

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Orderino – aplicație web pentru managementul
restaurantelor**

propusă de

Florin Cojocariu

Sesiunea: *iulie, 2018*

Coordonator științific

Drd. Colab. Florin Olariu

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

Orderino – aplicație web pentru managementul restaurantelor

Florin Cojocariu

Sesiunea: *iulie, 2018*

Coordonator științific

Drd. Colab. Florin Olariu

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele

Data

Semnătura

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul(a).....

domiciliul în.....

născut(ă) la data de, identificat prin CNP,

absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de

..... specializarea,

promoția....., 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:

_____elaborată sub îndrumarea dl. / d-na

_____, pe care urmează să o susțină 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 diploma 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 azi,

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Orderino – aplicație web pentru managementul restaurantelor*”, 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, *data*

Absolvent *Florin Cojocariu*

(semnătura în original)

Cuprins

Introducere.....	5
Contribuții.....	7
Capitolul 1 : Scopuri și cerințe ale aplicației.....	8
1.1 Utilizatorul de tip administrator	8
1.2 Utilizatorul obișnuit.....	9
Capitolul 2. Analiză, proiectare și detalii de implementare	10
2.1 Analiză și proiectare	10
2.1.1 Descrierea aplicației	10
2.1.2 Specificații tehnice	10
2.1.3 Scenarii de utilizare.....	15
2.2 Detalii de implementare.....	23
Concluzii.....	29
Bibliografie.....	30

Introducere

În ultima perioadă de timp, s-a putut observa o dezvoltare foarte rapidă a Web-ului, considerat unul dintre cele mai de succes servicii ale Internet-ului. Unul dintre motivele dezvoltării într-un ritm atât de alert este reprezentat de evoluția aplicațiilor web. Dacă la început scopul aplicațiilor web era acela de a facilita transferul de documente, în ziua de astăzi au ajuns să fie unul din pilonii principali ai economiei mondiale. Această dezvoltare a avut un impact asupra modului în care funcționează principalele domenii de activitate. Cele mai semnificative schimbări observate au fost în domenii precum marketing, vânzări și comunicare.

În cadrul lucrării de licență am decis să folosesc această metodă de reprezentare a unei soluții la o problemă dată.

Motivația alegerii acestei teme o constituie observarea anumitor probleme care au apărut în ultima vreme în domeniul restaurantelor, datorate de creșterea numărului de clienți raportat la calitatea serviciilor prestate de restaurante și de angajații acestora. Pornind de la această idee, am vrut să descopăr care sunt principalele cauze pentru care aceste probleme există și în același timp să dezvolt o rezolvare a acestora. Astfel, prin intermediul aplicației *Orderino* s-a dorit oferirea unei modalități prin intermediul căreia să se răspundă nevoilor pe care le au utilizatorii.

Consider că acest domeniu necesită o revoluționare deoarece din ce în ce mai multe persoane apelează la aceste servicii și pe aceste considerente ar trebui să aibă o experiență cât mai plăcută.

Gradul de noutate adus de aplicația *Orderino* vine din modul în care această aplicație a fost concepută, ideea de bază fiind oferirea unui mod plăcut și atractiv utilizatorului dornic să consume produse în cadrul unui restaurant. Consider că tema propusă este una nouă în acest domeniu deoarece nu există aplicații similare, singurele aplicații fiind fie cele de rezervare de mese pentru restaurante, fie cele de livrări la domiciliu.

Obiectivul general al aplicației *Orderino* este acela de a reduce costurile deținătorilor de restaurante prin reducerea personalului necesar preluării și prelucrării comenzilor clienților, dar și de a îmbunătăți experiența solicitanților, prin prezentarea într-o formă atractivă a meniului și oferirea posibilității de a comanda direct produsele dorite, fără a fi nevoie să aștepte personalul restaurantului pentru a-i aduce meniul, a prelua comanda dar și a-i aduce produsele dorite.

O descriere sumara a aplicației poate fi exprimata prin intermediul următoarelor cuvinte: restaurant, scop profesional, ușurință în utilizare, flexibilitate dar nu în ultimul rând extensibilitate. Prin combinarea celor mai noi tehnologii și cadre de lucru precum: *Laravel*, *Angular* s-a adus un plus aplicației în ceea ce privește partea de modularizare, extensibilitate și flexibilitate. De asemenea prin recurgerea la apelarea unor REST Api-uri se asigura o mai bună calitate a informațiilor necesare. În concluzie, *Orderino* este aplicația care se adresează persoanelor dornice să aibă o experiență cât mai plăcută pe toată perioada petrecută în cadrul unui restaurant.

Capitolele care alcătuiesc structura acestei lucrări sunt constituite într-o relație de interdependență. Prin intermediul acestora este relatată povestea realizării acestei aplicații din momentul alegerii temei, până în momentul implementării acestei aplicații, prezentând pas cu pas etapele urmate în procesul de dezvoltare.

Primul capitol descrie modul în care aplicația va funcționa. Ca urmare, se prezintă scopurile și structura aplicației. Este determinat scopul principal ca fiind crearea unei platforme, prin intermediul căreia clienții să se bucure de cele mai bune servicii posibile, facilitând astfel consumul de produse dar și integritatea acestora. De asemenea se descrie necesitatea adăugării unui nou tip de utilizator, utilizatorul de tip administrator. Administratorul are o privire de ansamblu asupra modului în care aplicația funcționează, având acces la datele cu caracter public ale utilizatorului, posedând abilitatea de a adăuga noi utilizatori de tip administrator. Tot în acest capitol este prezentată funcționalitatea care se oferă utilizatorului obișnuit.

Cel de al doilea capitol este dedicat descrierii aplicației împreună cu modul de funcționare al acesteia. Prima parte s-a axat pe tehnologiile utilizate la nivelul clientului, împreună cu scenariile de utilizare, urmând ca ulterior să explice funcționalitățile ce sunt în spatele cazurilor de utilizare prezentate.

Contribuții

Proiectul îmbină ideile personale cu cele ale d-lui profesor coordonator, pentru a realiza o structură unitară, cu funcționalități precise, ușor configurabile și care să afișeze rezultatele într-o manieră cât mai prietenoasă.

Interesul utilizatorului tipic este acela de a interacționa cu o interfață interactivă și intuitivă care să îi ofere posibilitatea consumului de bunuri în locația în care se află dar și de a afla mai multe informații referitoare la produsele disponibile, care să influențeze deciziile pe care le ia în privința alegerilor pe care le va face.

Modalitatea de dezvoltare a aplicației poate fi descrisă în două cuvinte: modularizare și extensibilitate. Pentru a putea realiza acest lucru, s-a apelat la o serie de resurse menite să faciliteze maniera de dezvoltare a aplicației, stabilindu-se o structură clară a aplicației, prezentată pe parcursul lucrării.

