Cuprins

Glosar de notații și abrevieri

API – Application Programming Interface
REST – Representational state transfer
HTTP – Hypertext Transfer Protocol
XML – EXtensible Markup Language
SDK – Software development kit
ADT - Android Developer Tools

Capitolul 1. Introducere

În lumea competitivă a business-ul din prezent, reducerea costurilor de operare și creșterea eficienței a devenit cel mai important lucru, mai ales că forța de muncă umană devine tot mai scumpă. Astfel, în acest context tehnologia informației și automatizarea proceselor a devenit treptat unul dintre cele mai importante instrumente în orice afacere.

Automatizarea diferitelor procese de business sau de producție contribuie semnificativ la creșterea afacerilor deoarece reduce costurile de operare și ajută la creșterea producției, concentrându-se pe minimizarea greșelilor cauzate de factorul uman și reducerea timpului de procesare. Aceste măsuri se aplică universal în afaceri, iar industria restaurantelor, ca orice alt domeniu, poate beneficia de aceste sisteme informatice pentru îmbunătățirea proceselor din restaurante, scăderea timpului de procesare a comenzilor și în final, garantarea unei experiențe cât mai plăcute pentru client. Această lucrare de licență are ca obiectiv principal dezvoltarea unui sistem distribuit de management al comenzilor dintr-un restaurant și evaluarea efectelor pe care această soluție le aduce în ceea ce privește performanța afacerii și gradul de satisfacere al clienților. Prezentul proiect se bazează pe 3 factori importanți: analiza cerințelor, implementare și evaluare.

1.1. Contextul proiectului

În general, oamenii merg la restaurant pentru relaxare, discuții și pentru a savura mâncăruri și băuturi într-un mod plăcut. De obicei la sfârșit de săptămână restaurantele sunt ocupate la capacitate maximă. În această periodă, oamenii trebuie să aștepte pe cineva un chlner pentru a face o comandă. În astfel de situații chelnerii sunt foarte ocupați, iar uneori pot uita să ia comenzile, să aducă băuturi sau chiar să servească o comandă greșită.

Deoarece forța de muncă este una dintre cele mai importante cheltuieli dintr-un restaurant și în același timpul un motiv principal pentru performanța scăzută, atunci o soluție informatică de automatizarea a unor procese și de management a comenzilor poate fi rezolvarea. O soluție de automatizare a unor procese poate ajuta în creșterea productivității, prin scăderea timpului și a eforturilor implicate în procese, chelnerii putând prelua mult mai practic și mai eficient comanda de la clienți, care datorită acestor soluții informatice automate poate ajunge la bucătărie mult mai repede decât în mod obișnuit. Toate aceste influențează direct satisfacția clientului, care în consecință se păstrează la același nivel sau chiar se poate îmbunătăți. O creștere a satisfacției clienților poate fi obținută de exemplu prin a avea posibilitatea de a comanda în limba natală, ceea ce ar fi foarte util în mediile multi-culturale, precum orașele mari. În același timp, posibilitatea de a consulta meniului unui restaurant, ofertele sau numărul de locuri disponibile în timp real înainte de a ajunge la locație este o facilitate care poate îmbunătăți semnificativ experiența clienților.

Lucrarea de față își propune să studieze impactul care îl poate avea asupra unui restaurant din punct de vedere al creșterii afacerii implementarea unor soluții informatice de automatizare a proceselor care țin de managementul comenzilor, a meniului și a experienței oferite clienților. În primul rând, proiectul a început de la o analiză a nevoilor restaurantelor și a clienților acestora. În al doilea rând, bazat pe aceste nevoi identificate

sistemul este proiectat, implementat și testat. În final, pentru partea de evaluare, sistemul este analizat pentru a putea fi studiate efectele introducerii sale în restaurant. Partea de evaluare urmărește dacă sistemul a scăzut timpul de procesare al comenzilor și dacă a contribuit la creșterea gradului de satisfacție al clienților.

xpressMenu se dorește a fi o solutie informatică completă pentru industria restaurantelor, adresându-se tuturor actorilor implicați: administratori de restaurante, personalul resturantelor și clienții acestora. În consecință, xpressMenu este compus din două componente, o componentă web adresată restaurantelor și o componentă mobilă adresată clienților acestor restaurante. Cu ajutorul componentei web, administratorii restaurantelor vor putea avea o evidență foarte clară asupra activității din restaurant, vor putea realiza cu ușurință modificări asupra meniului restaurantului, vor putea promova oferte speciale în rândul clienților care utilizeaza aplicația și vor putea obține informații despre preferințele clienților lor în funcție de comenzile pe care aceștia le fac.

În ceea ce privește componenta mobilă a sistemului, clienții resturantelor pot utiliza aplicația Android xpressMenu, prin intermediul căreia pot vizualiza meniul si să realizeze o comandă fără a mai fi nevoiți să aștepte un meniu din parte chelnerului, vor putea cere nota de plată, vor putea afla numărul de locuri libere sau ofertele dintr-un local înainte de a se decide să meargă acolo. Utilizatorii vor avea posibilitatea de a salva anumite localuri favorite și în funcție de fidelitatea lor să primească oferte din partea restaurantului.

1.2. Motivarea temei alese

Utilizarea unor sistem informatice de automatizare sau de gestiune și control au devenit în prezent o necesitate pentru orice tip de afacere, fiind cea mai bună soluție pentru a crește performanțele afacerii, iar industria restaurantelor nu face excepție de la această regulă. Un sistem informatic, pe lângă beneficiile care le aduce în automatizarea și îmbunătățirea proceselor, oferă și posibilitatea prezentării unor evidențe exacte asupra situației afacerii, evidențe care ajută administratorii de restaurante să ia decizii informate cu privire la evoluția afacerii pe termen lung.

Tema aleasă pentru această lucrare de licență are două motivări:

1. **Eficientizarea proceselor** într-un restaurant pentru a reduce costurile operaționale și pentru a crește productivitatea
2. **Creșterea gradului de satisfacție** al clienților prin îmbunătățirea serviciilor rezultate din implementarea soluției informatice, prin oferirea unei aplicații mobile care să le permită consultarea meniului, plasarea mai rapidă a comenzilor, consultarea ofertelor și a numărului de locuri disponibile în restaurant chiar dacă aceștia nu se află la locație, dar și oferirea unui sistem de fidelizare și recompensare a clienților în funcție de comenzile acestora.

Avantajele utilizării acestui sistem informatic de management al activității restaurantelor prezinta avataje multiple pentru toți actorii implicați (administratori de restaurante, personalulul din restaurante și clienți).

Luând în considerare motivarea de la care a pornit acest proiect și avantajele prezentate anterior putem concluziona foarte ușor că un asemenea sistem informatic modern este soluția perfectă pentru eficientizarea activității și creșterea unei afacerii din industria restaurantelor, oferind un avantaj clar în raport cu competiția.

1.3. Structura lucrării

Această lucrare este compusă din 8 capitole după cum urmează:

Capitolul 1 (Introducere) prezintă contextul general al proiectului şi tematica propuse ca proiect de diplomă însoțită de motivarea acesteia, precum şi structura acestei lucrări. Contextul general al proiectului prezintă situația actuală în ceea ce privește managementul în cadrul unui restaurant și prezintă avantajele pe care le-ar putea aduce implementarea unei soluții informatice în acest cadru.

Capitolul 2 (Obiectivele Proiectului) prezintă obiectivul general al sistemului informatic care face obiectul prezentei lucrări. De asemenea, capitolul mai prezintă obiectivele specifice ale sistemului însoțite de cerințele funcționale și non-funcționale ale sistemului, precum și detalierea modul în care aceste obiective și cerințe au fost stabilite în raport cu publicul țintă al sistemului proiectat.

Capitolul 3 (Studiu Bibliografic) descrie contextul apariției sistemelor informatice pentru gestiunea activității restaurantelor, precum și abordările existene în momentul de față pe piață împreună cu caracteristicile lor. Soluțiile evidențiate în acest capitol sunt similare cu sistemul care este prezentat în lucrarea de față, iar în finalul capitolului sunt oferite câteva concluzii comparative între avantajele sistemelor existente și sistemul xpressMenu.

Capitolul 4 (Analiză și Fundamentare teoretică) prezintă principiile funcționale ale sistemului implementat. Primul subcapitol prezintă arhitectura conceptuală și modelul de funcționare propus pentru implementarea sistemului, împreună cu toate componentele care îl constituie. De asemenea, este realizată o analiză detaliată a tehnologiilor și tool-urilor folosite pentru implementarea sistemului, dar și principalele avantaje ale folosirii lor.

Capitolul 5 (Proiectare de Detaliu și Implementare) cuprinde structura generală a sistemului și descrierea de implementare a principalelor componente ale acestuia: serviciul web de pe server care constituie punctul central al sistemului, aplicația mobilă dedicată clienților și aplicația web dedicată personalului restaurantului. De asemenea, sunt prezentare principalele cazuri de utilizare, structura bazei de date, diagramele de clase împreună cu o descriere a claselor și a metodelor mai importante ale sistemului.

Capitolul 6 (Testare și Validare) cuprinde partea de testare şi validare a sistemului, care urmărește cerințele de performanță și funcționalitățile oferite pentru toate componentele sistemului.

Capitolul 7 (Manual de Instalare și Utilizare) prezintă modalitățile prin care aplicația poate fi instalată și instrucțiunile de utilizare.

Capitolul 8 (Concluzii) reprezintă un set de concluzii împreună cu îmbunătățirile care se pot aduce acestui proiect.

Capitolul 2. Obiectivele Proiectului

Acest capitol prezintă obiectivul general al lucrării, precum și obiectivele specifice. Obiectivul general al lucrării îl constituie proiectarea și implementarea unui sistem de management al meniului și a comenzilor unui restaurant, cu scopul de a eficientiza procesele din restaurant și de a crește gradul de satisfacție al clienților.

2.1. Obiective generale

Scopul principal al acestui proiect este de a proiecta și dezvolta o soluție informatică completă care să ofere clienților unui restaurant posibilitatea de a vizualiza meniului și a realiza comenzi folosind o aplicație mobilă Android. Acest obiectiv presupune implementarea unor funcționalități prin care utilizatorul aplicației să poată avea acces la meniul unui restaurant folosind o aplicație instalată pe dispozitivul mobil personal. Pentru a avea posibilitatea de a accesa meniul și a plasa comenzi, aplicația mobilă ar trebui sa comunice cu un serviciu care să răspundă la cererile lansate, dar și să informeze, dacă este cazul, personalul restaurantului asupra acțiunilor utilizatorului aplicației (Exemplu: lansarea unei comenzi, cererea notei de plată, etc).

Odată ajunși la un restaurant, clienții vor putea utiliza aplicația xpressMenu pentru a scana un cod QR aflat pe masă, această acțiune având dublu rol. În primul rând prin intermediul acestui cod QR este identificată locația clientului în restaurant de către sistemul informatic, care va informa personalul restaurantului prin evidențierea pe aplicația dedicată web acestora din urmă. Un al doilea rol al acestui cod QR îl reprezintă identificarea restaurantului de către aplicația mobilă, care acum poate trimite o cerere către serviciul din backend pentru a primi informații generale despre restaurant, dar și meniului acestuia. După această fază preliminară, clientul are acum posibilitatea să vizualizeze atât informații despre restaurant, cât și meniul acestuia pe dispozitivul mobil personal și să plaseze o comandă. Comanda va fi vizualizată imediat de către chelner pe aplicația web dedicată restaurantelor.

Aplicația web dedicată restaurantelor poate fi accesată și de către administratorul restaurantului care poate realiza modificări la meniu, poate vizualiza statusul comenzile în timp real sau poate modifica informațiile generale despre restaurant.

2.2. Obiective specifice

Obiectivele specifice reprezintă atât funcționalități necesare implementării obiectivului general, cât și funcționalități care sunt necesare utilizării cât mai facile a aplicației mobile de către clienți, pentur a determina atât creșterea gradului de utilizare al aplicației cât și gradul de satisfacție în ceea ce privește experiența clientului în cadrul restaurantului. Aceste funcționalități reflectă cerințelor sistemului care reprezintă așteptările utilizatorului din partea sistemului, mai exact capabilități și constrângeri la care un sistem trebuie să se conformeze. Identificarea cerințelor este pasul cel mai important în dezvoltarea sistemului, deoarece influențează între procesul de dezvoltare al sistemului și poate pune dintr-o fază incipientă sub semnul întrebări finalizarea cu succes a sistemului. Datorită acestor aspecte am utilizat diverse metode pentru determinarea cât mai concretă a cerințelor sistemului din perspectiva tuturor actorilor implicați.

2.2.1. Sondaj în rândul potențialilor utilizatori

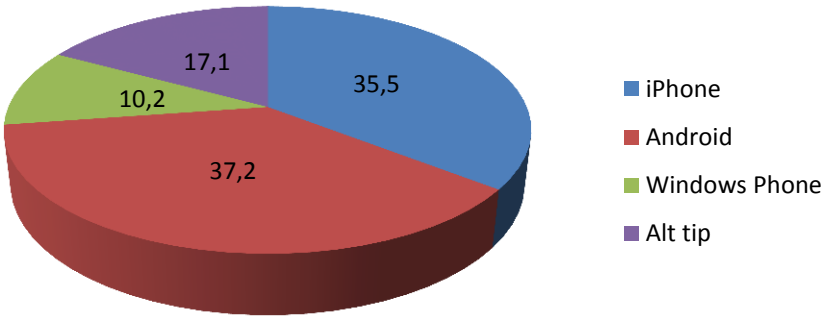
Una dintre modalitățile utilizate pentru a determina cerințele aplicației mobile utilizate de către clienții restaurantelor a fost realizarea unui sondaj online.

Gradul de utilizare al aplicației de către clienți ține foarte mult de înțelegerea preferințelor acestora și de proiectarea unei aplicații în acord cu așteptările și nevoile lor. Astfel, pentru a colecta cât mai multe informații posibile despre potențialii utilizatori ai aplicației, am realizat un sondaj pentru a înțelege tendințele acestora și a colecta informații care să ajute la stabilirea funcționalităților aplicației.

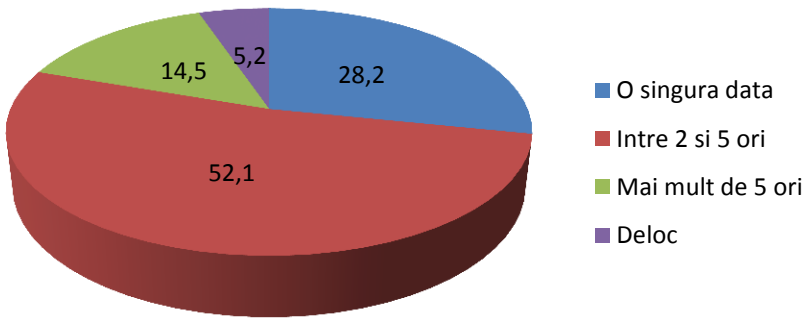
Sondajul a fost implementat cu șase întrebări obiective cu răspuns unic selectat din mai multe variante predefinite și în final o întrebare subiectivă cu răspuns deschis, astfel încât să putem în urma sondajului să concluzionăm asupra preferințelor potențialilor utilizatori în ceea ce privesc anumite aspecte ale aplicației. De asemenea, în finalul sondajului am colectat o serie de informații suplimentare (vârstă și ocupație) care să ne ajute la stabilirea grupului țintă al aplicației. Sondajul a fost promovat online în rândul unor persoane aleatorii timp de două săptămâni, iar criteriul de care s-a ținut cont în considerarea răspunsurilor pentru rezultatul final a fost gradul de frecventare al restaurantelor. Astfel, ne-am asigurat că rezultatul final al sondajului reflectă preferințele unor potențiali utilizatori ai aplicației.

În continuarea acestui capitol vom prezenta întrebările din cadrul sondajului, rezultatele obținute și interpretarea acestora.

