să lucrez cu această tehnologie este cel că *ASP.NET CORE* este unul dintre cele mai documentate medii de lucru, având un suport detaliat și multe pachete pre-implementate ce oferă funcționalități complexe în doar câteva linii de cod.

În cadrul acestui modul codul sursă este împărțit în *Controllere*, *Modele* și *Servicii*. Fișierele cu codul sursă sunt structurate așa cum se poate vedea în Fig. 25.

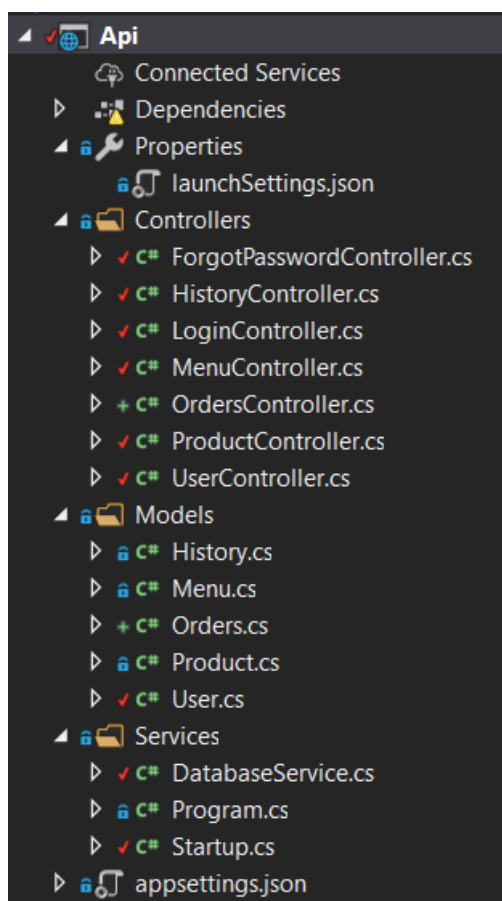


Fig. 25 Structura fișierelor sursă

În directorul *Controllers* sunt fișierele cu codul sursă în care se realizează schimb de *http requests* cu interfața web și cea mobile. În principal tipurile de *http request* folosite sunt cele de *GET*, *POST*, *PUT*, eventual *DELETE*.

În directorul *Models* am definit clasele necesare pentru a putea lucra cu baza de date și pentru a putea realiza schimbul de date într-un format sigur.

De asemenea, în directorul *Services*, fișierele cu codul sursă conțin mai multe funcționalități de bază cum ar fi: setarea opțiunilor metodelor și opțiunilor folosite în aplicație, precum și ordinea configurației lor, conectarea la baza de date, preluarea anumitor documente sau colecții din baza de date, etc.

3. CAPITOLUL III. Detalii de implementare

În continuare vă voi prezenta detaliile de implementare pentru fiecare modul în parte, alături, acolo unde este cazul de anumite porțiuni de cod.

3.1. Modulul API Gateway

În acest modul, așa cum am specificat și în 2.4, acesta este împărțit în 3 mari directoare: *Controllers*, *Models*, *Services*. Fiecare director conținând la rândul lui mai multe fișiere ce oferă anumite funcționalități.

Servicii

În acest director se află 3 fișiere cu extensia *.cs*. În *Startup.cs* se setează opțiunile metodelor și opțiunilor folosite în aplicație, precum și ordinea configurației lor. *Program.cs* conține metoda *main* în care se execută tot codul din cadrul acestui API.

În *Database.cs* se află toate funcțiile care au legătură cu baza de date *MongoDB*. Principalele funcționalități din cadrul acestui fișier sunt cele de a se conecta la baza de date, de a prelua anumite colecții din baza de date și lucrul efectiv asupra colecțiilor și documentelor din baza de date. Pentru a putea lucra cu baza de date *MongoDB* trebuie să importăm biblioteca *MongoDB.Driver*. Conectarea și preluarea colecțiilor se poate face așa cum se poate vedea în secvența de cod de ce urmează.

```
public DatabaseService(IConfiguration config)
{
    _config = config;
    var client = new MongoClient("mongodb+srv://admin:admin@cluster0-
vu6o6.mongodb.net/test?retryWrites=true&w=majority");
    var database = client.GetDatabase("gaudeamus");

    _users = database.GetCollection<User>("userslist");

    _products = database.GetCollection<Product>("products");

    _menus = database.GetCollection<Menu>("menus");

    _orders = database.GetCollection<Codes>("orders");

    _history = database.GetCollection<History>("history");}
```