De asemenea pentru realizarea lucrării am reușit să îmbin atât cunoștințele dobândite în timpul orelor de la facultate dar și în urma studiului individual. Principalele materii pe baza cărora am reușit să dezvolt această lucrare au fost: Tehnologii Web, unde au fost puse bazele WEB-ului în general, în special limbajul de programare *PHP*, pe lângă care am ales ca și cadru de dezvoltare *Laravel*, dar și ale JavaScript-ului, HTML-ului și ale CSS-ului, limbaje absolut necesare oricărui site WEB. Pot spune de asemenea, faptul că aptitudinile dobândite la disciplina OOP m-au ajutat să scriu un cod reutilizabil dar și flexibil la eventualele schimbări care vor apărea pe viitor.

Capitolul 1 : Scopuri și cerințe ale aplicației

Reamintim faptul că, *Orderino* este o aplicație web, care vine ca un răspuns unei nevoi de comercializare de produse și de găsim a informațiilor dorite într-un mod compact și rapid. Aflându-ne într-o perioadă în care Web-ul este o continuă dezvoltare, *Orderino* este tipul de aplicație care trebuie dezvoltată astfel încât să-și păstreze scopul principal, acela de a oferi o unealtă prin care utilizatorul să fie în permanență prima prioritate. Astfel, odată cu evoluarea tehnologiei, va fi relativ ușor de optimizat și de adăugat noi module de interes în cadrul aplicației, datorită modului în care aceasta este realizată.

Până în momentul de față, s-a evidențiat modul în care această aplicație a fost concepută și care este scopul final. Reluând cele menționate în paragraful anterior putem spune că scopul principal al acestei aplicații este acela de a oferi un sistem prin intermediul căruia să-i fie facilitat accesul la anumite produse puse în vânzare în cadrul unui restaurant. Utilizatorul are oportunitatea de a ocupa un loc în cadrul unui restaurant, de a comanda produse dar și de a plăti aceste produse, toate acestea efectuându-se fără intervenția vreunui factor extern.

Luând în considerare toate informațiile până în momentul actual, se pot contura o serie de funcționalități pe care aplicația ar trebui să le ofere. Astfel putem defini două tipuri de utilizatori: utilizatorul de tip administrator și utilizatorul obișnuit interesat de funcționalitatea propriu-zisă.

1.1 Utilizatorul de tip administrator

Necesitatea acestui tip de utilizator a reieșit din dorința de a oferi o experiență utilizatorului de zi cu zi al aplicației. Prin crearea acestui tip de utilizator, intră în joc o entitate prin intermediul căreia se poate păstra un control al aplicației. Un administrator are acces la o serie de funcționalități precum gestionarea resurselor pentru o bună funcționare a aplicației precum adăugarea, editarea, ștergerea de produse din cadrul unei anumite categorii, dar și vizualizarea tuturor comenzilor care au fost procesate în sistem.

1.2 Utilizatorul obișnuit

Am definit ca utilizator obișnuit tipul de persoană al cărei scop este utilizarea aplicației pentru funcționalitatea care îi este oferită. Astfel, pentru acest tip de utilizator s-a gândit un scenariu cât mai simplificat. La nivel de funcționalitate, modulele ce se doresc a fi implementate sunt în număr de trei. Acestea pot fi împărțite în: autentificare, necesară pentru a putea ști cu exactitate identitatea utilizatorului prezent în acel moment în restaurant, comandare de produse, folositor pentru identificarea cu precizie a produselor dorite de fiecare client în parte, plata produselor, necesară pentru o bună gestiune a localului.

Astfel un utilizator se va prezenta în cadrul restaurantului și se va așeza la o masă. Când acesta va dori să comande anumite produse va trebui să scaneze cu ajutorul telefonului personal un cod QR^1 care îl va introduce în aplicație.



Figura 1 – Exemplu de cod QR

Odată ajuns aici pentru a se putea autentifica va fi nevoie ca utilizatorul să își introducă adresa personală de e-mail unde va primi un cod de verificare. După ce un utilizator și-a confirmat prezența aplicația îl va așeza pe un scaun în cadrul mesei la care se află. De aici clientul va putea în continuare să-și comande produsele pe care dorește să le consume urmând ca apoi să efectueze plata acestora.

¹ Codurile QR stochează date folosind șabloane de puncte negre și spații albe, aranjate într-o grilă pătrată. Aceste șabloane pot fi scanate și traduse în informații inteligibile nouă, oamenilor, cu ajutorul unui dispozitiv de imagine precum o cameră foto sau un scanner, deși cea mai comună metodă de a scana coduri QR în prezent este să folosești camera smartphone-ului și o aplicație specializată pentru scanarea de coduri QR.

Capitolul 2. Analiză, proiectare și detalii de implementare

În capitolul anterior, s-a oferit o privire de ansamblu asupra scopului aplicației și a principalelor funcționalități dorite de utilizatori, dorindu-se astfel eficientizarea procesului de comandare de produse. Punctul de concentrare în cadrul acestui capitol va fi asupra modului în care aplicația este proiectată astfel încât să ofere funcționalitatea dorită împreună cu detaliile corespunzătoare de implementare. Începând cu motivarea alegerii anumitor tehnologii, precum și detalii specifice de implementare împreună cu eventuale dificultăți în realizarea anumitor module în ceea ce privește partea de funcționalitate.

2.1 Analiză și proiectare

În cadrul acestui subcapitol, va fi pus accentul pe modul în care a fost proiectată aplicația. Totodată va fi descris și modul în care utilizatorul se poate folosi de aceasta, cu ajutorul scenariilor de utilizare.

2.1.1 Descrierea aplicației

Orderino este o aplicație al cărei scop este acela de a oferi o experiență cât mai plăcută utilizatorilor, de a prezenta într-o manieră atractivă meniul unui restaurant și de a reduce costurile administratorilor de restaurante acest lucru implicând totodată și creșterea gradului de satisfacție al clienților. Astfel s-a luat decizia de a alege un aspect plăcut din punct de vedere vizual, dar și intuitiv, prin intermediul căruia accesarea funcționalității este simplificată și modularizată corespunzător.

2.1.2 Specificații tehnice

Componentele care alcătuiesc structura principală a aplicației, cu privire la transferul informațiilor între client și server sunt definite astfel:

- **Componenta server:** această funcționalitate este realizată cu ajutorul cadrului de dezvoltare *Laravel*, pentru PHP
- **Componenta bază de date:** această funcționalitate este realizată folosind sistemul de gestiune al bazelor de date MySQL

- **Componenta client:** această funcționalitate este realizată cu ajutorul platformei de programare Angular

În cadrul componentei server am ales să folosesc cadrul de dezvoltare *Laravel*, pentru PHP. Am ales să folosesc această platformă de programare deoarece *Laravel* vine cu o serie de specificații cu ajutorul cărora munca programatorilor este semnificativ diminuată și ușurată. Una dintre aceste specificații este modul în care se realizează rutarea, astfel realizându-se cu ușurință conceptul de „pretty URL”.