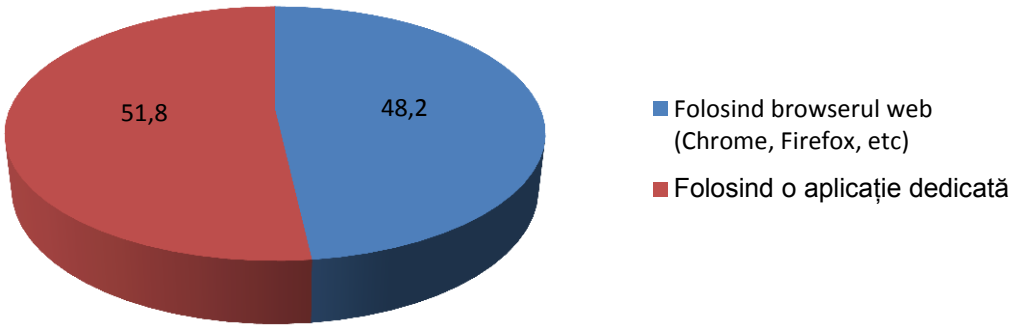
Întrebarea 1: Ce tip de telefon mobil dețineți?



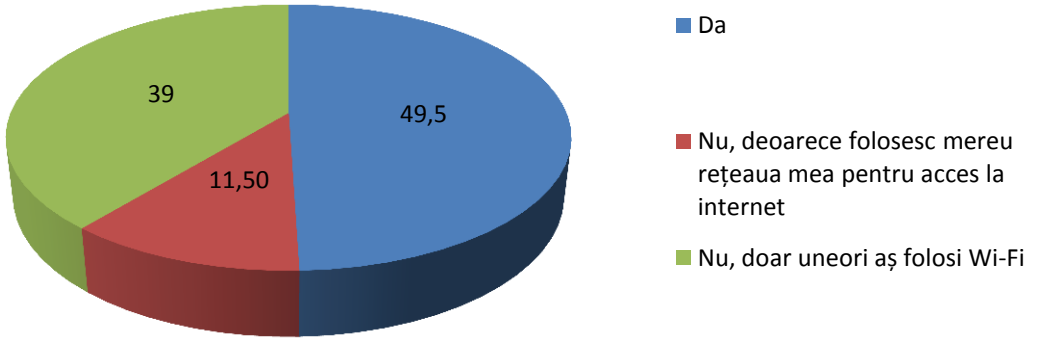
Întrebarea 2: De câte ori ați fost la un restaurant/pizzerie în ultima lună?



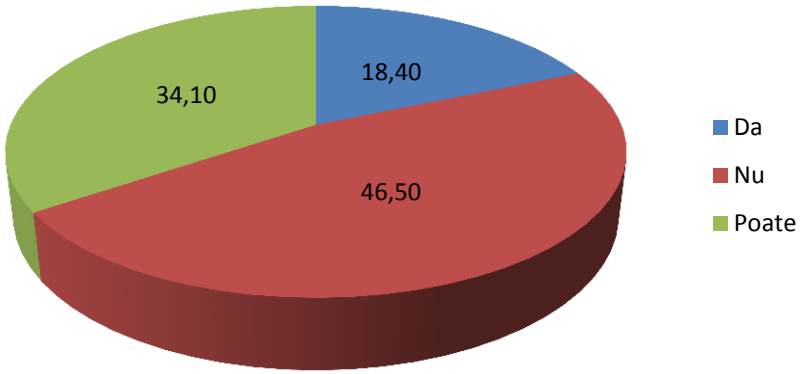
Întrebarea 3: Localul (restaurant/pizzerie) vostru preferat va ofera posibilitatea sa vizualizați meniul digital pe telefonul vostru mobil. Cum ați prefera să faceți acest lucru?



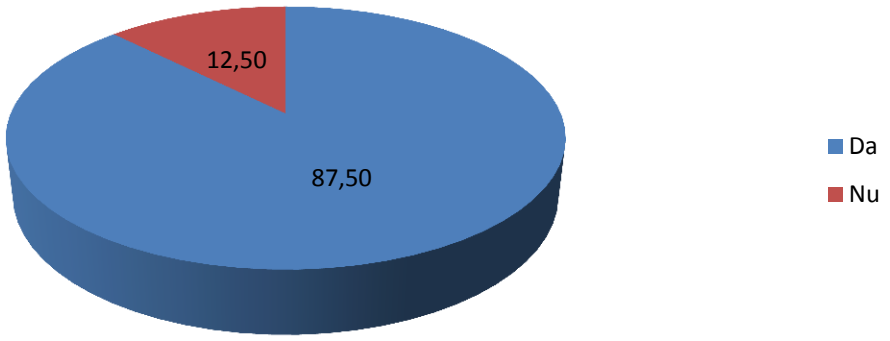
Întrebarea 4: Restaurantul în care vă aflați vă oferă acces gratuit la internet prin Wi-Fi. V-ați folosi întotdeauna de Wi-Fi pentru a putea accesa meniul digital al restaurantului?



Întrebarea 5: Daca aplicația de vizualizare a meniului digital v-ar cere să activați GPS-ul cât sunteți în restaurant, ați face acest lucru?



Întrebarea 6: Ați fi mai încurajat să utilizați o astfel de aplicație de vizualizare a meniului și de comandă în restaurantul vostru favorit dacă ați primi diferite oferte/reduceri în funcție de fidelitatea dumneavoastră (numărul de vizite în restaurant, numărul de comenzi, etc)?



Rezultatele acestui sondaj cu privire la modul de utilizare și preferințele tehnologice ale potențialilor utilizatori ai aplicației xpressMenu au fost luate în considerare la stabilirea funcționalităților secundare ale aplicației, dar și în stabilirea tehnologiilor utilizate în dezvoltarea acestui sistem informatic.

Întrebarea 1 și **întreparea 3** au fost principalele întrebări ale căror rezultate au fost esențiale în stabilirea tehnologiilor utilizate. Din **întrebarea 1** putem observa preferințele utilizatorilor spre sistemele de operare mobile Android și iPhone. Datorită numărului mare de dispozitive mobile compatibile și a flexibilității oferite din punct de vedere al dezvoltării s-a ales platforma Android, iar din întrebarea 3 s-a observat tendința utilizatorilor spre aplicații dedicate în detrimentul celor web care rulează într-un browser.

2.2.2. Cerințe funcționale

Cerințele funcționale descriu capabilitățile pe care trebuie să le îndeplinească sistemul, iar acestea descrierile lor trebuie să fie complete și prezentate din perspectiva utilizatorului.

Cerințele funcționale ale sistemului distribuit au fost stabilite atât în urma analizării rezultatelor sondajului realizat în rândul potențialilor utilizatori ai aplicației mobile xpressMenu, cât și în urma unei analize asupra nevoilor administratorilor de restaurante și a dificultăților întâmpinate de aceștia și de personalul restaurantului în ceea ce privesc diferitele procese operaționale. Cerințele functionale ale sistemului, care nu reflectă obiectivul principal al prezentei lucrări, se pot organiza în trei secțiuni în funcție de perspectiv celor trei actori principali ai sistemului: clientul restaurantului, ospătarul și administratorul restaurantului.

Tabel 2.1 prezintă cerințele funcționale prezentate din perspectiva clientului restaurantului și utilizator al aplicației mobile xpressMenu, cerințe care au fost identificate dupa analiza rezultatelor sondajului prezentat în subcapitolul anterior.

Tabel 2.1 Cerințele funcționale din perspectiva clientului restaurantului

	Descrierea funcționalităților
CF-1	Utilizatorul trebuie să aibă posibilitatea de a se înregistra în aplicație.
CF-2	Utilizatorul trebuie să poată să se autentifice în aplicație.
CF-2.1	Odată ce utilizatorul s-a autentificat în aplicație, trebuie să rămână autentificat până în momentul în care se deloghează.
CF-2.2	Utilizatorul care nu este autentificat în aplicație are dreptul doar să vizualizeze restaurantele și informațiile generale despre acestea.
CF-3	În eventualitatea în care utilizatorul nu-și mai amintește parola contului, acesta trebuie să aibă posibilitatea de a o recupera.
CF-4	Utilizatorul trebuie să aiba posibilitatea să se identifice în restaurant cu masa la care este așezat prin scanarea codul-ui QR aflat pe masă
CF-5	Utilizatorul trebuie să aibă posibilitatea de a vizualiza meniul restaurantului împreună cu toate categoriile de produse
CF-6	Utilizatorul trebuie să aibă posibilitatea sa viuzalizeze toate produsele dintr-o categorie selectată, împreună cu denumirea, cantitatea, prețul și poza acestora.
CF-7	Utilizatorul trebuie să aibă posibilitatea să realizeze o cautare rapidă după cuvinte cheie în cadrul meniului.
CF-8	Utilizatorul trebuie să aibă posibilitatea să-și gestioneze comanda, prin adăugarea de produse la comandă, specificând numărul de bucăți și diferite observații, prin ștergerea și editarea produselor deja existente.
CF-9	Utilizatorul trebuie să aibă posibilitatea să plaseze o comandă către chelner cu produselor selectate anterior.
CF-10	Utilizatorul trebuie să aibă posibilitatea să evalueze serviciile restaurantului.
CF-10.1	Utilizatorul trebuie să aibă posibilitatea să marcheze un anumit restaurant ca și "favorit", care să îi permita vizualizarea oricând a meniului sau primirea unor oferte special.
CF-10.2	Utilizatorul trebuie să aiba posibilitatea de a vizualiza informații despre restaurant împreună cu numele complet, descrierea, specificul restaurantului, adresa, poza, datele de contact și evaluările altor clienți.

Tabel 2.2 și Tabel 2.3 prezintă cerințele funcționale ale aplicației destinate ospătarilor, respectiv administratorului de restaurant. Aceste cerințe au fost stabilite în urma unei analize asupra nevoilor restaurantelor, dar și în urma corelării cu cerințele funcționale stabilite din perspectiva clienților.

Tabel 2.2 Cerințele funcționale din perspectiva ospătarului

	Descrierea funcționalităților
CF-1	Ospătarul trebuie să poată să se autentifice în aplicație.
CF-2	Ospătarul trebuie să aibă posibilitatea să vizualizeze mesele dintr-un restaurant
CF-3	Ospătarul trebuie să aibă posibilitatea să vizualizeze comenzile venite din partea clienților.
CF-3.1	Ospătarul trebuie să poată vizualiza informații generale despre comanda: masa de unde a venit comanda, produsele comandate și valoarea totală a comenzii.
CF-3.2	Ospătarul trebuie să aibă posibilitatea de a gestiona comenzile venite: sa adauge

	și să elimine produse sau să modifice cantitatea unui anumit produs comandat.
CF-4	Ospătarul trebuie să poată fi notificat asupra cerințelor venite din partea clienților
CF-3	În eventualitatea în care chelnerul nu-și mai amintește parola contului, acesta trebuie să aibă posibilitatea de a o recupera.

Tabel 2.3 Cerințele funcționale din perspectiva administratorului de restaurant

	Descrierea funcționalităților
CF-1	Administratorul trebuie să poată să se autentifice în aplicație.
CF-2	Administratorul trebuie să aibă posibilitatea de a gestiona informațiile generale despre restaurant: nume, descriere, adresa, date de contact, specific și poze.
CF-3	Administratorul trebuie să aibă posibilitatea să vizualizeze și să gestioneze mesele dintr-un restaurant: să adauge sau să elimine mese.
CF-4	Administratorul trebuie să aibă posibilatea să gestioneze categoriile de produse, produsele și meniurile restaurantului.
CF-5	Administratorul trebuie să aibă posibilitatea să gestioneze ospătarii restaurantului
CF-6	Administratorul trebuie să aibă posibilitatea să vizualizeze rapoarte cu privire la detaliile comenzilor realizate într-un anumit interval de timp specificat.
CF-7	Administratorul trebuie să aibă posibilitatea de a oferi reduceri clienților fideli.
CF-8	Administratorul trebuia să aibă posibilitatea de a trimite notificări cu oferte special clienților care au marcat restaurantul său ca “favorit”.
CF-9	În eventualitatea în care ospătarul nu-și mai amintește parola contului, acesta trebuie să aibă posibilitatea de a o recupera.

2.2.3. Cerințe non – funcționale

Spre deosebire de cerințele funcționale, cele non-funcționale reprezintă proprietăți și impun constrângeri asupra sistemului. Acestea specifică atributele sistemului și nu ceea ce sistemul trebuie să facă.

Aplicația mobilă xpressMenu va fi utilizată de un număr mare de utilizatori aleatorii care vor folosi multe dispozitive diferite, deci cerințe funcționale ca utilizabilitate și adaptabilitate dobândesc din această perspectivă o importanță la fel de mare pentru aplicație ca și cerințele funcționale. În cadrul sistemului de față cele mai importante cerințe non-funcționale care au fost identificate și care sunt descrise în continuarea acestui subcapitol sunt: utilizabilitatea, adaptabilitatea, performanța, disponibilitatea, scalabilitatea și securitatea.

2.2.3.1. Utilizabilitatea

Utilizabilitatea reflectă gradul de ușurința al unui sistem în ceea ce privește procesul său de învățare și utilizare. Este o cerință non-funcțională la fel de importantă ca și cele funcționale, deoarece poate fi elementul care face diferența între o aplicație de succes și un eșec total. În consecință este o cerință care influențează în mod direct numărul de utilizatori ai aplicației în raport cu alte soluții similare.

Utilizabilitatea unei aplicații este influențată semnificativ de designul cât mai prietenos al aplicației (user-friendly) dar în egală măsură și de gradul de intuitivitate în

utilizare al acesteia sau de modului cât mai facil de navigare în cadrul aplicației. Dacă potențialii utilizatori ai aplicației se vor lovi de o interfață greu de navigat sau prea încărcată și confuză, atunci chiar dacă aplicația ar fi cea mai bună de pe piață în ceea ce privesc cerințele funcționale ar avea foarte mult de pierdut în ceea ce privește numărul de utilizatori.

Deoarece utilizatorii aplicației nu pot fi instruiți înainte de folosirea aplicației, utilizabilitatea devine o caracteristică critică a sistemului. În acest context, am pornit proiectarea interfeței cu utilizatorul de la trei principii de baza ale utilizabilității [1]: focus pe tipurile de utilizatori și cerințele lor, măsurători empirice și proiectare iterativă. Am pus un accent deosebit pe dezvoltarea interfeței cu utilizatorul, astfel utilizarea aplicației conține pași simpli și elemente vizuale sugestive. Toate acestea oferă un grad de intuitivitate ridicat utilizării aplicației, nefiind necesar pentru utilizator să consulte documentații sau manuale de utilizare.

2.2.3.2. Adaptabilitatea

După cum prezentam la începutul acestui capitol, aplicația va fi utilizată de un număr ridicat de persoane pe diferite dispozitive mobile. Tendința de creștere a utilizării dispozitivelor mobile de către tot mai multe persoane, gama largă de produse cu specificații tehnice diferite și fragmentarea ridicată a sistemelor de operare destinate acestui sector fac din adaptabilitate o cerință non-funcțională importantă în procesul de proiectare software. Toate aceste aspecte au fost avute în vedere la proiectarea aplicației și a interfeței acesteia pentru a oferi o independență de specificații tehnice ca versiunea sistemului de operare, dimensiunea sau rezoluția ecranului.

2.2.3.3. Performanța

Cerințele de performanță se referă de obicei la timpul de răspuns al aplicației la diferitele cereri venite din partea utilizatorilor: executare de interogări, trimitere de comenzi, afișarea meniului, generarea și afișarea diferitelor rapoarte, etc. Timpul de răspuns al sistemului trebuie să fie optim indiferent de numărul de clienți aflați în restaurant, de numărul de comenzi lansate sau de dimensiunea bazei de date care odată cu implementarea sistemului în tot mai multe restaurante va crește considerabil. Indiferent de aceste aspecte, sistemul trebuie să se asigure că timpul de răspuns la cererea vizualizării unui meniu de către client sau la informațiile despre un anumite produs este păstrat în limite tolerabile. De asemenea, procesul de realizare a unei comenzi trebuie să dureze în medie 2 minute, iar timpul necesar ca această comandă sau orice altă notificare din partea clientului să ajungă la chelner trebuie să fie de maxim 30 de secunde.