După cum se poate observa mai sus, am realizat conectarea la baza de date creând un obiect de tipul *MongoClient* ce are ca și parametru url-ul generat de *cloud-ul MongoDB Atlas*. După se alege baza de date cu care vrem să lucrăm apelând funcția *GetDatabase(num-e_baza_de_date)*. După ce am realizat cu succes conectarea la baza de date, putem începe preluarea colecțiilor din cadrul acesteia. Acest lucru se poate face după cum se poate observa cu funcția *GetCollection<tipul_colecției>(nume_colecție)*.

De asemenea în cadrul acestui fișier am implementat mai multe funcții în care se preiau anumite informații din baza de date. Un exemplu ar fi cel din Cod 1.

```
public IMongoCollection<User> GetCollectionUser() {  
    return _users;  
}
```

Cod 1 – Funcție ce returnează toți utilizatorii din baza de date

To aici am implementat o funcție ce generează SHA-256 pentru un *string* dat ca și parametru. Această funcție am creat-o pentru a o apela asupra parolilor în situația în care un utilizator se înregistrează. Rolul acestei funcții este de a crește securitatea. Această funcție se poate observa în

```
public string ComputeSha256(string password)  
{  
    using (SHA256 sha256Hash = SHA256.Create())  
    {  
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(password));  
        StringBuilder builder = new StringBuilder();  
  
        for(int i = 0; i < bytes.Length; i++)  
        {  
            builder.Append(bytes[i].ToString("x2"));  
        }  
  
        return builder.ToString();  
    }  
}
```

Cod 2 – Funcție generare hash SHA-256

De asemenea, aici, am creat o funcție de generare JWT – *JSON web token*. Acesta este un standard pentru codificarea datelor care include informații despre rolul utilizatorului și alte date relevante și rolul acestuia este de a securiza acțiunile unui controler. Acesta oferă o metodă sigură de a transmite informații folosind un secret comun. Acest token este folosit pentru utilizatori în momentul în care și-au uitat parola. După ce aceștia introduc adresa de

email personala, va fi trimis un mail cu link-ul către pagina de resetare parola, urmat de token-ul generat de funcția menționată, care este valabilă oă, deci utilizatorul are la dispoziție doar 15 minute pentru a-și putea schimba parola. De asemenea acest token este folosit și în momentul în care un utilizator se conectează la aplicație, rolul acestuia fiind acela de a permite utilizatorul de a realiza anumite cereri *http* într-un anumit interval de timp setat la crearea acestuia. Timpul setat pentru acest caz este de 30 de zile pentru utilizatorii ce accesează aplicația mobile și 12 ore pentru cei ce accesează interfața Web. Funcția menționată anterior se poate observa în Cod 3 – Funcția de generare JWT.

```
public string GenerateJSONWebToken(string username, string type)
{
    var hours = 0;
    if(type == "web")
    {
        hours = 12;
    }
    else if (type == "password")
    {
        hours = 1;
    }
    else if (type == "mobile")
    {
        hours = 720;
    }
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config[
"Jwt:Key"]));
    var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.H
macSha256);
    var claims = new[] {
        new Claim(JwtRegisteredClaimNames.Sub, username),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
    };
    var token = new JwtSecurityToken(_config["Jwt:Issuer"],
        _config["Jwt:Issuer"],
        claims,
        expires: DateTime.Now.AddHours(hours),
        signingCredentials: credentials);
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

Cod 3 – Funcția de generare JWT

În principal în cadrul acestui fișier am implementat funcții ce lucrează asupra bazei de date cum ar fi: returnare colecții, anumite câmpuri din baza de date după un anumit criteriu, etc.

Modele

În principal, în cadrul acestui director se află fișiere ce definesc anumite datele pentru un utilizator din aplicație prin crearea de clase folosite în dezvoltarea aplicației, precum și clase ce au aceleași tipuri și aceleași variabile ca cele dintr-o colecție din baza de date. Aceste clase sunt folosite pentru a transmite date între diferite părți ale aplicației (ex: între servicii și controler), și pot fi utilizate de asemenea pentru a returna datele de răspuns *http* din metodele de acțiune ale unui controler. Un exemplu de clasă în care se definesc câmpurile dintr-o bază de date ar fi cel din Cod 4.