Figura 2 – Logo Laravel

Un alt factor important care m-a îndrumat să aleg *Laravel* a fost acela că este bazat pe paradigma de programare orientată pe obiecte. Datorită acestui lucru proiectul a fost împărțit pe module specifice fiecărei funcționalități : *Modele, Controlere, Migrații* dar și *Baze*(eng. *Seeds*).

Cu ajutorul *migrațiilor* se oferă o modalitate la îndemâna oricui de a-și contura structura unui tabel împreună cu restricțiile necesare stocării datelor, dar și asigurarea integrității acestora . Deși acest concept nu este nou, modul în care acest mediu de dezvoltare este proiectat îl face să fie foarte ușor de utilizat. Tot cu ajutorul acestui concept se poate face foarte ușor transferul oricărei baze de date către alt mediu de programare.

Un alt plus pe care îl oferă acest cadru de dezvoltare este acela al prepopulării unei baze de date cu informații necesare funcționării în condiții optime a unei aplicații.

Un alt aspect foarte important care a dus la alegerea acestui cadru de programare a fost securitatea. Utilizând *Laravel* securitatea este la un nivel foarte ridicat într-un timp foarte scurt,

autentificarea pe partea de server făcându-se pe baza unui pachet oficial numit *Passport*, pachet care permite realizarea unui modul ce are la baza tehnologia OAuth2.0, un standard de autentificare utilizat de orice programator care își dorește un nivel de securitate ridicat. *Laravel* permite programatorului să impună multe restricții menite să diminueze breșele de securitate arhicunoscute precum cele de tip *brute-force*.

Pe partea de baze de date am ales să folosesc MySQL, principalul motiv fiind acela că este un sistem de gestiune al bazelor de date care poate rula indiferent de sistemul de operare. Inițial am dezvoltat aplicația pe sistemul de operare *Windows*, urmând ca apoi aceasta să fie dezvoltată și menținută pe sistemul de operare *Ubuntu*.



Figura 3 – Logo MySQL

Un alt factor a fost acela că acest sistem de gestiune al bazelor de date este foarte eficient când vine vorba de un volum foarte mare de date. Este de știut faptul că în cadrul unui restaurant numărul de produse poate fi unul foarte mare și să nu uităm de numărul de comenzi care pot exista.

În *Figura*, se poate observa reprezentarea bazei de date prezentă în momentul actual în cadrul aplicației. Baza de date este formată din unsprezece tabele cu funcționalități bine definite, menite să stocheze datele necesare bunei funcționări a aplicației. Dintre cele mai importante tabele amintim: *Product*, *Order*, *User*.

Tabela *Product*, reprezintă nucleul aplicației, deoarece fără această tabelă nu s-ar putea furniza informații despre meniul restaurantului dar nici nu s-ar putea efectua comenzi în cadrul acestuia.

Tabela *User*, stochează datele necesare autentificării utilizatorilor, fie ca este vorba de clienții restaurantului fie de administratorii acestuia.

În cadrul tabelii *Order*, utilizatorul are posibilitatea de a plasa comenzi pentru anumite produse.

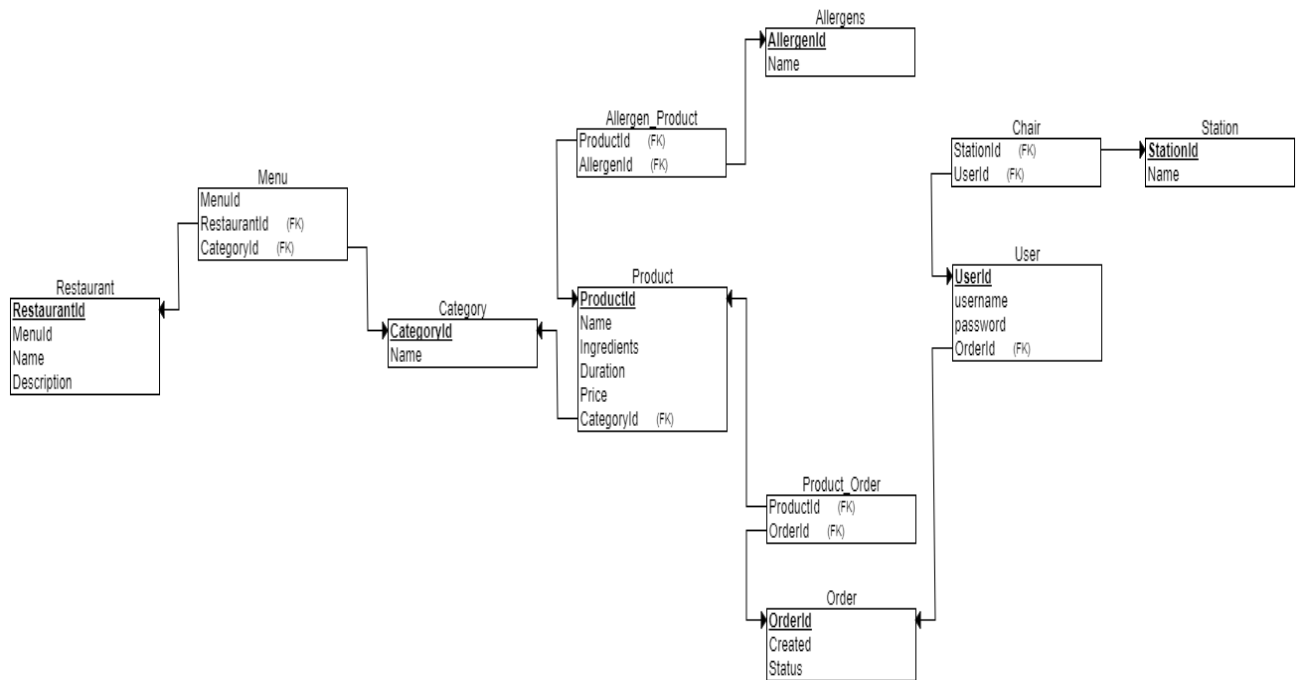


Figura 4 – Reprezentarea bazei de date

La nivel de client am decis să folosesc platforma de programare *Angular*, cu ajutorul căruia am reușit să creez o experiență inedită utilizatorului.

Principalul motiv pentru alegerea folosirii *Angular*, a fost acela că am dorit să dezvolt o aplicație cât mai performantă, dar în același timp cât mai atractivă pentru vizitatori, cu un design impecabil prin prisma animațiilor complexe.