2.2.3.4. Disponibilitatea

Disponibilitatea este o cerință non-funcțională care descrie măsura în care sistemul trebuie să funcționeze pentru utilizatori. Având în vederea faptul că acest sistem se dorește a fi o alternativă la sistemul manual de procesare a comenzilor, atunci întreaga activitate a restaurantului va depinde de sistem informatic. În consecință, disponibilitatea maximă a sistemului este o importantă cerință non-funcțională care trebuie avută în vedere la proiectarea arhitecturii sistemului. Această disponibilitate maximă se traduce într-o funcționare la standarde optime a sistemului 24 de ore din 24, iar un alt aspect care

trebuie avut în vedere este timpul de recuperare al sistemului, care nu trebuie să fie mai multe de 10-15 minute.

2.2.3.5. Scalabilitatea

Scalabilitatea este o cerința non-funcțională care descrie capacitatea unui sistem de a suporta un volum mare de încărcare sau de a permite extinderea sa fără costuri prea ridicate. Un sistem informatic se consideră scalabil dacă performanțele sale rămân constant și nu apar defecțiuni în momentul când volumul de date pe care îl procesează devine mai mare, mai ales când discutam despre o creștere bruscă a volumului într-un interval de timp foarte scurt. În cazul sistemului nostru, acesta trebuie să își păstreze același grad de performanță și fiabilitate indiferent de numărul utilizatorilor care folosesc aplicația mobilă sau de numărul restaurantelor care au implementat acest sistem de management al meniurilor și comenzilor.

2.2.3.6. Securitatea

Securitatea unui sistem informatic Sonda. Principalele caracteristici care stau la baza acestei cerințe non-funcționale sunt: confidențialitatea, disponibilitatea și integritatea datelor.

Securitatea este un aspect foarte important și avut în vedere la orice sistem informatic, dar aici deține un rol crucial deoarece se dorește ca funcționalitatea aplicației mobile xpressMenu de realizare a comenzilor să fie disponibilă doar în incinta restaurantului. În cazul în care această facilitate ar fi disponibilă în afara restaurantului, clienții ar putea să comande produse fără a fi prezenți fizic în restaurant. În consecință, verificarea locației clientului devine un aspect important în privința prevenirii folosirii aplicației în alte scopuri decât cele pentru care a fost intenționată.

Capitolul 3. Studiu Bibliografic

Acest capitol va expune principalele componente și funcționalități ale unui sistem informatic dedicat industriei restaurantelor. De asemenea vor fi prezentate și aplicații similare cu cea care face obiectul lucrării de față, fiind evidențiate comparativ avantajele și dezavantajele fiecăreia.

3.1. Soluții informatice pentru industria restaurantelor

În articolul [2] apare următoarea afirmație „Tehnologia și dezvoltarea permanentă a acesteia atinge fiecare aspect al vieții noastre, dar cu apariția dispozitivelor mobile și a cloud computing-ului impactul tehnologiei este mai puternic ca niciodată. Aceste progrese au avut un impact în toate aspectele care privesc viața noastră, iar unul din locurile în care vedem tot mai mult tehnologia este în industria alimentară și a restaurantelor."

Până acum câțiva ani, personalul din industria restaurantelor a arătat o reticență destul de ridicată în a utiliza tehnologia și progresele acesteia în activitatea lor zilnică, *însă în ultimii ani, odată cu conștientizarea importanței tehnologiei în ceea ce privește creșterea gradului de competitivitate, dar și pe fondul diminuării barierelor economico-financiare în ceea ce privește accesul la noile tehnologii, restaurantele au început treptat să adopte tehnologia pentru eficentizarea diferitelor procese specifice.*

Faza de tranziție tehnologică a surprins restaurantele trecând de la casa de marcat electronică, care reprezenta unul dintre primele contacte cu tehnologia, la sistemele informatice actuale de rezervări on-line sau de comandă. Prima revoluție tehnologică implementată în industria restaurantelor a fost bine cunoscutul POS (Point of Sale System). Dezvoltat la începutul anilor 1980, POS-ul s-a integrat foarte bine în industria restaurantelor și a automatizat îndatoririle ospătarilor și a personalului din bucătărie. Dintr-o dată preluarea comenzilor de la clienți și transmiterea acestora personal în scris de către ospătari la bucătărie nu mai reprezenta principala modalitate de procesare a comenzilor. Astfel, POS-ul a oferit o soluție tehnologică automatizată și mult mai eficientă din punct de vedere al timpului de procesare. [3]

În prezent, dominul IT oferă o varietate foarte mare de soluții tehnologice dedicate industriei restaurantelor, atât pentru eficentizarea activității personalului din cadrul restaurantelor cât și pentru clienții acestora. Printre principalele modalități prin care tehnologia a atins și a influențat viața clienților de restaurante putem aminti:

- **Prezența online a restaurantelor** – fie că discutăm de clasicele website-uri de prezentare a restaurantelor sau de importanța prezenței restaurantelor pe popularele rețele de socializare, tehnologia a ofert o modalitate excelentă restaurantelor de a se promova, iar clienților de a se informa cu privire la serviciile și oferta acestora.
- **Aplicații de comandă la domiciliu** – tehnologia a revoluționat deasemenea și modalitatea de realizare a banalelor comenzi la domiciliu, iar noile aplicații mobile cu ajutorul cărora putem realiza acest lucru oferă o eficiență îmbunătățită a întregului proces, atât pentru clienți cât și pentru restaurante.

- **Plata electronică** – după introducerea POS-urilor, plata electronică cu cardul a fost una dintre cele mai populare inovații tehnologice îmbrățișate de industria restaurantelor și de clienții acestora deopotrivă.
- **Aplicații de căutare a restaurantelor și clasificare a acestora** – primele aplicații destinate dispozitivelor mobile au oferit utilizatorilor posibilitatea de a căuta restaurante din proximitatea lor sau de a clasifica aceste restaurante în funcție de calitatea serviciilor oferite.

În ceea ce privește personalul restaurantelor, majoritatea au îmbrățișat tehnologia și în ceea ce privește automatizarea diferitelor procese și administrarea cât mai eficientă a activității. Astfel au fost dezvoltate aplicații software care au devenit foarte populare în rândul administratorilor de restaurante deoarece le ofera o modalitate mult mai facilă de gestionare a vânzărilor, a aprofizionării, a activității financiar-contabile sau a personalului.

Articolul [2] susține faptul că *tehnologia nu a fost niciodată mai intuitivă decât este astăzi, și nu poate decât să progreseze.* În același timp avertizează asupra modului de utilizare a acesteia, deoarece *dacă este folosită în locul nepotrivit și la momentul nepotrivit, poate face unui restaurant mai mult rău - așa cum este posibil în orice industrie. Cu o mai bună înțelegere a tehnologiei, industria are o mai bună șansă de a prospera. Și în vremuri grele, acele restaurante care se află în partea potrivită a ecuației vor avea o șansă mai bună de supraviețuire.* [2]

3.2. Sisteme similare existente pe piață

După cum am prezentat în subcapitolul anterior, tehnologia a pătruns treptat dar sigur și în industria restaurantelor, principalul scop fiind eficientizarea gestionării comenzilor și a proceselor de servire.

Primele sisteme software dedicate acestui sector au apărut sub forma unor soluții desktop dedicate ospătarilor și administratorilor de restaurante, prin intermediul cărora se pot gestiona exact și rapid comenzile din restaurant. Însă principalul dezavantaj al acestor soluții este faptul că sunt sisteme greoaie, dificil uneori de utilizat și necesită o investiție considerabilă pentru instalarea unui asemenea sistem.

În ceea ce privesc aplicațiile mobile dedicate industriei restaurantelor, acestea au explodat odată cu creșterea numărului de dispozitive mobile de pe piață și au început să fie foarte apreciate de utilizatori. Primele aplicații mobile s-au axat pe funcționalitatea de a asista utilizatorul în găsirea celui mai potrivit restaurant în funcție de locația acestuia sau de preferințe, iar treptat au început să înglobeze și alte funcționalități: realizarea de rezervări, evaluarea restaurantelor, asistarea utilizatorului în alegerea unui meniu sănătos sau recomandarea unor localuri pe baza preferințelor utilizatorului.

Din multitudinea aplicațiilor dedicate restaurantelor și a funcționalităților oferite de către acestea, se remarcă succesul aplicațiilor care oferă utilizatorilor posibilitatea de a plasa o comandă de la domiciliu. Spre deosebire de comanda clasică prin telefon, aceste aplicații oferă utilizatorilor libertatea de a vizualiza meniul complet al restaurantului și oferă flexibilitate în modul de plasare al comenzii.

Există și aplicații mai inedite care oferă gurmanzilor posibilitatea de încărca poze cu feluri de mâncare preferate și să împărtășească această experiență cu prietenii lor, aceasta fiind o metodă foarte eficientă de a primi recomandări de restaurante și meniuri. Alte aplicații oferă posibilitatea de a localiza sau de a primi recomandări de restaurante în

funcție de specificul acestora sau de preferințele utilizatorilor. Există și aplicații care își propun să ofere administratorilor o formă de fidelizare a clienților, iar acestora din urmă o serie de reduceri în restaurantele favorite.

Din cele prezentate mai sus se pot observa două moduri diferite în care tehnologia a influențat industria restaurantelor. Sistemul xpressMenu dorește să ofere o cale de mijloc, să îmbine funcționalitățile oferite de cele două abordări și să ofere o soluție care răspunde la nevoile tuturor actorilor implicați. În acest sens, în continuarea acestui capitol vor fi analizate soluții specifice celor două abordări identificate. În primul rând va fi analizat un sistem informatic denumit „Software gestiune restaurant" [4], iar din categoria aplicațiilor mobile sunt prezentate „OpenTable", „Savored" și „HipMenu".

3.2.1. Software gestiune restaurant

„Software gestiune restaurant" este un sistem informatic dezvoltat de firma românească SC Sunsys Software SRL din București, care se adresează localurilor de tip restaurant care doresc să își automatizeze procesul de gestiune a activității.

Acest software este unul dezvoltat pentru platforme de tip desktop, oferind soluții complete pentru restaurante, pornind de la gestiunea comenzilor de către ospătari, personalizare în funcție de locație, rapoarte dedicate administratorilor și tipărirea documentelor specifice (notă de plată, bon fiscal, note de recepție, facturi, avize, etc.).

Sistemul xpressMenu nu va implementa funcționalități de tipărire a documentelor specifice contabilității restaurantelor deoarece obiectivul său principal este de a oferi clienților posibilitatea de a realiza comenzi. Aceste funcționalități menționate anterior sunt de interes pentru restaurante și pot fi integrate ulterior în sistem, însă momentan nu au o implicație importantă în realizarea obiectivului principal.

În **Figura 3.1** poate fi observată interfața aplicației „Software gestiune restaurant" dedicată ospătarilor.

Figura 3.1 Interfață dedicată ospătarilor

În ceea ce privesc clienții restaurantelor, software-ul de față nu le oferă beneficii și nici nu influențează în mod direct experiența acestora în restaurant. Singura funcționalitate care putem spune că vizează clienții restaurantelor este posibilitatea aplicației de a utiliza carduri de fidelitate, dar beneficiile reale sunt tot axate către restaurante deoarece oferă posibilitatea automatizării și gestiunii exacte a acestui proces.

Sistemul xpressMenu dorește spre deosebire de sistemele actuale din piață să se focuseze mai mult pe clienții restaurantelor și pe experiența acestora din restaurant. În ceea ce privesc beneficiile restaurantului, sistemul pe care xpressMenu dorește să îl pună la dispoziția acestora își propune în primul rând să ofere un mod de utilizare mult mai facil și intuitiv. Iar datorită faptului că este bazat pe tehnologii web cerințele sistemelor necesare rulării acestei aplicații scade considerabil, scăzând în consecință și costurile financiare necesare instalării sistemului xpressMenu.

3.2.2. OpenTable

OpenTable este liderul mondial în rezervări online pentru industria de restaurante, având o rețea internațională de aproximativ 32000 de restaurante și peste de 15 milioane de rezervari online lunar. Rețeaua OpenTable pune în legătură restaurantele cu potențiali clienți, ajutându-i pe aceștia din urmă să gasească locația perfectă și pe restaurante să ofere servicii personalizate de calitate.

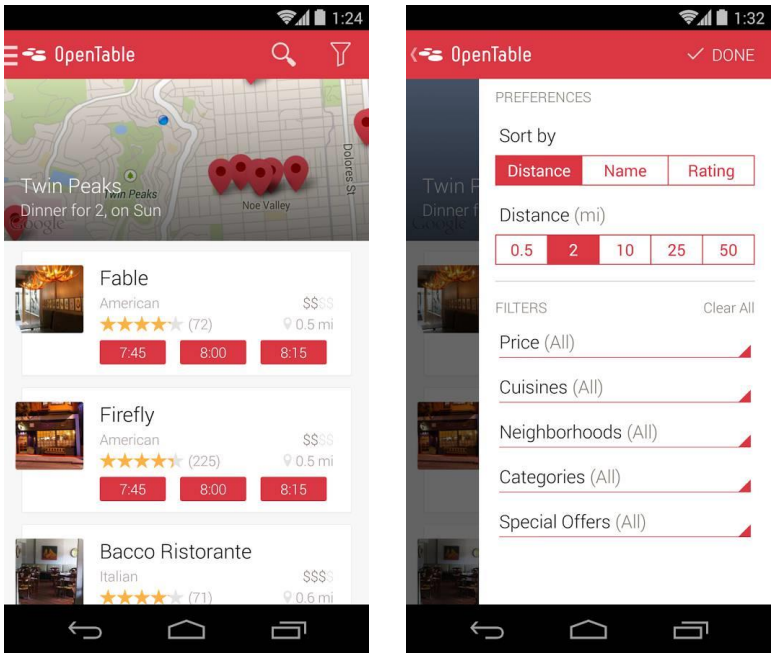


Figura 3.2 Interfața aplicației mobile OpenTable

OpenTable a luat naștere în anul 1998 sub forma unei platforme web, iar ulterior s-a extins și în zona dispozitivelor mobile prin dezvoltarea unor aplicații dedicate sistemelor de operare iPhone și Android. În **Figura 3.2** se poate observa interfața aplicației mobile OpenTable, fiind prezentate două capturi de ecran care exemplifică

funcționalitatea de căutare a restaurantelor din apropiere și posibilitatea setării unor filtre de căutare.

Principalele funcționalități ale aplicației mobile OpenTable sunt [5]:

- Căutarea restaurantelor din apropiere sau în funcție de diferite filtre
- Vizualizare informații restaurant: descriere, recenzii, preparate populare
- Realizarea unei rezervări
- Trimiterea de invitații către prieteni pentru o anumită rezervare
- Fidelizarea utilizatorilor în funcție de fiecare rezervare

OpenTable este cel mai răspândit serviciu de rezervări din lume, fiind prezent în foarte multe orașe din Statele Unite ale Americii, Canada, Germania, China, Japonia, Marea Britanie, Arabia Saudita, Mexico, Portugalia și alte câteva țări. Avantajul principal al soluției OpenTable este rețeaua de parteneri dezvoltată în decursul anilor și posibilitatea de utilizare atât a platformei web, cât și a aplicației mobile.

Aplicația oferă toate funcționalitățile necesare căutării, identificării și rezervării restaurantului perfect pentru utilizator, însă nu oferă nici o funcționalitate referitoare la realizarea de comenzi. Aplicația xpressMenu înglobează la rândul său aceste funcționalități de vizualizare a restaurantelor în funcție de locația utilizatorului stabilită prin intermediul GPS-ului și oferă în plus funcționalități pentru vizualizarea detaliată a meniului și gestionarea unei comenzi de către utilizator odată ajuns la restaurant.