```
public class User
{
    [BsonId]
    [BsonElement("_id")]
    public int _id { get; set; }

    [BsonElement("email")]
    public string email { get; set; }

    [BsonElement("password")]
    public string password { get; set; }

    [BsonElement("role")]
    public string role { get; set; }

    [BsonElement("type")]
    public string type { get; set; }

    [BsonElement("token")]
    public string token { get; set; }
}
```

Cod 4 – Creare clasă cu variabile de același tip și același nume din baza de date

În care *BsonId* specifică faptul că variabila respectivă este cea prin care documentul se identifică unic, iar *BsonElement* specifică faptul că variabila ce urmează să fie declarată după face parte dintr-o colecție din cadrul bazei de date *MongoDB*.

Clasele definite în cadrul acestui modul sunt:

- *Product* – cu attributele: *int _id*, *string name*, *string category*, *double professorPrice*, *double studentPrice*, *int weight*, *string description*;
- *Orders* – cu attributele: *int _id*, *int idUser*, *IDictionary<string, int>idProductsAndAmounts*, *int code*, *DateTime date*, *double totalPrice*;

- *History* - cu attributele: *int _id, DateTime date, IDictionary<string, int> nameProductsAndAmounts, totalPrice;*
- *Meniu* – cu attributele: *_int _id, DateTime dateMenu, IDictionary<string, int> idProductsAndAmounts;*
- *User* – cu attributele: *int _id, string email, string password, string role, string type, string token;*

Controlere

Un controler definește și gestionează toate rutele / punctele finale pentru API, acesta include autentificarea și operațiile *CRUD* standard – *GET, POST, PUT, DELETE*. În cadrul fiecărei rute, controlerul apelează la serviciul utilizatorului pentru a efectua acțiunea necesară. Acțiunile controlerului sunt securizate cu JWT, așa cum am specificat mai sus, folosind atributul *[AUTHORIZE]*. Ruta unui controler se definește astfel: *[Route("nume_rută")]*. Un exemplu de o operație *CRUD* se poate observa în Cod 5.