Timpul de încărcare al paginilor a fost un alt criteriu important. Dacă facem un exercițiu de imaginație vom observa că aplicația trebuie să fie eficientă din acest punct de vedere, fiind

folosită în majoritatea timpului de pe dispozitive mobile, tablete telefoane mobile, iar aceste suporturi folosesc un timp de încărcare al paginilor superior unei conexiuni clasice. În cazul nostru, când librăriile JavaScript sunt aduse de pe server, ele sunt minificate și comprimate. Pe lângă dimensiunea cadrului de dezvoltare, trebuie însă luate în calcul și dimensiunile altor librării necesare funcționării.



Figura 5 – Logo Angular

Un ultim aspect dar la fel de important a fost acela că am dorit ca aplicația dezvoltată să fie destinată cu precădere dispozitivelor mobile. Acesta este motivul pentru care am dorit să apelez la o structură de tip „Aplicație pe o singură pagină”(eng. *Single Page Application*), pentru ca utilizatorul să nu fie nevoit să reîmprospăteze mereu pagina pentru a vedea ultimele noutăți, conținutul paginii se va schimba automat odată cu accesarea unei referințe.

Totodată împreună cu *Angular* am apelat și la HTML, CSS, Bootstrap. HTML reprezintă un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate aplicației client. Acesta împreună cu CSS-ul, cu ajutorul căruia sunt adăugate foi de stiluri paginilor web, sunt două componente absolut esențiale în dezvoltarea aplicațiilor web.

Bootstrap este cel mai popular cadru de dezvoltare la nivel global, folosit în dezvoltarea aplicațiilor web la nivel de client pentru realizarea de pagini web adaptive. Acesta permite dispunerea conținutului unei pagini în funcție de rezoluția ecranului pe care este vizualizat, acest lucru fiind prioritar în cadrul aplicației dezvoltate.

2.1.3 Scenarii de utilizare

1. Utilizatorul dorește sa se așeze la masă

Pași: Utilizatorul scanează codul QR prezent pe scaunul de la masa unde este așezat. Odată întreprinsă această acțiune el va fi introdus în aplicație, unde primul pas pe care va trebui să îl facă este acela de a-și completa adresa de email personală. Acest lucru este exemplificat în figura următoare.

The screenshot displays the 'ORDERINO' app interface. At the top, there is a dark header bar with 'ORDERINO' on the left and 'MENU' on the right. Below the header, the text 'Email address' is shown in a light gray font. A yellow rectangular input field contains the email address 'cojocariu.cojocariug@gmail.com'. Below the input field, a small gray text line reads 'We'll never share your email with anyone else.' An orange rectangular button with the text 'Submit' is positioned below the input field. At the bottom of the screen, a light green rounded rectangular box contains the message: 'You entered an valid email, please go check your email in order to seat on a table!'.

Figura 6– Autentificarea unui utilizator

Pe adresa personală de email clientul va primi un mesaj care dorește sa verifice identitatea utilizatorului. De aici el va putea fie sa apese pe butonul *EAT*, fie sa

acceseze adresa prezentă în partea de jos a mail-ului pentru a-i fi verificată identitatea.

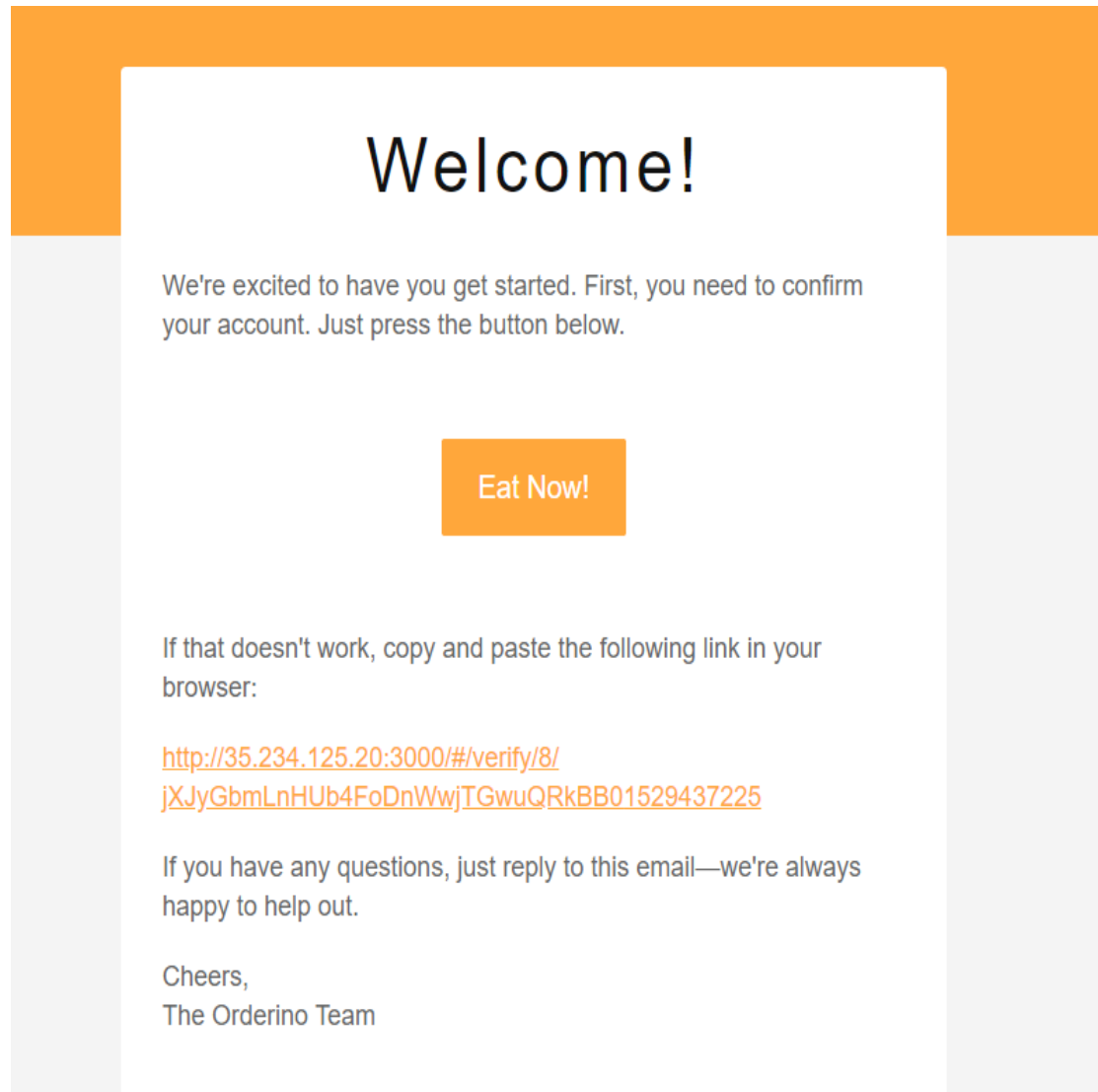


Figura 6 – Verificarea utilizatorului prin intermediul adresei personale de e-mail

După ce pasul anterior a fost efectuat cu succes, utilizatorul va fi readus în aplicație unde îi este afișat un mesaj de succes, în caz ca utilizatorul va încerca sa acceseze în mod repetat adresa prezentă în e-mail, acest lucru nu va fi posibil, pe ecran se va arăta un mesaj de eroare. Când toți pașii de mai sus au fost îndepliniți utilizatorul va fi așezat la masă.