OpenTable plănuiește să introducă pe platforma mobilă dedicată dispozitivelor Apple facilitatea de plată prin intermediul aplicației, funcționalitate care poate fi considerată pentru o dezvoltare ulterioară în cadrul sistemului xpressMenu.

3.2.3. Savored

Savored este o soluție care ajută restaurantele să atragă clientelă în anumite intervale de timp când fluxul de clienți este scăzut sau când sunt anulate anumite rezervări. Pentru a nu rămâne cu mese libere, proprietarii de restaurante au posibilitatea de a promova diferite oferte speciale către potențiali clienți prin intermediul platformei web sau a aplicației mobile dezvoltate de Savored.

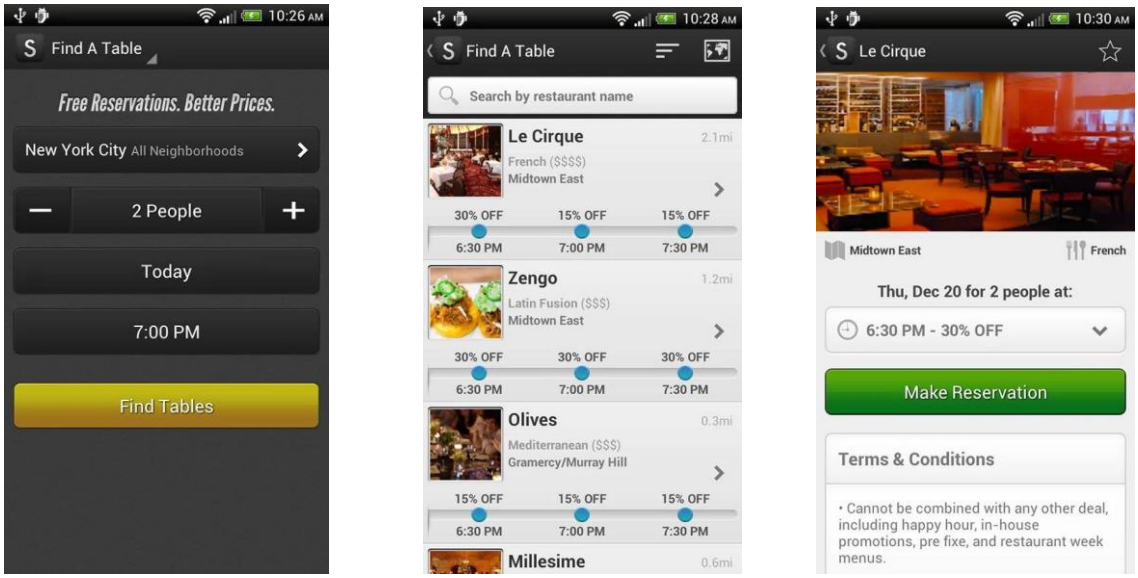


Figura 3.3 Interfața aplicației mobile Savored

Platforma Savored a luat naștere în anul 2010, iar în anul 2012 a fost achiziționată de către Groupon, liderul mondial în servicii de reduceri. Savored este utilizat de peste 1000 de restaurante din Statele Unite ale Americii care pun la dispoziția utilizatorilor aplicației cele mai bune oferte.

Utilizatorul începe prin a precizeaza când ar dori să facă o rezervare, iar aplicația ii oferă o listă cu restaurantele care au cea mai bună ofertă din punct de vedere al prețului. Când găseste localul și oferta preferată, utilizatorul are posibilitatea de a face o rezervare direct din aplicație. Unul dintre avantajele aplicației este faptul că utilizatorii pot beneficia de reduceri la restaurante fără a mai fi nevoiți să folosească diferite cupoane sau carduri de fidelitate.

În **Figura 3.3** pot fi vizualizate trei ecrane din interfața aplicației Savored, care exemplifică procesul de introducere a preferințelor legate de localul căutat, urmat de vizualizarea tuturor ofertelor disponibile în concordanță cu specificațiile anterioare, iar în al treilea ecran se pot vizualiza informații detaliate despre restaurantul ales și se poate realiza rezervarea.

Conceptul de fidelizare este foarte eficient în promovarea aplicației și atragerea clienților în restaurante, fiind prevăzut și în cerințele funcționale ale sistemului xpressMenu. Spre deosebire de aplicația Savored care se focusează doar pe a pune la dispoziția utilizatorilor cele mai bune oferte, pentru xpressMenu această funcționalitate este una complementara care să vină în ajutorul utilizatorilor și a restaurantelor, obiectivul principal fiind funcționalitățile referitoare la realizare comenzilor.

3.2.4. HipMenu

Comparativ cu aplicațiile prezentare anterior, obiectivul principal al aplicației HipMenu este de a oferi asistență utilizatorilor în a realiza comenzi la o locație precizată. Spre deosebire de HipMenu, aplicația xpressMenu se focusează cu precădere pe experiența clientului în incinta restaurantului și pe interacțiunea acestuia cu personalul și serviciile restaurantului.

Figura 3.4 Interfața aplicației HipMenu

HipMenu este o aplicație dezvoltată la Cluj – Napoca și reunește peste 20 de restaurante locale, oferind servicii de livrare la domiciliu sau de comandă în avans la pachet. Utilizatorul are posibilitatea de a căuta într-o listă de restaurante din proximitatea sa, să vizualizeze meniul și preparatele fiecărui restaurant în parte, iar apoi să plaseze comanda dorita. În decursul procesului de comandă utilizatorul specifică tipul comenzii: livrare la domiciliu sau la pachet cu ridicare personală de la restaurant.

O altă funcționalitate importantă a aplicației HipMenu, care îi oferă un avantaj în comaparație cu alte aplicații similare este posibilitatea de a realiza comenzi de grup. Fiecare utilizator are posibilitatea să își aleagă preparatele favorite de pe telefonul propriu, iar apoi administratorul grupului se va ocupa de trimiterea comenzii.

Deși HipMenu este o aplicație care are o creștere foarte bună a numărului de utilizatori datorită funcționalităților oferite și a interfeței grafice atractive, nu este un competitor direct al aplicației xpressMenu deoarece au obiective principale diferite. Cu toate acestea, funcționalitățile de bază și modul de prezentare al aplicației xpressMenu este la un nivel apropiat de cel al HipMenu și în plus oferă facilități suplimentare pentru restaurante.

3.3. Concluzii comparative între sistemele similare existente

În urma studiului realizat pe sisteme similare aplicației xpressMenu se poate observa că această nu poate fi încadradă într-o categorie anume, conceptul pe care îl implementează fiind unul nou.

Sistemul se dorește a fi unul care să ofere beneficii semnificative atât restaurantelor cât și clienților acestora. Prin intermediul aplicației mobile xpressMenu, se oferă clientului o experiență de servire îmbunătățită, acesta având un control mult mai exact asupra conținutului comenzii sale. Pe lângă funcționalitățile referitoare la vizualizarea digitală a meniului și la comenzi, sistemul dorește să ofere funcționalități suplimentare cum ar fi: vizualizarea numărului de locuri libere dintr-un restaurant, realizarea unei rezervări, posibilitatea de a beneficia de diferite reduceri.

Pe lângă beneficiile oferite clienților, sistemul xpressMenu se remarcă și prin funcționalitățile oferite restaurantelor, pe care puține dintre sistemele similare le oferă.

În **Tabel 3.1** este prezentată o comparație între funcționalitățile oferite de sistemul xpressMenu și sistemele similare analizate anterior.

Tabel 3.1 Prezentare comparativă sisteme similare

	Software gestiune restaurant	OpenTable	Savored	HipMenu	xpressMenu
Autentificare Înregistrare	Da	Da	Da	Da	Da
Autentificare folosind rețele de socializare	NU	NU	DA	DA	DA
Listare restaurante din proximitatea utilizatorului	NU	DA	DA	DA	DA
Restaurante favorite	NU	NU	DA	NU	DA
Efectura unei comenzi în incinta restaurantului	NU	NU	NU	NU	DA
Efecturea unei comenzi de la domiciliu	NU	NU	NU	DA	NU
Vizualizare informații generale restaurant	NU	DA	DA	DA	DA
Istoric comenzi	NU	NU	DA	DA	DA
Integrare rețele de socializare	NU	DA	DA	DA	DA
Contactare telefonică restaurant din aplicație	NU	NU	NU	DA	DA
Realizarea unei rezervări	NU	DA	DA	NU	DA
Localizarea pe hartă a restaurantului	NU	DA	DA	DA	DA
Vizualizarea meniului împreuna cu poze și informații	NU	DA	DA	DA	DA
Evaluarea serviciilor restaurantului	NU	DA	DA	DA	DA
Beneficii de fidelizare (reduceri, oferte speciale)	NU	NU	DA	NU	DA
Statistici cu privire la tendințele clienților *	NU	NU	DA	NU	DA
Promovarea unor oferte în rândul clienților favoriti *	NU	NU	DA	NU	DA
Preluare și gestionare comandă *	DA	NU	NU	DA	DA
Vizualizarea meselor din restaurant *	DA	NU	NU	NU	DA
Vizualizare notificări din partea clienților *	NU	NU	NU	DA	DA
Printare notă de plată *	DA	NU	NU	NU	NU

* Funcționalități compontei web dedicate ospătarilor și administratorilor de restaurante

Capitolul 4. Analiză şi Fundamentare Teoretică

Arhitectura unui sistem software constă în structurile lui, descompunerea în componente şi interfeţele şi relaţile dintre ele[1]. Arhitectura descrie atât aspectele statice cât şi cele dinamice ale sistemului software.

Alegerea unei soluții arhitecturale adecvate este primul pas și probabil unul dintre cele mai importante în ceea ce privește procesul de dezvoltare al unei aplicații software, deoarece este fundația aplicației și influențează ulterior întregul proces de dezvoltare. Odata stabilită arhitectura sistemului se va trece la alegerea tehnologiilor necesare implementării aplicației software, ținând cont de cerințele inițiale.

În contianrea acestui capitol vor fi analizare arhitectura și tehnologiile alese pentru implementarea aplicației xpressMenu.

4.1. Arhitectura sistemului

În implementarea acestui sistem informatic s-a ales arhitectura client – server, iar fiecare componentă a sistemului va avea propriul stil arhitectural. Aplicaţiile client-server reprezintă o metodă arhitecturală care se bazează pe un dialog între două categorii de entităţi (client, respectiv server) şi are ca scop furnizarea de informaţii, stocate pe server, către un utilizator – client.

Clientul – este entitatea care asigură interfaţa cu utilizatorul, lansează cereri de executare a unor operaţii către o entitate server şi prezintă utilizatorului într-o anumită formă datele primite de la server în urma executării operaţiei. Un client poate să efectueze cereri către mai multe servere simultan.

Serverul – este entitatea care recepţionează, interpretează şi execută cererile (operaţiile) lansate de clienţi, iar la final furnizează răspunsul către aceştia. Un server poate oferi un serviciu mai multor clienți (în mod concurent) și în același timp poate oferi mai multe tipuri de servicii (web, stocare de date, e-mail, etc).

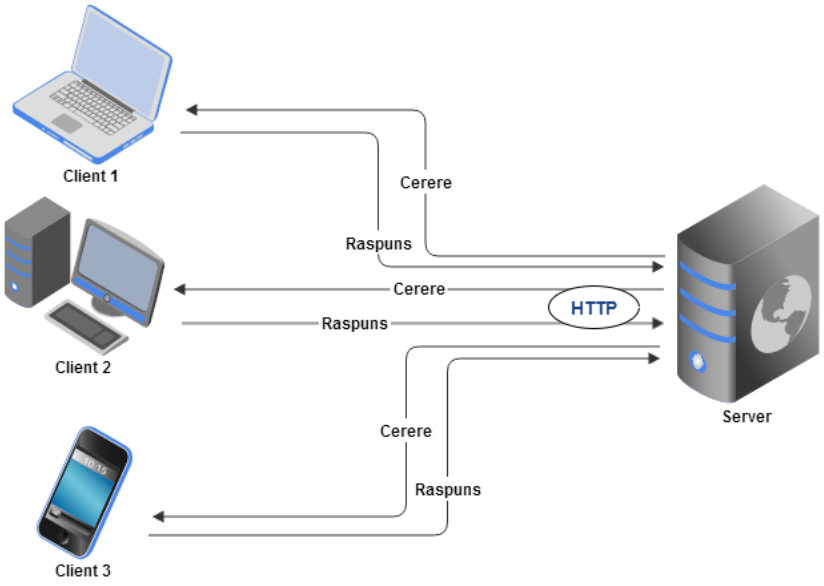


Figura 4.1 Arhitectura client - server

Cele două entităţi se pot regăsi sub formă de calculatoare diferite, sau pot convieţui pe acelaşi calculator. Comunicarea dintre cele două entități este de forma cerere-răspuns și se realizează cu ajutorul protocoalelor de comunicare (FTP, HTTP, etc).

Figura 4.1ilustrează entitațile implicare în arhitectura client – server.

Pentru proiectul prezentat în lucrarea de față clientul poate fi reprezentat de:

- Telefonul mobil al unui client din restaurant, care utilizează aplicația Android xpressMenu
- Laptop-ul administratorului restaurantului care utilizează aplicația web xpressMenu pentru a administra meniul restaurantului
- Telefonul mobil al unui chelner care primește notificarea unei comenzi

Serverul este reprezentat de un serviciu web de tipul Google Cloud Endpoint care rulează pe Google App Engine și care răspunde la cererile venite din partea clienților. Serverul primește cererile, le procesează, accesează baza de date dacă este nevoie, iar în final întoarce un răspuns clientului.

Clientul și serverul din lucrarea de față vor comunica prin protocolul HTTP, folosind metodele specifice: GET și POST. Datele vor fi transmise între cele două entități în formatul JSON, un format text care este complet independent de limbaj, ușor de parsat și generat de către mașini ceea ce face din JSON un limbaj ideal pentru interschimbarea datelor [2].

În **Figura 4.2** pot fi observate componentele principale ale sistemului și legăturile dintre ele.

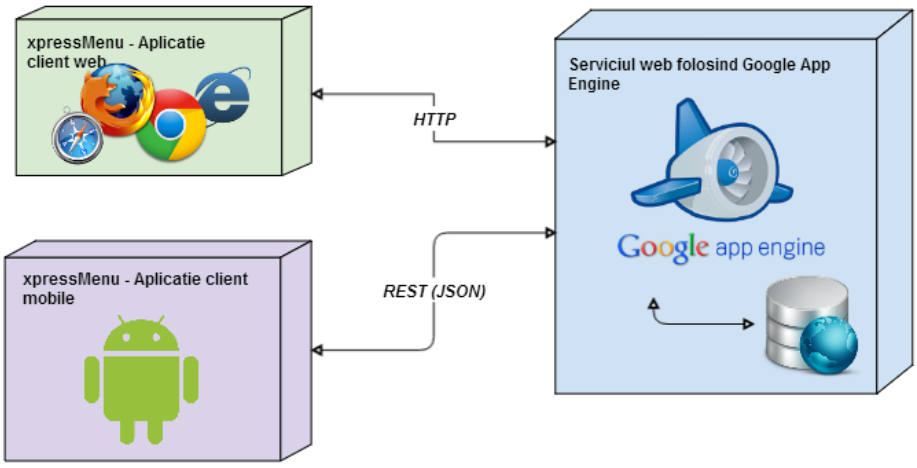


Figura 4.2: Componentele sistemului

Din **Figura 4.2** se poate observa ca sistemul de față este organizat sub forma unui sistem distribuit, diferitele componente ale sistemului fiind plasate pe platforme diferite. Această separare a responsabilităților între componente și plasarea lor pe diferite platforme oferă o serie de avantaje atât pentru dezvoltatori, cât și pentru utilizatori. Pentru dezvoltatori, această distribuire a responsabilităților rezultă într-o întreținere și modificare ulterioară a sistemului mult mai ușoară, diferitele componente putând să fie schimbate fără a influența funcționarea sistemului în ansamblu. Având finalizat serverul

web plasat pe Google App Engine acesta poate răspunde la cereri din partea oricăror clienți.