```
[HttpPost]
public IDictionary<String, String> Post([FromBody] IDictionary<String, String>
    request)
{
    List<User> users = _userService.GetUsers();
    String type = request["type"];
    IDictionary<String, String> dict = new Dictionary<String, String>();
    if(users.Exists(x => x.email == request["email"] && x.password == _userService.ComputeSha256(request["password"]))) {
        User _user = users.Find(x => x.email == request["email"]);
        var tokenString = _userService.GenerateJSONWebToken(request["email"],
type);
        dict.Add("response", "true");
        dict.Add("id_user", _user._id.ToString());
        dict.Add("role", _user.role);
        dict.Add("type", _user.type);
        dict.Add("token", tokenString);
        return dict;
    }
    else
    {
        dict.Add("response", "false");
        return dict;
    }
}
```

Cod 5 - Post

În această porțiune de cod am făcut un post care joacă rolul funcției de *Login*, în care am primit în *body* un dicționar, în care cheia și valoarea sunt de tip *string*. În acest dicționar avem date despre adresa de email, parola, precum și tipul utilizatorului. Se verifică dacă există adresa de email în baza de date, respectiv dacă hash-ul *SHA-256* aplicat asupra parolei corespunde cu cel din baza de date. Acesta la rândul ei returnează tot un dicționar ce conține răspunsul care este *true* sau *false*, id-ul utilizatorului, rolul acestuia, tipul și un *JWT* generat de funcția menționată în Cod 3 – Funcția de generare *JWT*. În cazul în care utilizatorul nu există în baza de date, sau parola este greșită se va returna un dicționar de forma: *{“response” : “false”}*.

În continuare o să vă descriu cum ce funcții există în fiecare controller:

- *UserController.cs* - în care avem funcția *GET* ce returnează o listă cu toți utilizatorii din baza de date, dar și un *GET* ce are ca și parametru id-ul unui utilizator și returnează produsele pe baza unei recomandări, *POST* ce realizează înregistrarea unui utilizator nou în baza de date și funcția *PUT* în care se modifică anumite date despre un anumit utilizator
- *LoginController.cs* – unde avem funcția *PUT* ce realizează autentificarea unui utilizator
- *ProductController.cs* – ce conține funcția *GET* ce returnează toate produsele din baza de date, dar *GET* ce primește ca parametru o dată calendaristică și returnează produsele disponibile din acea dată, *POST* ce adaugă un nou produs în baza de date, dar și funcția *PUT* care modifică anumite informații asupra unui produs.
- *MenuController.cs* – în care avem funcția *GET(DateTime date)* ce returnează meniurile disponibile dintr-o zi dată ca și parametru, funcția *POST* în care se adăuga un nou meniu în baza de date, dar și funcția *PUT* care modifică numărul de produse disponibile din baza
- *ForgotPasswordController.cs* – în care în funcția *POST* se trimite un link de resetare parolă cu un token generat de funcția de la Cod 3 și o funcția *PUT* în care se modifică parola.
- *HistoryController.cs* – în care avem doar funcția *GET* ce returnează o listă cu toate vânzările realizate într-o anumită zi.

3.2. Modulul interfață web

Așa cum am menționat și în 1.3, Angular Framework este folosit pentru dezvoltarea aplicațiilor web. Acesta folosește pentru design *HTML*, iar pentru stilizare *CSS*. Limbajul de programare care este folosit de Angular este *TypeScript*, fiind un limbaj orientat-obiect.

Structura acestui modul este împărțită în componente. Fiecare componentă reprezintă o pagină web și în interiorul acesteia se află 4 fișiere: *.html*, *.css*, *.ts*, *.spec.ts*. Unde fișierul *.html* este folosit pentru design, *.css* este folosit pentru stilizare, iar *.ts* și *.spec.ts* sunt folosite pentru a realiza schimb de informații dintre interfața web și modului *API Gateway*, respectiv baza de date. De asemenea structura acestuia mai conține și modele și anumite servicii. În principal acest modul se ocupă cu realizarea de apeluri către API, realizarea de anumite acțiuni asupra interfeței, precum și afișarea unor anumite informații într-un mod cât mai plăcut. În continuare vă voi prezenta fiecare componentă în parte și voi descrie funcționalitățile principale, precum și anumite porțiuni de cod.

Prima pagina este cea de *login* – pagina de autentificare (vezi Fig. 3 Pagina de autentificare - web), în care se completează adresa de email și parola, se verifică dacă există în baza de date, și în funcție de tipul acestuia va fi redirectionat către o anumită pagină. În cadrul acestei componente am dat import la biblioteca *MD5* și *Router*. Prima este folosită pentru a calcula hash-ul MD5 al parolei înainte de a o trimite către API, iar a doua este folosită pentru a putea realiza redirectionarea către anumite pagini. În cadrul acestei pagini după ce numele și parola au fost preluate (via input HTML), am creat un dicționar *data = {“email” : email, “password” : password, “type” : type}*, unde *type* reprezintă tipul utilizatorului, pentru a ști intervalul de timp când este creat JWT și le-am trimis către API prin intermediul unui apel HTTP. Un exemplu de apel HTTP se poate observa în Cod 6 – apel HTTP.

```
let data = { "email" : email, "password" : password, "type" : "web"};
this.http.post<String>(this._urlLogin, data, {headers:{'Accept' : 'application
/json', 'Content-Type' : 'application/json'}})
.subscribe( data => {
  console.log(data);},
error => {
  console.log(error);})
```

Cod 6 – apel HTTP

Dacă utilizatorul a introdus datele corecte atunci acesta va fi redirecționat către o altă pagină, iar în *local storage* vor fi setate JWT și id-ul utilizatorului, în caz contrar va fi afișat un mesaj de eroare. Redirecționarea către o anumită pagină se realizează astfel:

```
this._router.navigate(['/user']);
```

A doua pagină este cea de *register* în care se vor completa adresa de email și parola de două ori și se alege tipul utilizatorului (profesor sau student), se verifică dacă cele două parole corespund, iar apoi se realizează un apel HTTP, în care vor fi trimise adresa de email, parola, precum și tipul utilizatorului și se va insera în baza de date un nou utilizator. În cazul în care ceva nu a funcționat cum trebuie, sau unele date sunt introduse greșit va fi afișat un mesaj de eroare.

Cea de-a treia pagină este *forgot-password* și este folosită în cazul în care un utilizator și-a uitat parola. Se introduce adresa de email către care va fi trimis un link de resetare parolă.

A patra pagină este *reset-password*. Principala funcționalitate a acestei pagini este cea de schimbare parolă. Aceasta poate fi accesată într-un interval de o oră după ce a fost trimis link-ul de resetare parolă menționat anterior

Cea de-a cincea pagina este *user* ce conține informații despre meniul din ziua curentă. Utilizatorul poate alege produsele dorite, iar apoi prin acționarea unui buton va fi generat automat un cod de 5 cifre, care va fi utilizat mai târziu de către casierite. Afișarea meniului se realizează printr-un apel HTTP GET ce returnează o listă de produse, și unul ce returnează meniul din ziua curentă și se formează două dicționare *buyProductsTotal* și *buyProductsNumber* care sunt sub forma *{nume_produs : cantitate}* ce rețin numărul total de produse disponibile și numărul de produse alese. În cadrul acestui meniu vor fi afișate doar produsele disponibile. Acest lucru le-am realizat cu ajutorul unei directive structurale **ngIf* care este folosită în cadrul fișierului *html* și este folosită pentru a evalua o anumită expresie. Când expresia este adevărată, Angular redă șablonul furnizat într-o clauză *then*, iar când este fals într-o clauză *else*. Un exemplu de directivă structurală **ngIf* se poate observa în exemplul următor:

```
<div *ngIf = "typeUser == 'student'; then thenBlock; else elseBlock"></div>
<ng-template #thenBlock>
  <li><p>{{product.studentPrice}} lei</p></li>
</ng-template>
```



```
<ng-template #elseBlock>
  <li><p>{{product.professorPrice}} lei</p></li>
</ng-template>
```

Cod 7 – directiva *ngIf

În Cod 7 – directiva *ngIf, se verifică tipul utilizatorului. Dacă acesta este student, atunci va fi afișat prețul studentului, în caz contrar, va fi afișat prețul profesorului.

Cea de a șasea pagină *admin-create-menu* – principala funcționalitate din cadrul acestei pagini este cea de completare a unor câmpuri (input *html*), de creare a meniului dintr-o anumită zi prin selectarea produselor din lista de produse disponibile, precum și introducerea cantității pentru fiecare produs în parte. Produsele împreună cu cantitatea fiecăruia le-am pus într-un dicționar de forma *{nume_produs : cantitate}*. Cantitatea am setat-o cu ajutorul unui *event listener* (vezi Cod 8 – Schimbare cantitate produs cu *event listener*), care modifică valoarea pentru fiecare produs oricând aceasta este modificată. În cazul apariției unei erori, va fi afișat un mesaj de eroare.