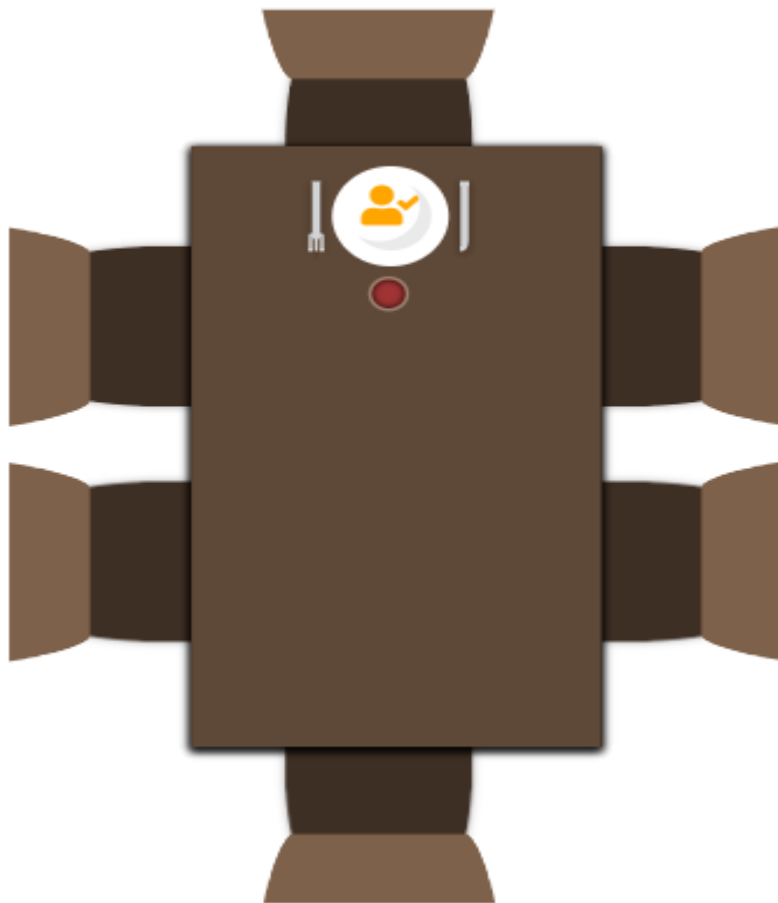


Figura 7 – Un utilizator așezat la masă

2. Utilizatorul dorește sa comande produse

Pași: Utilizatorul trebuie să fie autentificat pentru a putea comanda produse. Odată ce clientul este autentificat acesta poate accesa meniul, iar pe ecran se vor afișa toate categoriile prezente în momentul actual, împreună cu produsele pe care le conțin.

Delicious Menu

You will find a cozy restaurant with classic interior, dominant with calm and warm colors. The restaurant could serve up to 24 persons at the same time.

PIZZA
SALADS
PASTAS
FISH
SOUPS
BREAKFAST
BEVERAGE



Pizza Delicio

Ingredients: Bacon, Chicken breast, Tomatoes, Olives, Bread

\$ 4.99



Figura 8 – Categoriile meniului împreună cu produsele specifice

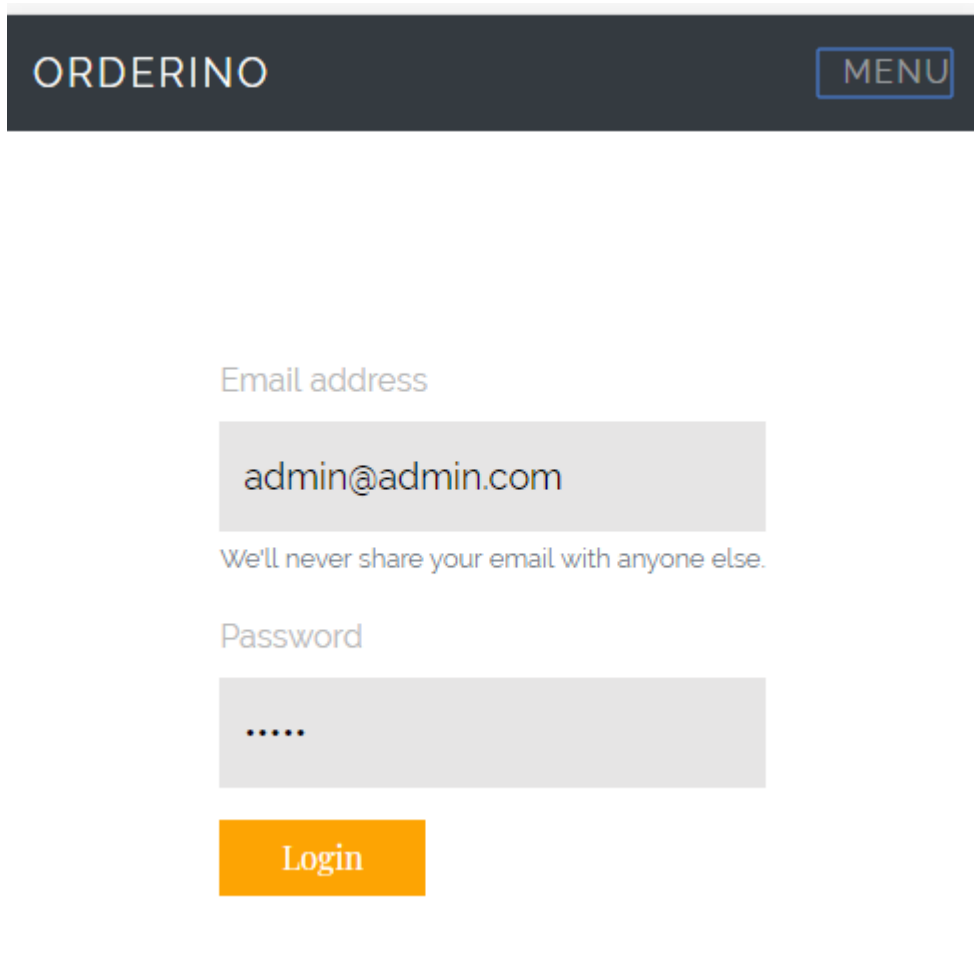
În cadrul acestei pagini utilizatorul va putea comanda produse prin apăsarea butonului *Comandă*(eng. *Order*), acesta nefiind restricționat la numărul dar și tipul produselor pe care le dorește.