4.2. Fundamentare teoretică

4.2.1. Cloud computing

Împreună cu o creștere explozivă a aplicațiilor mobile și a conceptului de cloud computing în curs de dezvoltare, mobile cloud computing (MCC) a fost introdus ca o potențială tehnologie pentru servicii destinate utilizatorilor de dispozitive mobile. Dispozitivele mobile (smartphone-uri, tablete etc) devin din ce în ce mai mult o parte esențială a vieții unei persoane, ca urmare a faptului că cele mai eficiente și convenabile instrumente de comunicare nu sunt limitate de timp și locație. Utilizatorii acestora acumulează experiență bogată în diverse servicii și aplicații mobile, care rulează pe dispozitive și/sau pe servere aflate la distanță, prin intermediul rețelelor wireless.

Cloud Computing a fost recunoscut la nivel global ca și noua generație a infrastructurii sistemelor de calcul, ce oferă o gamă largă de avantaje, permițându-le totodată utilizatorilor să folosească infrastructura (servere, rețele, dispozitive de stocare), platformele (servicii middleware, sisteme de operare) și software-ul puse la dispoziție de către așa-numiții cloud providers (Google, Amazon, Salesforce). Ca urmare, aplicațiile mobile pot fi dezvoltate și lansate rapid, cu eforturi de management sau interacțiuni cu furnizorul de servicii considerabil diminuate.

Aşadar, termenul "cloud computing" defineşte un concept IT dezvoltat în ultimii ani. Mai clar, cloud computing se referă la un serviciu de închiriere a unor resurse virtuale hardware şi software. Prin acest serviciu, clientul nu va obţine fizic serverele pe care urmează să fie instalate anumite aplicaţii software, ci nişte capacităţi virtuale de procesare şi stocare pe care le poate accesa online. Astfel de servicii se pot închiria în funcţie de capacitatea de calcul (milioane de operaţii pe secundă), de capacitatea de stocare pusă la dispoziţie (GB) şi banda de transfer utilizată. Pe un eventual client nu îl vor interesa niciodată aspectele legate de locaţia fizică a resurselor şi de întreţinere a acestora.

Serviciile de cloud computing se încadrează în 3 categorii [6]:

1. **Software as a Service – SaaS (Exemplu: Microsoft pentru Saas)** – clientul poate utiliza aplicaţiile software puse la dispoziţie de furnizor pe o infrastructură de tip "cloud" – este cazul furnizării de servicii de găzduire web, servicii email, etc. Clientul nu poate configura parametrii infrastructurii utilizate (bandă de transfer, servere, sisteme de operare, spaţiu de stocare).
2. **Platform as a Service – PaaS (Exemplu: Google App Engine)** – clientul poate instala şi configura pe infrastructura "cloud" furnizată aplicaţiile software proprii.
3. **Infrastructure as a Service – IaaS (Exemplu: Amazon)** – clientul are posibilitatea să acceseze şi să configureze resursele de calcul puse la dispoziţie de infrastructura cloud conform necesităţilor. Poate instala orice tip de software, inclusiv sisteme de operare. De asemenea, poate configura în anumite limite resursele de reţea alocate – firewall-uri, filtre spam, etc.

Cloud Computing oferă o serie de avantaje și beneficii, precum: costuri scăzute; performanță; sincronizarea datelor utilizatorului care folosește mai multe dispozitive legate de cloud (de exemplu un smartphone, o tabletă, un notebook, dar și un PC) este

simplificată; siguranța datelor; securitatea informației; viteză de calcul și capacitate de stocare sporite, dar fără investiții în propria configurație; documentele online din cloud se pot prelucra cu ajutorul unor aplicații web.

Ca orice nouă tehnologie, cloud computing are și câteva dezavantaje: securitatea necesară a datelor din cloud poate prezenta probleme și poate produce neîncrederea utilizatorilor; e necesară o conexiune internet rapidă și stabilă; situația legală este de obicei complexă, deoarece utilizatorul nu află nici măcar în ce țară sau în ce țări se află serverele care îi găzduiesc datele sale.

4.2.2. Google Cloud Platfom

Google Cloud Platform este o platformă de cloud computing produsă de Google, care permite crearea de aplicații web, de la website-uri simple la aplicații complexe. Este compusă dintr-o familie de produse, fiecare incluzând o interfață web, un instrument de linie de comandă şi un REST API. Aceste produse sunt, după cum urmează:

- **Google App Engine** este o Platform as a Service pentru aplicațiile web de tip sandbox. App Engine oferă o scalare automată cu resurse care cresc în mod automat pentru a face față încărcării serverului.
- **Google Compute Engine** este o Infrastructure asa a Service, componentă a Google Cloud Platform, care permite utilizatorilor să lanseze mașini vurtuale (VMs) la cerere.
- **Google Cloud Storage** este un depozit online pentru fișiere.
- **Google Cloud Datastore** este un depozit de date NoSQL foarte reușit pentru date ne-relaționale care includ un REST API.
- **Google Cloud SQL** este o bază de date MySQL, componentă a structurii Google Cloud.
- **Google BigQuery** este un instrument de analizăde date care utilizează interogări SQL pentru a procesa seturi mari de date în câteva secunde.
- **Google Cloud Endpoints** este un instrument pentru a crea servicii în interiorul App Engine, care pot fi conectate cu ușurință la clienți iOS, Android și JavaScript.
- **Google Cloud DNS** este un serviciu DNS găzduit în infrastructura Google Cloud.

În **Figura 4.3** este prezentată arhitectura conceptuală a unei componenete server destinate unei aplicații mobile utilizând produse dezvoltate de către Google în cadrul platformei Google Cloud Platform.

4.2.2.1. Google AppEngine

Google App Engine (adesea denumite GAE sau pur și simplu App Engine) este o *Platform as a Service* (PaaS), o platformă cloud computing pentru dezvoltarea și stocarea de aplicații web în centrele de date Google gestionate. Cererile sunt de tip sandbox, ce rulează pe mai multe servere. App Engine poate opri și porni serverele în funcție de cerințele de trafic. O aplicație poate accesa alte calculatoare de pe internet doar prin intermediul unui URL furnizat, iar ea poate fii accesată din exterior doar prin cereri de tip HTTP sau HTTPS pe porturile standard. Aplicațiile nu pot scrie în sistemul de fișiere, și pot citi doar fișierele care cuprind codul aplicației. Codul aplicației rulează doar ca

răspuns la o cerere de tip web, o sarcină din coada de aşteptare, sau o activitate programată, şi trebuie să răspundă în maxim 60 de secunde.

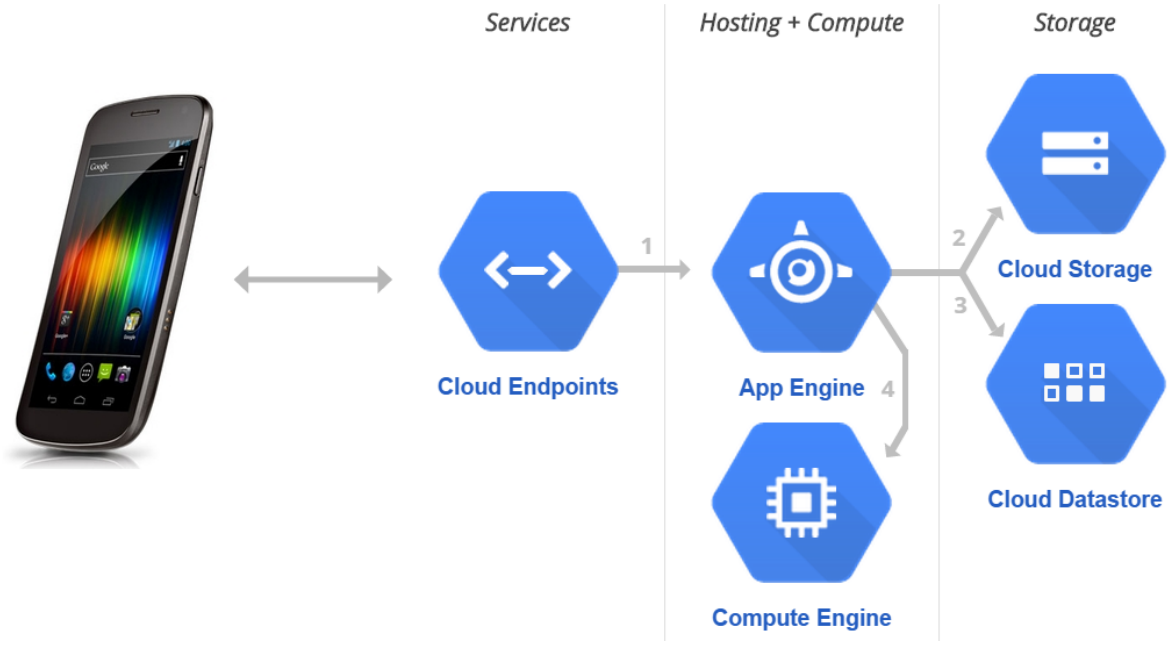


Figura 4.3 Arhitectura unui componente server pentru o aplicație mobilă folosind Google Cloud Platform [12]

App Engine oferă scalarea automată pentru aplicaţii web: odată ce numărul cererilor creşte pentru o aplicație, App Engine alocă automat mai multe resurse pentru aplicația web, pentru a putea gestiona cererile suplimentare.

În prezent, sunt suportate limbaje de programare Python, Java (şi prin extensie alte limbaje JVM ca și Groovy, JRuby, Scala, Clojure), GO şi PHP. Go şi PHP sunt în stare experimentală. Însă pe viitor se intenționează sprijinirea mai multor limbaje.

App Engine pune la dispoziția programatorului două servicii de stocare a datelor :

1. **Cloud SQL** - pentru baze de date relaționale, în special MySQL. Spațiul de stocare este disponibil din Europa sau SUA fiind de 100 GB spațiu de stocare efectiv și 16GB RAM
2. **Cloud DataStore** - destinat bazelor de date ne-relaționale folosind NoSQL. În modul gratuit permite 50 K per citiri/scrieri, 200 indecși, 1 GB date stocate timp de o lună.

4.2.2.2. Google Cloud Endpoints

Google Cloud Endpoints constă în instrumente, biblioteci şi capabilităţi care permit generarea de biblioteci API și biblioteci ale clientului la o cerere de App Engine, menţionată ca un backend API, pentru a simplifica accesul clientului la date din alte aplicaţii. Endpoints-urile fac mai uşoară crearea unui backend web pentru clienții web și clienții mobili, precum Android sau Apple iOS.

Pentru programatorii de mobile, endpoints-urile oferă un mod simplu de dezvoltare a backend-ului web partajat şi oferă, de asemenea, infrastructuri critice, cum ar fi autentificare OAuth 2.0, eliminând o mare parte din munca necesară în caz contrat. În plus, deoarece API backend este o aplicație App Engine, programatorii de mobile pot utiliza toate serviciile şi caracteristicile disponibile în App Engine, precum Datastore, Google Cloud Storage, Mail, Url Fetch, Task Queues şi aşa mai departe. Şi în cele din urmă, folosind App Engine pentru backend, dezvoltatorii sunt eliberaţi de munca de administrare a sistemului, de sarcina de echilibrare, scalarea şi întreţinerea serverului. Bibliotecile-client generate de Endpoint-uri permite efectuarea de apelurile directe Api.

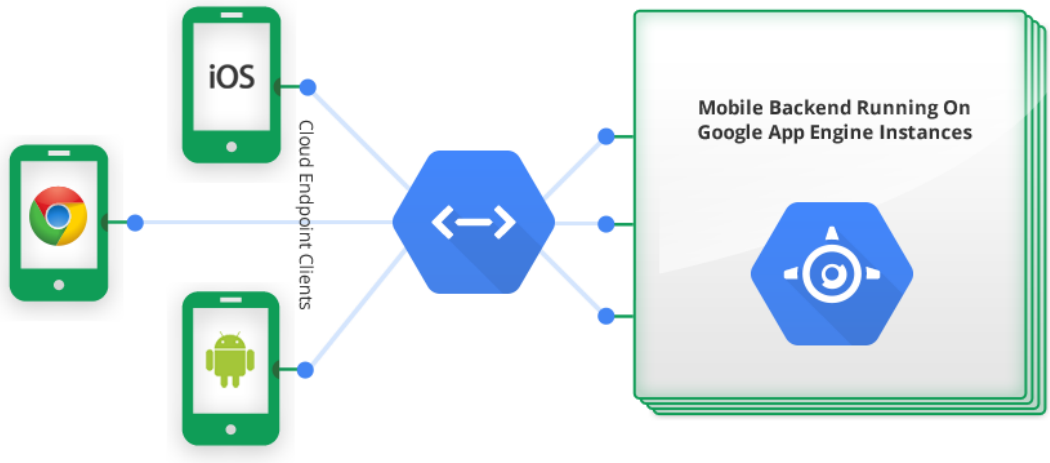


Figura 4.4 Arhitectura generală a unui sistem care utilizează Google Cloud Endpoints [13]

Aşa cum se poate observa în **Figura 4.4**, API back-end este o aplicație App Engine care realizează elemente de logică business şi alte funcţii pentru clienții Android şi iOS, precum şi clienți web folosind JavaScript. Funcţionalitatea de back-end este pusă la dispoziția clienților prin obiective ce expun un API pe care aceştia îl pot apela.

4.2.2.3. Google App Engine Datastore

App Engine Datastore este un depozit de date NoSQL care furnizează un spațiu robust și scalabil de stocare pentru aplicații web, cu următoarele caracteristici: tranzacții atomice, disponibilitate ridicată de citire și scriere, coerență puternică pentru citire și interogări, eventuală consistență pentru alte interogări.

Datastore-ul deține obiecte de date cunoscute ca entități. O entitate are una sau mai multe proprietăți, numite valori, dintre mai multe tipuri de date suportate: de exemplu, o proprietate poate fi un șir, un număr întreg sau o referință la o altă entitate. Fiecare entitate este identificată în funcție de acest gen, care clasifică entitățile pentru interogări, şi o cheie care se identifică în acest gen. Structura entităților din DataStore nu este predefinită ci se formează în funcție de codul aplicației.

Datastore poate executa mai multe operaţiuni într-o singură tranzacție. O tranzacție poate modifica doar entitățile care aparțin aceluiași grup. Entitățile aceluiași grup sunt stocate împreună pentru a eficientiza execuția tranzacțiilor. O entitate se asignează la un grup de entități la creare. Prin definiţie, o tranzacţie nu se poate face până când fiecare dintre operaţiunile sale nu au fost realizate; dacă oricare dintre operaţiuni nu reuşeşte, tranzacţia este automat refăcută. Acest lucru este util mai ales pentru aplicații web distribuite, în cazul în care mai mulţi utilizatori pot solicita accesarea sau manipularea acelorași date în acelaşi timp, asigurându-se astfel integritatea datelor. O tranzacție poate modifica doar entitățile care aparțin aceluiași grup. Entitățile aceluiași grup sunt stocate împreună pentru a eficientiza execuția tranzacțiilor. O entitate se asignează la un grup de entități la creare.

4.2.3. Android

Android este un sistem de operare pentru dispozitive și telefoane mobile, construit în jurul nucleului Linux, dezvoltat inițial de Google, iar mai apoi de Open Handset Alliance. Andoid permite programatorilor să scrie cod Java și să controleze dispozitivul prin intermediul unor biblioteci dezvoltate de Google. Sistemul de operare a fost lansat de Google sub licența Apache, o licență de tip free software și open source.