```
onChangeInput(event, product){
  this.myProducts[product.toString()] = +event.target.value;
}
```

Cod 8 – Schimbare cantitate produs cu *event listener*

Cea de a opta pagină este *admin-add-product*. În cadrul acestei pagini vor fi completate anumite câmpuri, ce reprezintă informațiile despre un anumit produs. Aceste informații vor fi puse într-un dicționar sub forma *{nume : valoare}* și vor fi trimise către API prin intermediul unui apel HTTP în care produsul va fi adăugat în baza de date, în caz contrar va fi afișat un mesaj de eroare.

Cea de a noua componentă este *admin-add-admin*, unde se completează un câmp cu adresa de email a unui utilizator deja existent în baza de date. Rolul acesteia este cel de a schimba tipul utilizatorului, cel de a adăuga personal nou în cadrul cantinei.

Cea de a zecea pagină este *update-product* și este folosită pentru a schimba anumite informații despre un produs existent în baza de date.

Cea de-a unsprezecea pagină este *admin*. Această pagină este pentru personalul din cadrul cantinei. Conține informații despre meniul din ziua curentă și oferă funcționalități precum: alegerea produselor, introducerea unui cod de 5 cifre și generarea automată a unor produse alese de către un anumit utilizator, precum și funcția de cumpărare, unde va fi generat un bon și va fi printat.

O alta componentă este *product-details* în care se caută anumite produse din baza de date, în cazul în care nu se mai știu anumite informații despre un produs.

Ultima pagină este *history*. Aici se selectează o anumită dată calendaristică, iar după acționarea unui buton va fi afișat un istoric cu toate comenzile din ziua respectivă

3.3. Modulul interfață mobile

Flutter framework este considerat unul dintre cele mai documentate medii de lucru și este folosit în dezvoltarea aplicațiilor mobile pentru două dintre cele mai cunoscute sisteme de operare folosind o singură bază de cod. Limbajul folosit de această tehnologie este Dart, fiind un limbaj orientat-obiect.

Structura acestui modul este formată din pagini. În principal, acest modul realizează apeluri HTTP către API prin afișarea unor anumite informații din baza de date, precum și modificarea anumitor documente din baza de date.

Principalele pagini din cadrul acestui modul sunt: *main.dart*, *register.dart*, *forgotpassword.dart*, *user.dart*. În fiecare dintre acestea sunt implementate metode ce se ocupă cu apeluri către API.

```
Login (String email, String password) async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  password = generateMd5(password);

  HttpClient client = new HttpClient();
  Map data = {
    'email' : email,
    'password' : password,
    'type' : "web"
  };

  HttpClientRequest request = await client.postUrl(Uri.parse(urlLogin));
  request.headers.set('content-type', 'application/json');
  request.add(utf8.encode(json.encode(data)));

  HttpClientResponse response = await request.close();

  return response;
}
```

Cod 9 – Exemplu apel Http Dart

După cum se poate observa în

Cod 9 – Exemplu apel Http Dart, metodele de apel către API sunt asincrone. Se setează adresa *url*, apoi se setează *header*-ul, apoi se setează *body*-ul, acolo unde este cazul, iar la final se obține răspunsul.

De asemenea, pentru setarea și obținerea de valori din memoria locală a dispozitivului am folosit pachetul *shared_preferences*, unde informațiile sunt stocate sub forma *{cheie : valoare}*. Un exemplu de obținere a unei valori din memoria locală a dispozitivului se poate observa în Cod 10 - Shared preferences Dart.

```
SharedPreferences prefs = await SharedPreferences.getInstance();
setState(() {
  this.token = prefs.get("token");
  this.type = prefs.get("type");
});
```

Cod 10 - Shared preferences Dart

În principal, paginile din cadrul acestui modul au aceleași funcționalități cu cele din cadrul Modulul interfață web, diferența fiind tehnologia folosită.

Concluzii

Consider că aplicația este utilă și oferă soluții pentru reducerea timpului de așteptare pentru realizarea comenzilor din cadrul unei cantine. Soluția oferită dispune de funcționalități precum generarea unui cod de 5 cifre în cadrul unui utilizator care va fi introdus apoi de către angajații din cadrul personalului și generarea unui bon automat pe baza acestuia, recomandare de produse, creare de meniu, adăugarea de produse, vizualizarea istoricului, precum și alte funcționalități ce fac munca personalului mai ușoară și reduce timpul de așteptare al clienților și utilizatorilor din cadrul acesteia.

Datorită tehnologiilor folosite, interfața aplicației poate fi dezvoltată pe viitor pe mai multe sisteme de operare. De asemenea aceasta a fost dezvoltată astfel încât să poată să fie dezvoltată ușor, prin adăugarea de noi funcționalități și de noi module, cum ar fi introducerea unei noi opțiuni de traducere automată a informațiilor pentru utilizatorii de alte naționalități sau un sistem de *speech-to-text* și *text-to-speech* pentru ajuta utilizatorul să afle anumite informații în timpul în care acesta nu are acces constant la *device* (ex: la volan).

Bibliografie

Kouwer, B. J. (1949). *Colors and Their Character: A Psychological Study*.