3. Utilizatorul dorește sa plătească fie produsele comandate

Pași: După ce toate produsele au fost livrate și consumate de către clienți, aceștia vor avea posibilitatea de a-și plăti toate comenzile. De aceea prin apăsarea butonului *Plătește*(eng. *Pay*) , utilizatorul va avea două posibilități fie să plătească doar pentru produsele comandate de el, fie să plătească pentru produsele comandate de întreaga masă. După ce a luat această decizie din contul „virtual” se va extrage suma reprezentată de prețul tuturor produselor comandate, urmând ca apoi utilizatorul să fie automat deautentificat, acest lucru implicând si eliberarea locului la masă.

4. Administratorul dorește să se autentifice

Pași: Administratorul va trebui să acceseze pagina destinată autentificării. Odată ce a ajuns pe această pagină va trebui sa completeze adresa de email dar si parola pe care o deține.



The image shows a web interface for an administrator login. At the top, there is a dark header bar with the text "ORDERINO" on the left and a "MENU" button on the right. Below the header, the page has a light gray background. The login form consists of two main sections: "Email address" and "Password". The "Email address" section has a text input field containing "admin@admin.com" and a small text line below it that says "We'll never share your email with anyone else." The "Password" section has a password input field with five dots indicating the password length. Below these fields is an orange "Login" button.

Figura 9 – Autentificarea unui administrator

5. Administratorul dorește să adauge / editeze / șteargă produse

Pași: Pentru ca un administrator să poată adăuga / editeze sau să șteargă produsele meniului acesta trebuie ca în prealabil acesta să fie autentificat. După ce autentificarea a avut loc cu succes administratorul va fi redirecționat către pagina unde poate vedea toate categoriile prezente în meniu. Dacă un administrator dorește să adauge un produs nou va fi necesar să selecteze categoria în cadrul căreia dorește să facă schimbarea urmând ca apoi să apese butonul „+”.

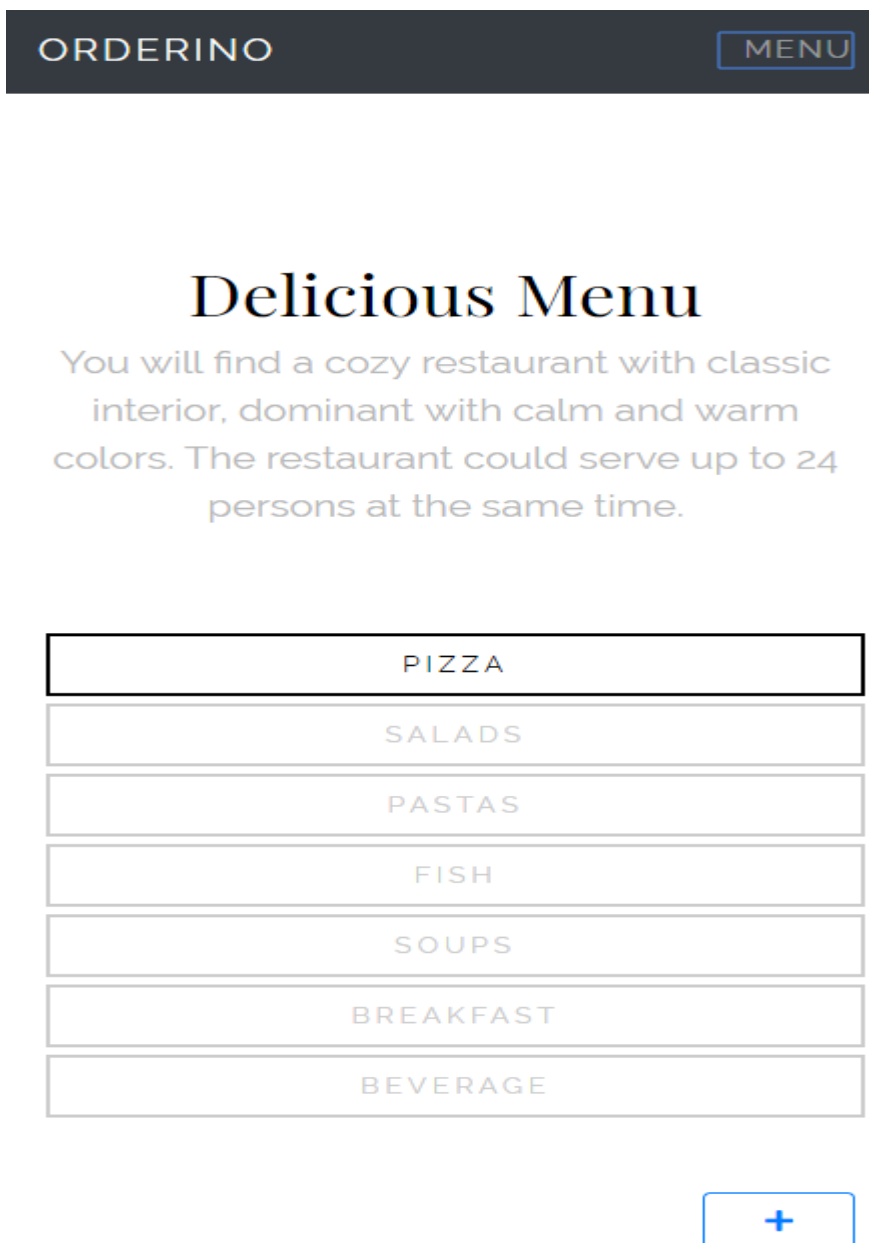


Figura 10 – Exemplu buton adăugare produs

Pe ecran se va afișa un formular pe care administratorul trebuie să îl completeze pentru ca acțiunea să fie finalizată.


Name	Sparkling Water
Price	1.29
Allergens	Select allergens
Ingredients	Water
<div>Choose file... Browse</div>	
	
<div>Submit</div>	

Figura 11 – Exemplu formular adăugare produs

Pentru operația de editare procedeul este similar, singura diferență fiind faptul că

administratorul va fi nevoit să apese butonul „*Editează*”. Pentru comanda de ștergere administratorul va trebui să acceseze butonul „*Ștergere*”.