SDK-ul Android include un set complet de instrumente de dezvoltare. Acestea includ un program de depanare, biblioteci, un emulator de dispozitiv (bazat pe QEMU), documentație, mostre de cod și tutoriale. Platformele de dezvoltare sprijinite în prezent includ calculatoare bazate pe x86 care rulează Linux (orice distribuție Linux desktop modernă), Mac OS X 10.4.8 sau mai recent Windows XP sau Vista. Cerințele includ, de asemenea, Java Development Kit, Apache Ant, și Python 2.2 sau o versiune ulterioară. Mediul de dezvoltare (IDE) suportat oficial este Eclipse (3.2 sau mai recent), utilizând plug-in-ul Android Development Tools (ADT), deși dezvoltatorii pot folosi orice editor de text pentru a edita fișiere XML și Java și apoi să utilizeze unelte din linia de comandă pentru a crea, a construi și depana aplicații Android.

Android este structurat pe 4 nivele, enumerate mai jos, de la bază spre vârf:

- **Kernel**-ul Linux;
- **Librăriile software** – conţin şi modulul "runtime" ce permite rularea proceselor software pentru aplicaţiile Java. Toate procesele sunt rulate în maşini virtuale independente (Dalvik Virtual Machine) optimizate pentru dispozitivele mobile. Aplicaţiile pentru Android sunt compilate în formatul .dex – Dalvik Executable – ce este rulat de maşinile virtuale în momentul execuţiei.
- **Application framework** – cadrul de dezvoltare pentru aplicaţii – conţine clasele de obiecte necesare programatorilor în procesul de dezvoltare al aplicaţiilor cum ar fi: clase de obiecte folosite pentru afişare – views, clase de obiecte folosite pentru accesarea datelor din alte aplicaţii – content providers, etc;
- **Nivelul aplicațiilor** – aici se găsesc aplicațiile software livrate odată cu sistemul de operare: clientul email, managerul SMS, managerul de apeluri şi contacte, browser-ul web, etc. Toate aceste aplicaţii sunt scrise în Java.

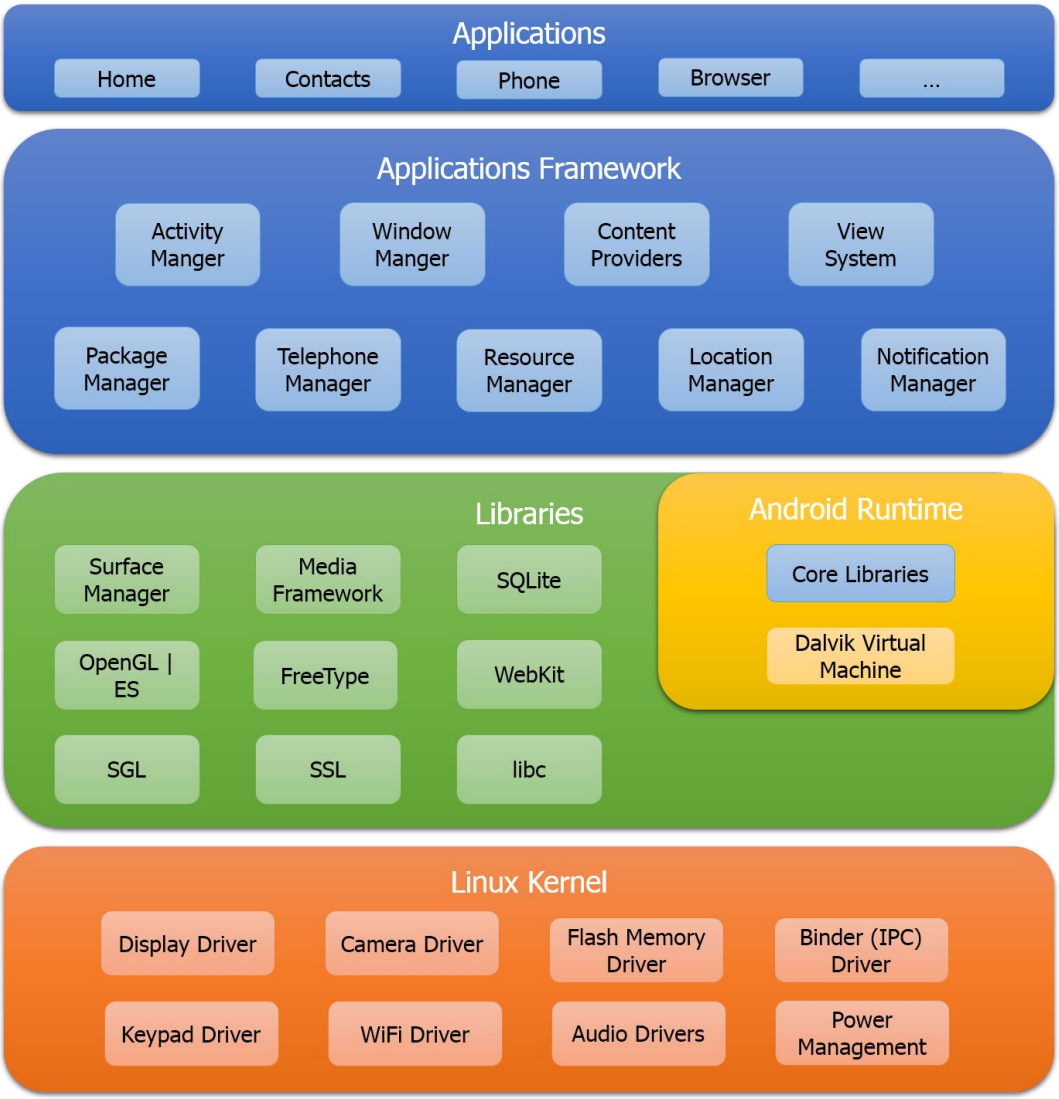


Figura 4.5 Arhitectura sistemului de operare Android

Principalele avantaje ale sistemului de operare Android sunt următoarele:

- **Utilizatorul poate alege platforma hardware pe care să ruleze Android**: sistemul Android este suportat de o sumedenie de telefoane mobile, tablete și alte dispozitive asemănătoare
- **Suportul pentru multitasking:** Android poate rula mai multe aplicații în același timp
- Pentru Android există **ROM-uri customizate**: există un nucleu de utilizatori care dezvoltă ROM-uri (imagini ale sistemului de operare Android) customizate ce pot fi instalate și rulate pe majoritatea dispozitivelor. Aceste ROM-uri aduc fie schimbări de design și funcționalitate sau chiar versiuni noi ale sistemului de operare care în mod normal nu mai sunt dezvoltate de producătorul dispozitivului
- **Andoid include integrarea cu Google** și rețelele de socializare

Printre dezavantajele care le prezintă sistemul de operare Android se numără: folosirea intensă de resurse disponibile, utilizarea mai greoaie a sistemului de către utilizatorii mai puțin inițiați în acest domeniu, infrastructură de testare inadecvată.

Cu toate acestea, faptul că Android este un sistem de operare cu sursă deschisă şi licenţă gratuită, permite programatorilor să dezvolte rapid şi cu uşurinţă aplicaţii care să aducă inovaţia în utilizarea dispozitivelor mobile.

4.2.4. Java Persistence API

Java Persistence API (JPA) reprezintă o interfață standard JAVA pentru accesarea și gestionarea datelor dintr-o bază de date relațională. Standardele JAVA definesc interfețele de adnotare a obiectelor Java, regăsirea obiectelor folosind interogări și modalitatea de interacțiune cu o bază de date utilizând tranzacții. Pentru a reprezenta cât și a persista cât mai facil datele dintr-o bază de date, JPA folosește obiecte simple Java (POJO – Plain Old Java Object) și adnotări specifice acestora pentru a defini modalitatea în care clasele Java se mapează pe tabelele bazei de date. JPA este în totalitate orientat - obiect ceea ce permite salvarea directă a obiectelor POJO nefiind nevoie de modele diferite de date sau tipuri suplimentare folosite pentru persistență. O aplicație care utilizează interfața JPA poate lucra cu tipuri diferite de baze de date fără a fi nevoie utilizarea de cod specific furnizorului bazei de date. Astfel, datorită JPA, aplicația noastră este independentă de tipul bazei de date utilizate, ceea ce simplifică considerabil portarea aplicației între diferiți furnizori de baze de date.

Google App Engine oferă suport atât pentru JPA versiunea 1.0, cât și pentru noua versiune JPA 2.0, prin intermediul plugin-ului DataNucleus. DataNucleus este cea mai cunoscută și platformă open-source de acces și persistență a datelor, fiind compatibilă cu o serie largă de standarde de persistență. Cu toate acestea, există o serie de funcționalități specifice JPA pe care Google App Engine nu le suportă în momentul de față, cum ar fi interogările de tip „agregare” (group by, having, sum, etc) și „join”. Limitarea interogărirlor de tip „Join” se datorează faptului că nu se poate folosi un câmp al unei entități fiu într-o interogare asupra unei entități părinte. De asemenea, nu sunt suportate de către App Engine nici interogările polimorfe, ceea ce înseamnă că nu putem interoga cu o clasă și să primim o instanță a unei subclase. În Datastore-ul App Engine-ului fiecare clasă este reprezentată ca o entitate separată de un anumit tip. Pe lângă aceste restricții de interogări, există restricții și în ceea ce privesc relațiile dintre entități, ne fiind suportate relațiile cu posesor de tip mai mulți-la-mai mulți.

4.3. Tool-uri folosite

4.3.1. Eclipse IDE Juno

Pentru implementarea și testarea aplicației am folosit mediul de dezvoltare integrat Eclipse IDE, versiunea Juno.

Pe lângă funcționalitățile de bază specifice unui mediu de dezvoltare integrat, Eclipse oferă și un sistem de plug-in extensibil pentru personalizare. Scris în mare parte în Java, Eclipse poate fi folosit pentru a dezvolta aplicații utilizaând o gamă larga de limbaje de programare: Ada, ABAP, C, C ++, COBOL, Fortran, Haskell, JavaScript, Lasso, natural, Perl, PHP, Prolog, Python, Ruby (inclusiv Ruby on Rails), Scala, Clojure,

Groovy, Schema și Erlang. Dintre plugin-urile Eclipse utilizate în dezvoltarea sistemului xpressMenu pot aminti: Android Developer Tools (ADT) și Google Plugin for Eclipse.

Pluginul ADT, instrumentul de dezvoltare Android, este un plugin care are ca scop ofeririea unui mediu integrat care ajută la construirea aplicațiilor Android. ADT extinde capacitățile Eclipse, pentru a permite dezvoltatorilor să realizeze noi proiecte Android, să creeze o interfață pentru propria aplicație, să adauge pachete bazate pe API Android, să realizeze depanarea aplicațiilor lor, folosind instrumentele SDK Android și să-și distribuie aplicațiile pe dispozitive compatibile Android, cu ajutorul fișierelor .apk exportate.

Google Plugin for Eclipse, este un set de instrumente de dezvoltare software destinat dezvoltatorilor Java pentru a proiecta rapid, construi, optimiza și implementa aplicații bazate pe soluția de cloud de la Google, App Engine. De asemenea, plugin-ul ajută dezvoltatorii la crearea în mod eficient a unei interfețe de utilizare prietenoase și interactive, prin generarea de cod Ajax folosind Google Web Toolkit și distribuirea fără efort a aplicației în App Engine.

4.3.2. MySQL Workbench

Odată cu extinderea volumului de date, a cloud computingului și a utilizării de mobile computing, au crescut și provocările de management pentru profesioniștii din domeniul bazelor de date. Pentru a ajuta dezvoltatorii și administratorii să gestioneze mai bine aceste medii de date dinamice, MySQL Workbench oferă ușurință în utilizare și permite utilizatorilor să dezvolte baze de date, să le proiecteze și să le administreze.

Cu alte cuvinte, MySQL Workbench este un instrument de proiectare a bazei de date care integrează dezvoltarea SQL, administrarea, proiectarea bazei de date, crearea și întreținerea, într-un singur mediu de dezvoltare integrat.

În plus, MySQL Workbench simplifică proiectarea bazelor de date și întreținerea lor și îmbunătățește comunicarea între echipele DBA și dezvoltator. Aceasta permite dezvoltatorilor să vizualizeze cerințele, să comunice cu părțile interesate, și să rezolve problemele de design, înainte de o investiție majoră de timp și resurse.

4.3.3. Adobe Photoshop

Pentru realizarea interfeței grafice a aplicației cu utilizatorul am folosit editorul Adobe Photoshop. Adobe Photoshop este un software folosit pentru editarea imaginilor digitale pe calculator, program produs și distribuit de compania americană Adobe Systems și care se se adresează în special profesioniștilor domeniului.

Photoshop este un program cu o interfață intuitivă și care permite o multitudine extraordinară de modificări necesare în mod curent profesioniștilor și nu numai: editări de luminozitate și contrast, culoare, focalizare, aplicare de efecte pe imagine sau pe zone (selecții), retușare de imagini degradate, număr arbitrar de canale de culoare, suport de canale de culoare pe 8, 16 sau 32 biți, efecte third-party etc.

Photoshop poate citi majoritatea fișierelor raster și vector. De asemenea, are o serie de formate proprii: PSD (abreviere pentru Photoshop Document) - acesta este un format popular și des răspândit în rândul profesioniștilor, astfel că este compatibil și cu unele aplicații concurente Photoshop; PSB (denumit Large Document Format) este o versiune mai nouă a formatului PSD, conceput special pentru fișiere mai mari (2GB); PDD este un format mai puțin întâlnit, fiind asociat inițial aplicației Adobe PhotoDeluxe.

Principalele elemente prin care Photshop se diferențiază de aplicațiile concurente și prin care stabilește noi standarde în industria prelucrării de imagini digitale sunt: selecțiile, straturile (Layers), măștile (Masks), canalele (Channels), retușarea, optimizarea imaginilor pentru Web.

4.3.4. Gliffy

Gliffy este o aplicație web online care poate fi folosită pentru a crea orice, de la diagrame, machete de rețea și Analiză SWOT la planurile de podea. Partajarea și colaborarea sunt ușor de făcut cu Gliffy, făcându-l un instrument puternic pentru grupurile utilizatorilor de Internet. Și, pentru că serviciul este web-based, utilizatorii îl pot accesa de pe diferite sisteme de operare și pot folosi diferite browsere fără probleme de incompatibilitate (deși este necesar Flash). Gliffy are o mare acoperire în rândul studenților, permițându-le să practice crearea de reprezentări vizuale de date.

Gliffy este singurul software grafic de creare a diagramelor în mediul online, dar cu toate acestea are unele avantaje: este ușor de utilizat și vine cu o varietate de template-uri. Este mult mai accesibil decât altele în domeniul său, în timp ce oferă și o funcționalitate completă. În plus, ușurința de colaborare on-line face Gliffy un program ideal pentru utilizarea lui în setările de grup.

Capitolul 5. Proiectare de Detaliu și Implementare

Aceste capitol începe prin a prezenta principalele cazuri de utilizare ale sistemului xpressMenu în funcție de cerințele funcționale și actorii sistemului. De asemenea, prezintă structura generală a sistemului prin evidențierea principaleleor componente care intră în alcătuirea sistemului informatic, dar și detalii de implementare. Vor fi prezentate structura bazei de date, diagramele de clase împreună cu o descriere a claselor și a metodelor mai importante.

5.1. Cazuri de utilizare

Una dintre cele mai frecvente provocări cu care se confruntă echipele de dezvoltare software sunt cerințele inexacte, incomplete sau inconsistente. Acest lucru afectează atât latura IT responsabilă de dezvoltarea produsului software cât și latura de business - putând duce la furnizarea unor sisteme care nu satisfac nevoile reale de business, și nu aduc valoarea așteptată clientului. Realizarea așa numitor cazuri de utilizare oferă o soluție excelentă pentru această provocare, soluție care sa dovedit a fi una dintre cele mai adoptate practici care să ajute dezvoltatorii să înțeleagă funcționalitatea sistemului din perspectiva utilizatorilor și astfel să stabilească cât mai exact cerințele unui sistem software.