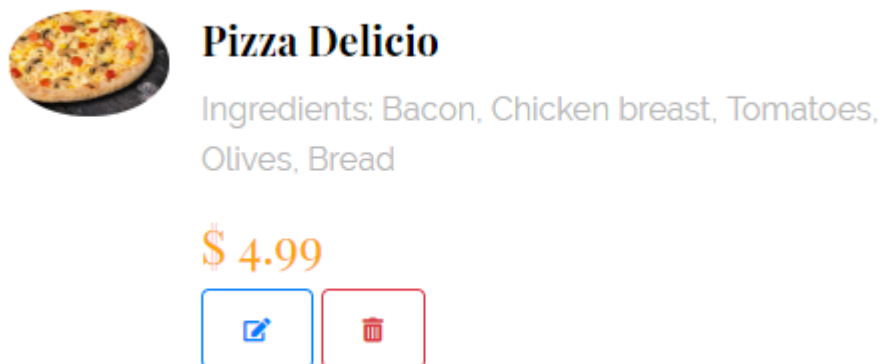


Figura 12 – Exemplificarea butoanelor de editare / ștergere

6. Administratorul dorește să vizualizeze toate comenzile din sistem.

Pași: Administratorul va trebui să acceseze din meniul contextual opțiunea „*Vizualizează comenzi*”(eng. *View Orders*). Administratorul are posibilitatea de a vizualiza comenzile pentru ziua curentă indiferent dacă acestea au fost plătite sau încă sunt în curs de procesare.

2.2 Detalii de implementare

În cadrul acestui subcapitol, se evidențiază implementarea principalelor module ale aplicației, accentul fiind pus asupra modului în care s-a realizat funcționalitatea descrisă în subcapitolele anterioare.

În cadrul aplicației am folosit o arhitectură dispusă pe mai multe straturi(module). Acest lucru a determinat o modularizare și o scalabilitate foarte mare, astfel aplicația permite adăugarea de noi module fără a fi nevoie de rescrierea completă a celor existente. Totodată am obținut o aplicație flexibilă, un cod reutilizabil dar și ușor inteligibil. De aceea, am hotărât să apelez la paradigma REST, construind astfel un API REST bazat pe Laravel, ce comunică cu componenta client prin intermediul cererilor de tip HTTP.

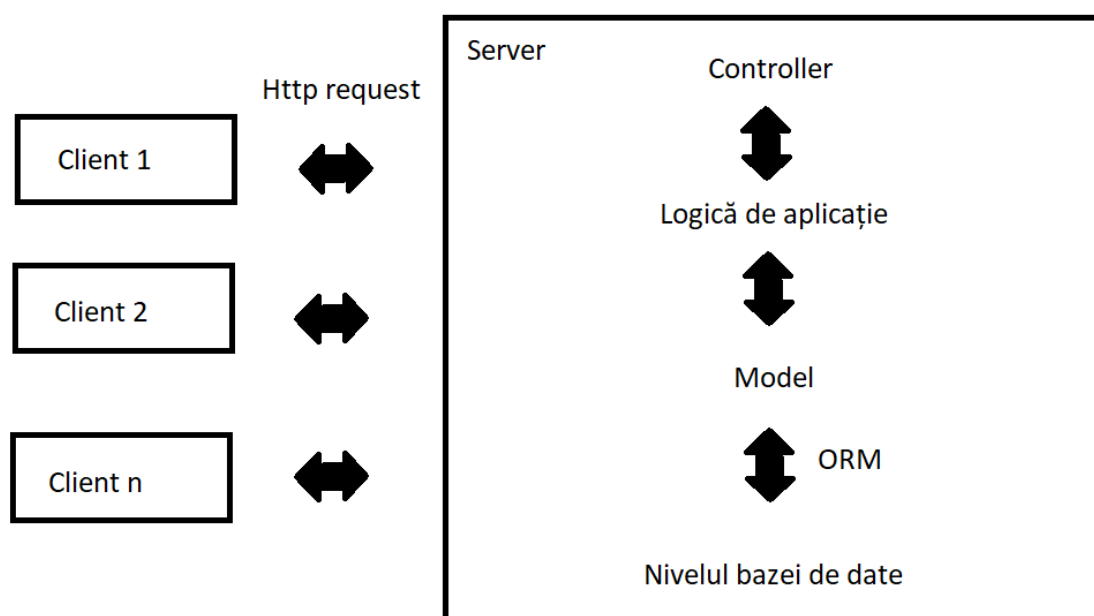


Figura 13 – Structura modulelor aplicației.

În cadrul proiectului am folosit tehnica „*Dependency Injection*” care m-a ajutat la obținerea unui cuplaj redus între obiecte. Cu alte cuvinte, dacă o clasă necesită un alt obiect sau o listă de obiecte, acestea nu sunt instanțiate în mod direct, în schimb sunt “injectate” cel mai adesea prin intermediul constructorului. În aplicația dezvoltată, m-am folosit de această tehnică la scrierea codului pentru controlere, dar și pentru implementarea fiecărui *repository*, injectând

dependențele prin intermediul constructorului.

Cum a fost menționat și anterior modul de scriere al codului din cadrul aplicației este bazat pe tiparul „*Repository Pattern*”. Acesta este folosit pentru crearea unui nivel de abstractizare între datele prezente în baza de date și logica aplicației. Acest lucru are un beneficiu enorm întrucât orice schimbare, apărută pe parcursul dezvoltării aplicației, la nivelul bazei de date, nu afectează în nici o manieră nivelurile superioare.

```
class ProductRepository
{
    protected $model;

    public function __construct(Product $product)
    {
        $this->model = $product;
    }

    public function index() {...}

    public function show($id) {...}

    public function store(ProductRequest $request) {...}

    public function update(ProductUpdateRequest $request, $id){}

    public function delete(Product $product) {...}
}
```

Figura 14 – Exemplu de clasă pentru tiparul „Repository Pattern”

Lucrul cu baze de date s-a realizat prin intermediul *Eloquent*², obiect pentru mapare de date relaționale. Acest obiect pune la dispoziție o serie de facilități printre care crearea bazei de date prin intermediul migrațiilor.

² Modelele *Eloquent* sunt modele care extind clasa `Illuminate\Database\Eloquent\Model`. În proiectarea arhitecturală MVC, modelele sunt utilizate pentru a interacționa cu sursele de date. ORM-ul *Eloquent* este un Mapper Obiect Relational dezvoltat de Laravel. Acesta implementează modelul de înregistrare activ și este folosit pentru a interacționa cu bazele de date relaționale. ORM *Eloquent* poate fi folosit în interiorul Laravel sau în afara Laravel, deoarece este un pachet complet pe cont propriu.

```

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

```

Figura 15 – Exemplu de migrație

De asemenea *Eloquent*, ajută și la crearea de modele destinate legăturii dintre obiectul din baza de date și obiectul necesar transferului către interfața client.

```

class User extends Authenticatable
{
    use HasApiTokens, Notifiable;
    protected $fillable = [
        'name', 'email', 'password', 'confirmation_token', 'chair_id',
        'isAdmin'
    ];
    protected $hidden = [
        'password', 'remember_token',
    ];
    public function chair()
    {
        return $this->belongsTo(Chair::class);
    }
}

```

Figura 16 – Exemplu Modelul User

Pentru a trimite datele, cu scopul de a fi procesate pe partea de client, am folosit controlere. Acestea au rolul de a delega acțiuni, de a răspunde cererilor venite de la client și de a organiza comportamentul aplicației în clase specifice, care să se ocupe cu răspunderea cererilor aflate în strânsă legătură cu datele cerute.

```

class ProductsController extends Controller
{
    protected $productRepository;

    public function __construct(ProductRepository $productRepository)
    {
        $this->productRepository = $productRepository;
    }

    public function index()
    {
        $products = $this->productRepository->index();
    }

    public function show(Product $product)
    {
        return response()->json([
            'data' => $product->load('allergens', 'photos'),
            'errors' => null,
            'status' => Response::HTTP_OK,
        ], Response::HTTP_OK);
    }
}

```

Figura 17 – Exemplu Controller Produs

Atunci când se realizează o cerere de autentificare / înregistrare utilizatorului ii se creează un identificator unic pe baza căruia va putea face ulterior cereri pentru a obține date.

```

public function login(UserRequest $request)
{
    $user = User::where('email', $request->input('email'))->first();
    if (!$user) {
        $name = explode('@', $request->input('email'))[0];
        $user = User::create([
            'name' => $name,
            'email' => $request->input('email'),
            'confirmation_token' => str_random('30') . time(),
        ]);
    } else {
        $user->confirmation_token = str_random('30') . time();
        $user->save();
    }
    $chair_id = $request->input('chair_id');
    Mail::to($user)->send(new ConfirmationEmail($chair_id, $user->confirmation_token));
    return response()->json([
        'data' => $user,
        'errors' => null,
        'status' => Response::HTTP_CREATED,
    ], Response::HTTP_CREATED);
}

```

Figura 18 – Modul în care se înregistrează /autentifică un utilizator

Pentru a controla utilizatorii, care se înregistrează în sistem se cere confirmarea printr-un email a unui cod unic de acces. Atunci când utilizatorul trimite acest identificator primește în

schimb dreptul de a vizualiza datele și de a accesa funcționalitățile aplicației.

```
public function verify($chairId, $confirmation_token)
{
    $user = User::where('confirmation_token', $confirmation_token)-
>first();
    if ($user) {
        $user->confirmation_token = null;
        $user->chair_id = $chairId;
        $user->save();
        Auth::login($user);
        $loggedInUser = Auth::user();
        $success['token'] = $loggedInUser->createToken('Orderino')-
>accessToken;
        return response()->json([
            'data' => $success,
            'errors' => null,
            'status' => Response::HTTP_OK,
        ], Response::HTTP_OK);
    } else {
        return response()->json([
            'data' => null,
            'error' => 'Unauthorized',
            'status' => Response::HTTP_UNAUTHORIZED,
        ], Response::HTTP_UNAUTHORIZED);
    }
}
```

Figura 19 – Modul în care se face verificarea utilizatorilor

Totodată în cadrul aplicației am dorit restricționarea acțiunilor utilizatorilor care nu dețin suficiente drepturi, astfel am reușit să separ funcționalitățile destinate administratorilor de funcționalitățile destinate utilizatorului obișnuit.

```
class ProductRequest extends FormRequest
{
    public function authorize()
    {
        return Auth::user()->isAdmin == true;
    }
    public function rules()
    {
        return [
            'name' => 'required|string|max:50',
            'ingredients' => 'required|string|max:50',
            'category_id' => 'required',
            'photo' => 'required|image|mimes:jpeg, bmp, png, jpg',
            'allergens' => '',
            'price' => 'required|numeric|min:0'
        ];
    }
}
```

Figura 20 – Exemplu de verificare a cererii

Prin patternul utilizat, abstractizarea conexiunii la baza de date și prin intermediul mecanismului de „*Dependency Injection*” s-a reușit crearea unei aplicații modulare. Având această structură se pot aduce îmbunătățiri semnificative aplicației cu un efort minim.

Concluzii

Atunci când am conceput lucrarea am folosit cunoștințele însușite de-a lungul celor trei ani de facultate, dar și timpului folosit pentru studiul individual. Printre materiile care m-au călăuzit în aceasta activitate se regăsesc:

- Programare orientate-obiect; această materie m-a ajutat să înțeleg concepte precum încapsulare, polimorfism, moștenire dar și modalități prin care codul scris să fie mentenabil și reutilizabil.
- Tehnologii Web; în cadrul acestei materii am învățat bazele WEB-ului dar și proiectarea, programarea și întreținerea aplicațiilor WEB.
- Baze de date; materie care m-a familiarizat cu termeni extrem de necesari precum, Tabel, Cheie Stăina, Cheie Primara, Înregistrare s.a.m.d. Termeni ce stau la baza oricărei aplicații în care se dorește stocarea datelor din mediul virtual.
- Ingineria programării; materie care m-a învățat modul de lucru și cerințele atât din cadrul unui proiect cât și din cadrul unei firme. Aici am învățat despre termeni precum “Design Pattern”.

Consider că în final proiectul și-a atins obiectivele și am reușit să construiesc o aplicație care merită folosită pentru a diminua erorile umane care se produc în restaurante, astfel încât clienții să aibă parte de o experiență cât mai plăcută și administratorii să își reducă cât de mult posibil costurile.

Deoarece am folosit cele mai noi tehnologii aplicația poate fi oricând îmbunătățită prin adăugarea unor noi funcționalități care să respecte cerințele utilizatorilor odată cu trecerea timpului.

De asemenea aplicația poate fi extinsă cu un modul de plată și totodată cu un modul de răsplată a clienților fideli.

Bibliografie

1. Aplicație WEB - https://en.wikipedia.org/wiki/Web_application
2. Definiție și utilizări QR code - <https://www.digitalcitizen.ro/intrebari-simple-ce-sunt-codurile-qr-si-de-ce-sunt-ele-utile>
3. Documentație Laravel - <https://laravel.com/docs/5.6>
4. Motivele alegerii MySQL - <https://www.mysql.com/why-mysql/white-papers/mysql-and-hadoop-guide-to-big-data-integration/>
5. Diagramă baze de date - <https://erdplus.com/#/>
6. Generare de coduri QR - <https://www.the-qrcode-generator.com/whats-a-qr-code>
7. Documentație Angular <https://angular.io/docs>
8. Aplicații pe o singură pagină - <https://medium.com/@pshrmn/demystifying-single-page-applications-3068d0555d46>
9. Cum se utilizează un REST API - <https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>