Un caz de utilizare este o secvență de acțiuni, efectuate de către unul sau mai mulți actori (persoane sau entități non-umane din afara sistemului) și de sistemul în sine, care produce unul sau mai multe rezultate observabil pentru unul dintre actori. [7]

Cazurile de utilizare sunt scrise într-un limbaj comun astfel încât să fie ușor de înțeles de către toți stakeholderi (utilizatori, programatori, arhitecți, testeri, manageri, etc) și pentru acțiunea care face obiectului unui caz de utilizare trebuie să analizeze toate scenariile posibile, atât de succes cât și de eșec.

Pentru aplicația xpressMenu vom prezenta în continuare actorii sistemului și cazurile de utilizare atât sub formă de diagramă UML cât și sub formă scrisă.

5.1.1. Actorii sistemului

Un caz de utilizare trebuie să fie inițiat de către cineva sau ceva din afara domeniului de aplicare al cazului de utilizare: un actor. Un actor nu e nevoie să fie un utilizator uman; orice sistem extern sau o entitate externa cazului de utilizare poate declanșa cazul utilizării (sau să fie beneficiarul rezultatelor cazului de utilizare) și ar trebui să fie modelat ca un actor. [8]

Sistemul informatic xpressMenu are 3 actori principali:

- Clientul restaurantului – este entitatea care utilizează aplicația mobilă dedicată sistemelor de operare Android, prin intermediul căreia odată ajuns într-un restaurant poate scana codul QR aflat pe masă, să primească instant meniul resturantului și să trimită o comandă.
- Ospătarul – este entitatea care utilizează aplicația web destinată personalului angajat al restaurantului, prin intermediul căreia acesta poate vizualiza și gestiona comenzile venite din partea clienților din restaurant.

- Administratorul restaurantului – este entitatea cu drepturi supreme asupra aplicației web dedicate restaurantelor, putând gestiona întreaga activitate a restaurantului, meniurile și personalul din această aplicație.

5.1.2. Cazuri de utilizare

În descrierea unui caz de utilizare intră mai multe elemente printre care: numele cazului, actorul principal care inițiază cazul de utilizare, o scurtă descriere, alți participanți, precondiții, postcondiții, principalul scenariu de succes, extensii.

În continuarea acestui subcapitol vor fi prezentate principalele cazuri de utilizare, atât în scris cât și sub forma de diagramă UML, pentru cei 3 actori ai sistemului.

5.1.2.1. Utilizator: Clientul restaurantului

Principalele cazuri de utilizare inițiate de către *Clientul restaurantului* sunt prezentate în **Figura 5.1**, iar două dintre acestea sunt detaliate și în scris mai jos.

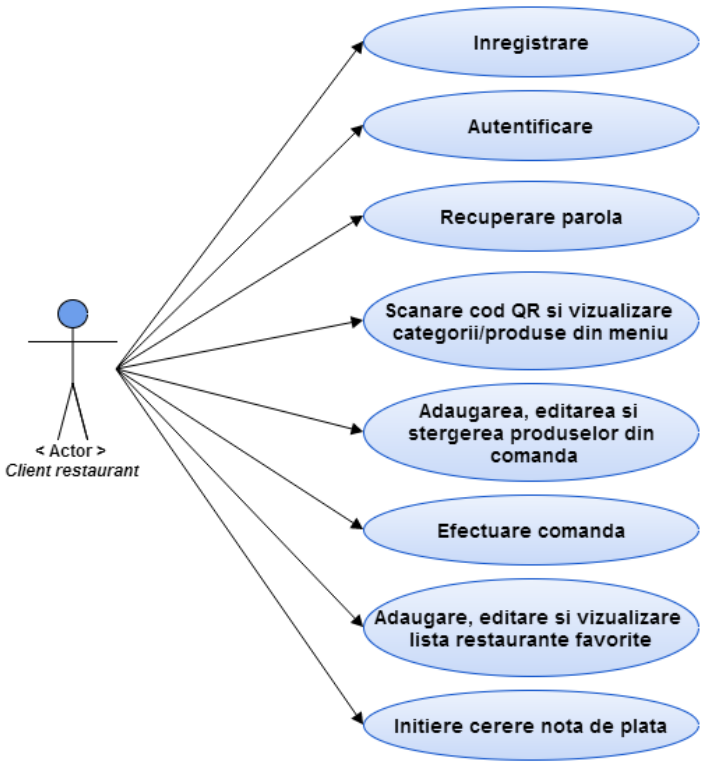


Figura 5.1 Diagrama cazurilor de utilizare pentru "Client restaurant"

Caz de utilizare: Înregistrare utilizator aplicație mobile (Client restaurant)

Actor principal: Utilizator aplicație mobile (Client restaurant)
Participanți și interese: Utilizatorul aplicației mobile: dorește să se înregistreze în cadrul aplicației pentru a putea utiliza funcționalitățile aplicației.
Precondiții: Aplicația trebuie să fie instalată și pornită.
Post condiții: Utilizatorul este înregistrat în aplicație.

Principalul scenariu de succes (Flux de Bază):
1. Utilizatorul accesează secțiunea de înregistrare.
2. Utilizatorul completează datele persoale, împreună cu numele de utilizator și parola dorită.
3. Utilizatorul trimite datele completate pentru a fi salvate de către sistem.
4. Noul cont este salvat cu succes de către sistem.
5. Utilizatorul primește un mesaj de înregistrare cu succes. Utilizatorul are acum posibilitatea de a se autentifica în aplicație.
Extensii (Fluxuri Alternative):
4a. Sistemul detectează că nu au fost completate toate câmpurile obligatorii din formular:
1. Sistemul oprește procesul de înregistrare și trimite utilizatorului un mesaj corespunzător.
4b. Emailul introdus este invalid:
1. Sistemul oprește procesul de înregistrare și trimite utilizatorului un mesaj de tipul "Emailul introdus este invalid!".
4c. Emailul sau numele de utilizator introduce se află deja în sistem:
1. Sistemul informează utilizatorul de existent în sistem a email-ului sau a numelui de utilizator introdus.
4d. Parolă introdusă în formularul de înregistrare nu este destul de puternică (lungimea caracterelor >= 8, să conțină cel puțin o cifră, un character special și o literă mare):
1. Sistemul oprește procesul de înregistrare și informează utilizatorul.

Caz de utilizare: Efectuare comandă

Actor principal: Utilizator aplicație mobilă (Client restaurant)
Participanți și interese: Utilizatorul: dorește să vizualizeze meniul restaurantului și să efectueze o comandă.
Ospătarul: dorește să vizualizeze cât mai repede comanda din partea clientului pentru a o onora în cel mai scurt timp.
Precondiții: Aplicația trebuie să fie instalată și pornită.
Utilizatorul trebuie să fie autentificat în aplicație.
Post condiții: Comanda realizată de către utilizator este vizualizată de către ospătar.
Principalul scenariu de succes (Flux de Bază):
1. Utilizatorul scanează codul QR aflat pe masa restaurantului și apoi poate vizualiza meniul restaurantului pe aplicație.
2. Utilizatorul alege una dintre categoriile de produse pe care restaurantul le oferă.
3. Utilizatorul vizualizează lista de produse din categoria aleasă și selectează unul dintre produsele oferite.
4. Aplicația prezintă informații detaliate despre produs: imagine, denumire, cantitate, ingredientele din care se compune produsul și prețul acestuia.
5. Utilizatorul selectează numărul de produse și dacă dorește specifică informații speciale pe care ospătarul le poate vizualiza odata cu primirea comenzii.
6. Utilizatorul adaugă cu succes produsul selectat în coșul de cumpărături.
7. Utilizatorul parcurge din nou pașii 2-6 dacă dorește să selecteze altă categorie de produse, pașii 3-6 dacă dorește să introducă în comandă un produs din categoria actuală, pașii 4-5 dacă dorește să modifice numărul de bucăți al produsului adăugat anterior.

Această secvență de pași este efectuată până când utilizatorul are lista finală a produselor pe care dorește să le comande.
8. Utilizatorul coșul de cumpărături: fiecare produs adăugat împreună cu numărul de bucăți, prețul fiecărui produs și suma totală de plată.
11. Utilizatorul trimite comanda către Ospătar. Sistemul adaugă comanda în baza de date și notifică ospătarul restaurantului cu privire la datele comenzii. Toate produsele din coș sunt eliminate.
12. Utilizatorul este notificat în momentul în care comanda este preluată.
Extensii (Fluxuri Alternative):
1a. Utilizatorul dorește să vizualizeze locația restaurantului.
1. Sistemul localizează restaurantul pe hartă.
2-8a. Utilizatorul dorește să modifice coșul de produse:
1. Utilizatorul accesează coșul de produse, unde are posibilitatea să modifice numărul de bucăți al fiecărui produs sau să îl steargă.

5.1.2.2. Utilizator: Ospătarul

Cazurile de utilizare inițiate de către *Ospătar*-ul restaurantului sunt prezentate în **Figura 5.2**, iar unul dintre acestea este detaliate în scris mai jos.

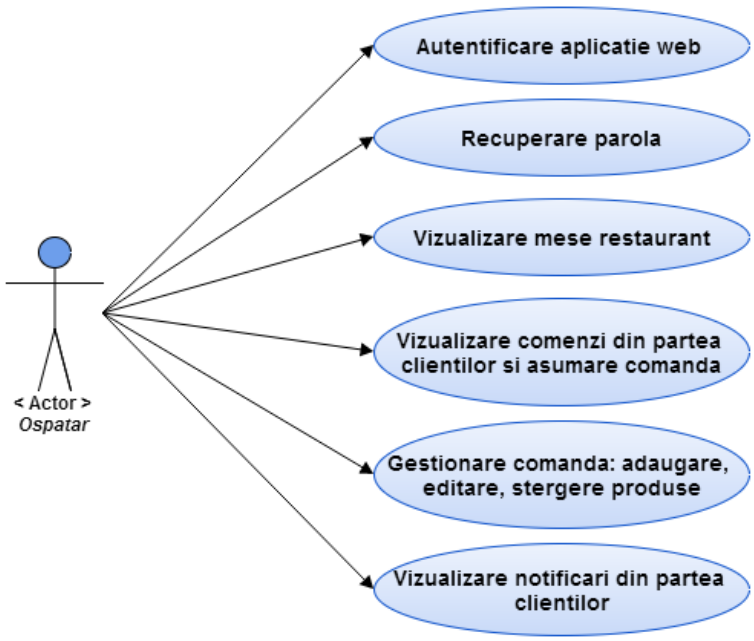


Figura 5.2 Diagrama cazurilor de utilizare pentru "Ospatar"

Caz de utilizare: Autentificare utilizator pe platform web (Ospătar)

Actor principal: Ospătar
Participanți și interese: Ospătarul: dorește să se autentifice în sistem pentru a putea vizualiza și gestiona comenzile venite din partea clienților.
Precondiții: Aplicația trebuie să fie instalată și pornită.

Ospătarul trebuie să dețină un cont de utilizator valid creeat de Administratorul restaurantului.
Post condiții: Ospătarul este autentificat în sistem.
Principalul scenariu de succes (Flux de Bază):
1. Ospătarul introduce numele de utilizator și parola în aplicația web.
2. Ospătarul trimite datele completate pentru a fi validate de către sistem.
3. Sistemul validează datele trimise de către ospătar.
4. Sistemul redirecționează ospătarul către interfața principal a aplicației web.
Extensii (Fluxuri Alternative):
1a. Ospătarul este deja autentificat în aplicație la deschiderea acesteia:
    1. Se trece la efectuarea pasului 4 din fluxul principal.
3a. Nu au fost furnizate unul dintre: numele de utilizator sau parola:
    1. Ospătarul nu este autentificat și primește un mesaj corespunzător.
3c. Numele de utilizator sau parolă nu se potrivesc cu baza de date:
    1. Ospătarul nu este autentificat și primește un mesaj de eroare de la sistem.

5.1.2.3. Utilizator: Administratorul restaurantului

Principalele cazuri de utilizare inițiate de către *Administratorul restaurantului* sunt prezentate în **Figura 5.3**, iar unul dintre acestea este detaliate și în scris mai jos.

Caz de utilizare: Adaugare ospătar

Actor principal: Administrator restaurant
Participanți și interese: Aministratorul: dorește să introducă un nou ospătar în sistem
Precondiții: Aplicația trebuie să fie instalată și pornită.
Administratorul trebuie să dețină un cont de utilizator valid.
Post condiții: Administratorul este autentificat în sistem.
Principalul scenariu de succes (Flux de Bază):
1. Administratorul accesează secțiunea de utilizatori din cadrul aplicației web.
2. Administratorul selectează opțiunea de adaugare a unui nou ospătar.
3. Completează formularul de adăugare ospătar și trimite formularul către server.
4. Sistemul informează administratorul că cont de ospătar a fost creeat.
Extensii (Fluxuri Alternative):
3a. Sistemul detectează că nu au fost completate toate câmpurile obligatorii din formular:
    1. Sistemul oprește procesul și afisează un mesaj corespunzător.
3b. Emailul introdus este invalid:
    1. Sistemul oprește procesul și afișează un mesaj de tipul "Emailul introdus este invalid!".
3c. Emailul sau numele de ospătarului introdus se află deja în sistem:
    1. Sistemul informează utilizatorul de existent în sistem a email-ului sau a numelui de utilizator introdus.

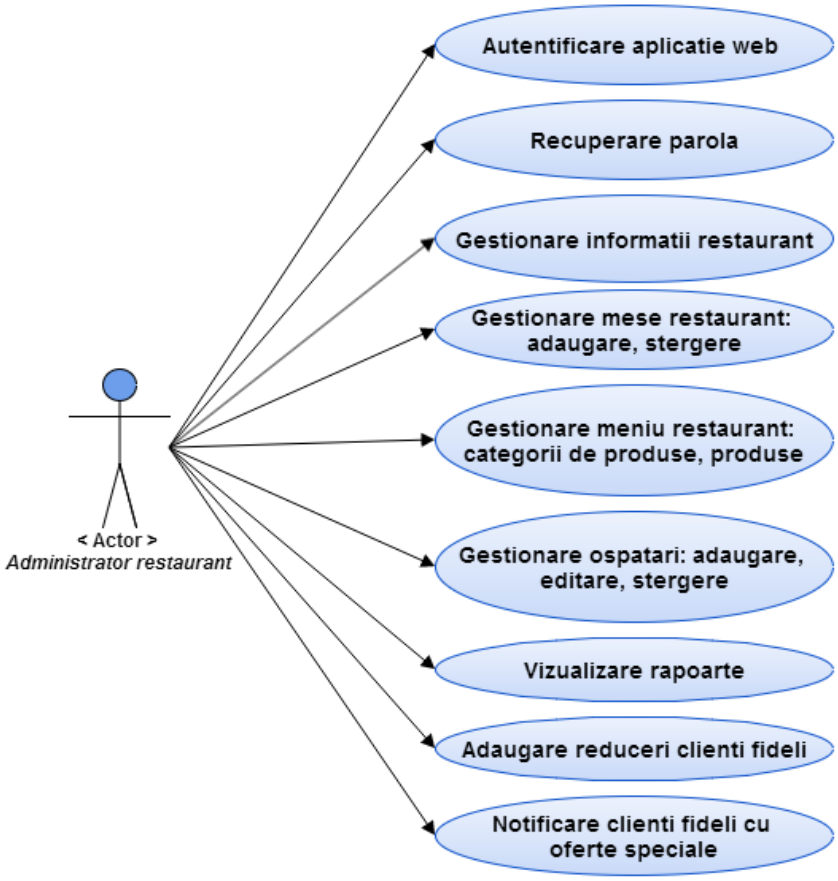


Figura 5.3 Diagrama cazurilor de utilizare pentru „Administratorul restaurantului”

5.2. Structura generală a sistemului

Sistemul xpressMenu a fost conceput având la bază arhitectura Client – Server și cele mai noi tehnologii web și mobile, iar structura s-a generală împreună cu principalele componente și tehnologii este prezentă în **Figura 5.4**.

Cele trei componente principale care intră în strucutra xpressMenu sunt:

1. **Componenta server** – este baza întregului sistem informatic, fiind responsabilă de a procesa și a răspunde cererilor venite din partea componentelor client prin intermediul protocolului HTTP.
2. **Componenta mobilă** – reprezintă aplicața destinată clienților restaurantelor și poate rula pe dispozitivele Android, având la bază modelul de dezvoltare Model-View-Controller.
3. **Componenta web** – este aplicația destinată ospătarilor și administratorilor de restaurante, prin intermediul căreia aceștia pot gestiona activitatea din restaurant. Este implementată sub forma unei aplicații web independente de sistemul de operare și de navigatorul de internet utilizate, iar la baza s-a stau tehnologii ca HTML, CSS și JavaScript, împreună cu diferite framework-uri de dezvoltare web.

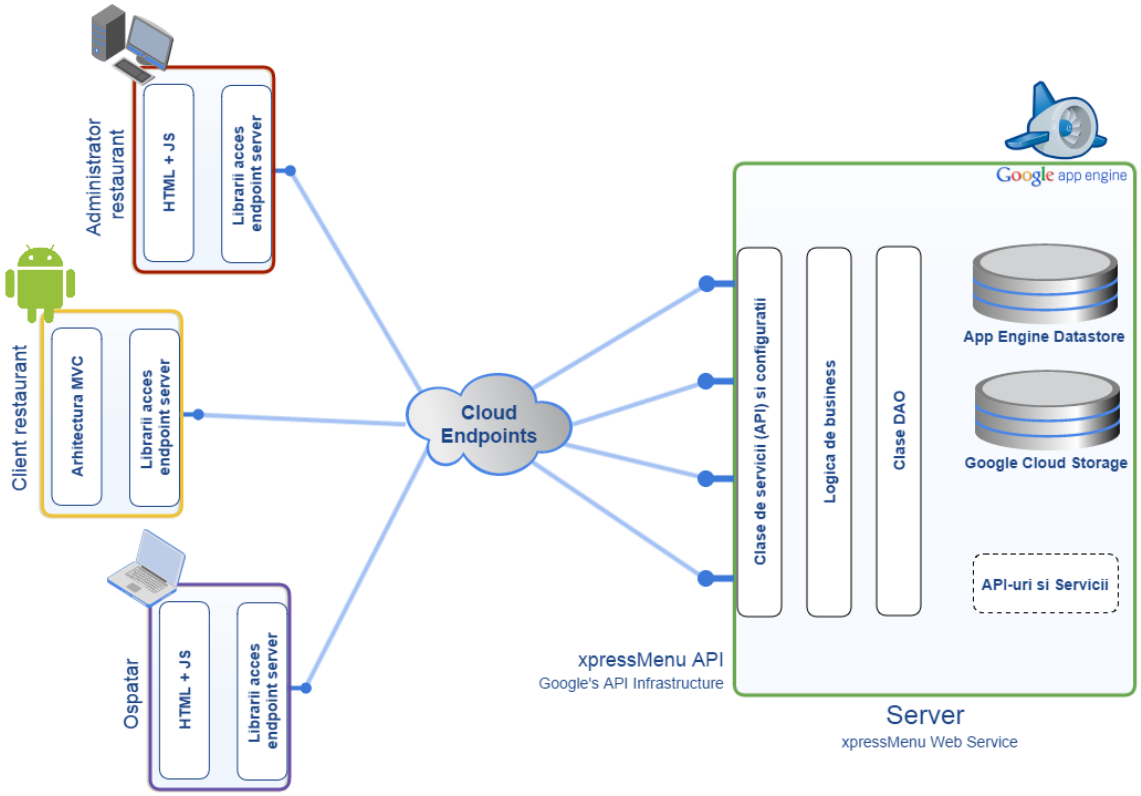


Figura 5.4 Structura generală a sistemului xpressMenu

În **Figura 5.4** se poate observa *Componenta Server* (în partea dreaptă a figurii) care este dezvoltată și rulează pe platforma Google App Engine. Pe lângă clasele de servicii web, logica de business și de acces la date, se mai poate observa că Google App Engine este o soluție integrată care oferă și alte facilități, cum ar fi: acces la bază de date(App Engine Datastore), stocare de fișiere (Google Cloud Storage), dar și posibilitate de utilizare a unui număr mare de API-uri sau servicii web furnizate de diferiți dezvoltatori sub o licență deschisă. Tot în **Figura 5.4**, în partea stângă, se pot observa celelalte două componente ale sistemului în funcție de actorii sistemului. În consecința, avem componenta mobile utilizată de clientul restaurantului și două instanțe ale componentei web – pentru ospătar și pentru administratorul de restaurant.

Comunicarea între componentele sistemului se face prin intermediul unei interfete de acces(engl. API) publice. În momentul de față API-urile sunt considerate ca fiind cel mai eficient mecanism de integrare a două sisteme, mai ales dacă discutăm despre sisteme incompatibile din punct de vedere al tehnologiilor implementate, acesta fiind și unul dintre principalele motive pentru care a fost aleasă această abordare. Un alt argument care recomandă utilizarea API-urilor în integrarea unor sisteme este flexibilitatea. În general API-urile sunt rulate de către o aplicație de pe partea de server, iar serviciile lor pot fi utilizate de o gamă largă de clienți: aplicații server-side, aplicații mobile native sau aplicații web care rulează într-un navigator web (pe un sistem desktop sau mobile). Flexibilitatea este dată de dependența foarte scăzută față de tehnologiile utilizate, serviciul web putând fi implementat de exemplu in Java, iar serviciile lui expusă printr-un API public pot fi accesate dintr-un client scris în Java, o aplicație client scrisă pentru Android/iOS sau de client web scris în JavaScript. Această flexibilitate este oferită

de o serie de protocoale și tehnologii standardizate datorită cărora conceptul de API a putut lua naștere. Arhitectura pe care se bazează aceste concept este arhitectura REST, datorită simplității acesteia, iar protocol de comunicare utilizat este HTTP. Cererile între principalele componente ale sistemului vor fi lansate utilizând metodele HTTP GET, POST, PUT și DELETE, conținutul acestor cereri și răspunsurile utilizând formatul JSON. Un alt criteriu pentru care a fost aleasă arhitectura REST îl reprezintă proprietatea sa de „stateless", orice cerere HTTP întâmplându-se într-o izolare completă. Serverul nu trebuie să cunoască locația clientului sau alte informații despre starea clientului, deoarece nu se bazează niciodată pe cereri precedente sau pe cereri din partea altor clienți, toate informațiile de care are nevoie fiind trimise cu fiecare cerere în parte. Această izolare a cererilor între ele, oferă posibilitatea ca acestea să fie tratate de servere diferite fără a fi nevoie să comunicare între ele, aspect care îmbunătățește considerabil gradul de scalabilitate al aplicației.

În continuarea acestui capitol sunt prezentate fiecare dintre componentele care intră în alcătuirea sistemului xpressMenu, împreună cu diagrame specifice și detalii de implementare.

5.3. Proiectarea bazei de date

După cum am prezentat în Capitolul 4, datele utilizate de sistem sunt salvate folosind Google App Engine Datastore, o facilitate oferită de Google Cloud Platform.

Baza de date a sistemului informatic xpressMenu are în componență 17 tabele. În următoarea listă se pot regăsi cele mai importante tabele și o scurtă descriere a acestora:

- User – tabela în care se salvează utilizatorii sistemului
- Restaurant – tabela pentru salvarea informațiilor despre restaurantele care utilizează sistemul xpressMenu
- Order – tabela pentru salvarea comenzilor
- Menu – tabela pentru salvarea meniurilor unui restaurant
- MenuItem – tabela pentru salvarea produselor care intră în componența meniului
- TableAllocation – tabela care salvează informații despre mesele unui restaurant

Diagrama bazei de date este prezentată în **Figura 5.5** unde se pot observa toate tabele care intră în componența bazei de date, câmpurile acestora și relațiile dintre ele.

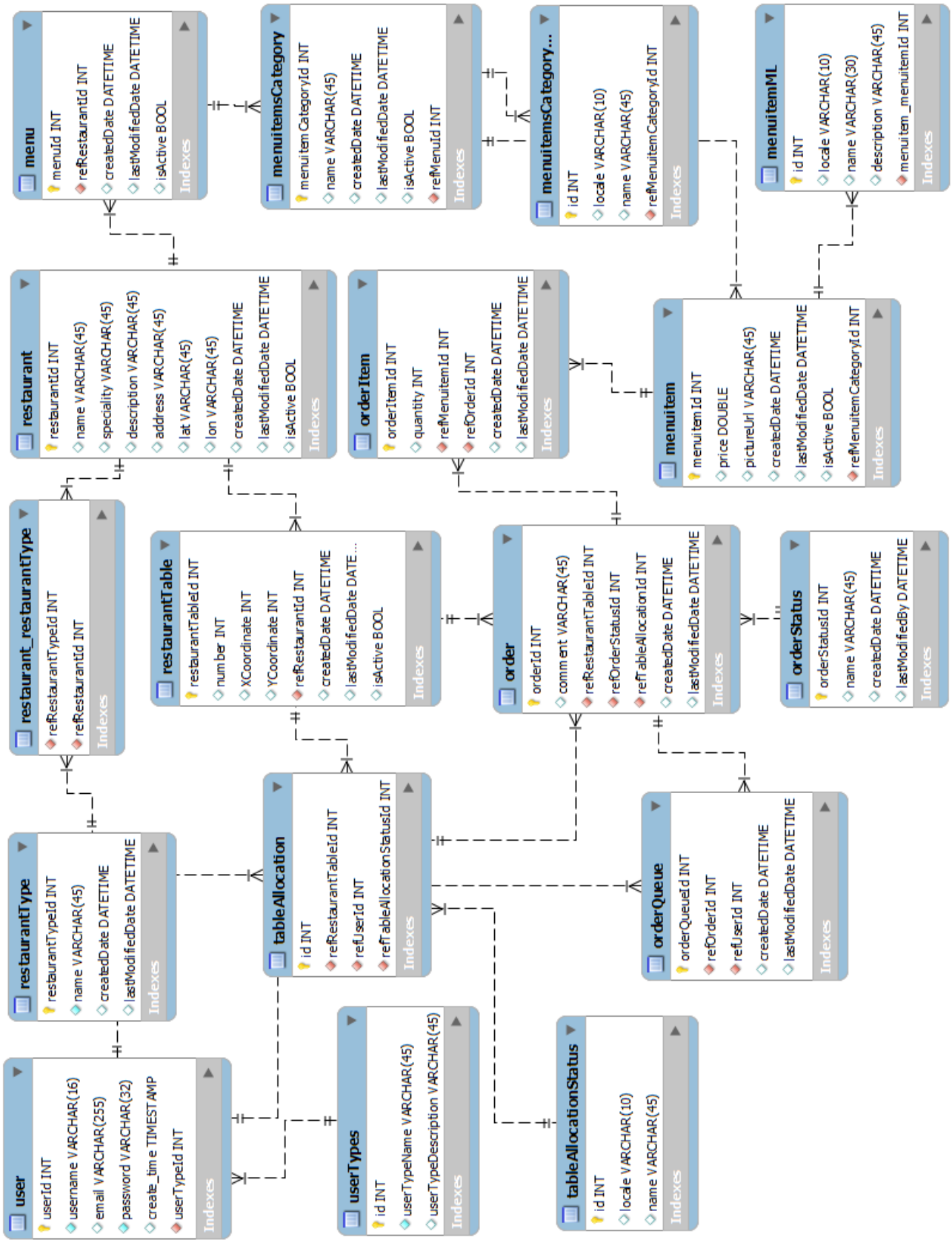


Figura 5.5 Structura bazei de date

5.4. Componenta server

Componenta server a sistemului xpressMenu care rulează pe soluția cloud Google App Engine este punctul central al sistemului, oferind clienților prin intermediul unui API public servicii care permit realizarea operațiiloor de salvare sau interogare a datelor salvate în baza de date Google DataStore. Pe lângă serviciile web puse la dispoziția aplicației mobile xpressMenu, componenta server este o interfața și pentru componenta web utilizată de către ospătari și adminstratorii de restaurante.

Prin utilizarea tehnologiilor oferite de platforma Google Cloud Platform, mediul de dezvoltare și distribuite Google App Engine, soluția pentru salvarea datelor – Google App Engine DataStore și soluția pentru dezvoltarea API-urilor – Google Cloud Endpoints, se mărește considerabil gradul de performanța și scalabilitate al sistemului xpressMenu comparativ cu alte soluții similare.

În spatele Google Cloud Endpoints se află arhitectura simplă dar robustă REST, care utilizează pentru comunicare protocolul HTTP și protocolul JSON pentru schimbul de informații între serviciile web și clienți. La realizarea unei cereri către server, informațiile despre metoda care se apelează sunt deduse din metoda HTTP utilizată și din URI-ul apelat, iar argumentele metodei sunt transmise în corpul cereri în format JSON. Am ales JSON pentru transmiterea informațiilor deoarece *este un format standard de interschimbare a datelor. Este ușor de citit și scrie pentru oameni. Este ușor de parsat și generat de către mașini. JSON este un format text care este complet independent de limbaj dar folosește convenții care le sunt familiare programatorilor familiei de limbaje C. Aceste proprietăți fac din JSON un limbaj ideal pentru interschimbarea datelor.* [9]

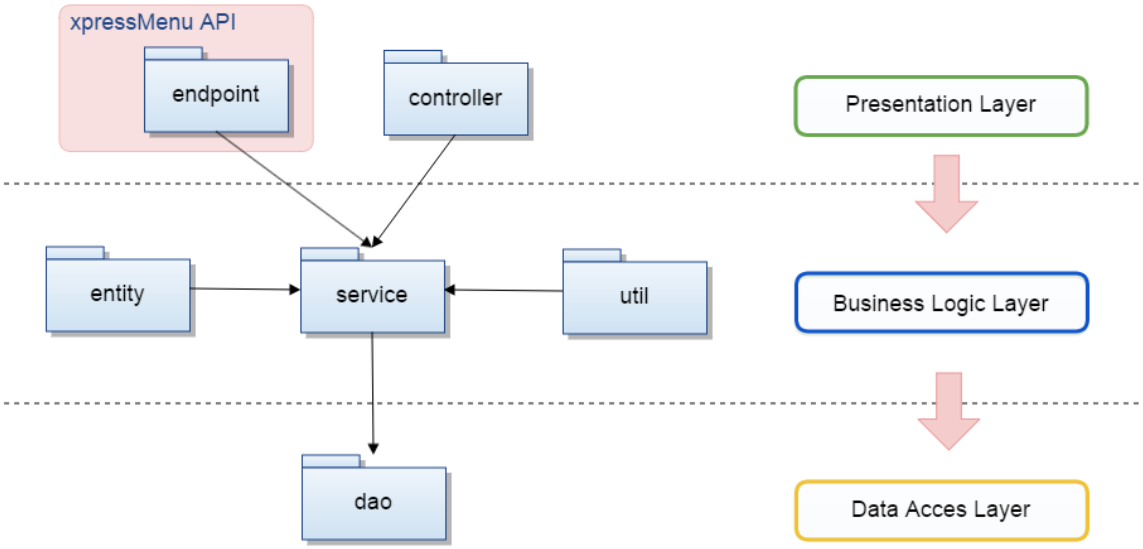


Figura 5.6 Arhitectura pe nivele a componentei server

În **Figura 5.6** este prezentată arhitectura pe nivele a componentei server a sistemului xpressMenu, componentă care rulează pe Google App Engine.

Arhitectura componentei server este structurată pe trei nivele:

- *Presentation* – reprezintă interfața atât cu utilizatorul componentei web a sistemului cât și cu aplicația xpressMenu reprezentată de componenta mobile. Acest nivel este format din pachetele controller și endpoint. Pachetul endpoint reprezintă interfața publică (xpressMenu API) prin intermediul căreia sunt expuse și utilizate serviciile componentei server de către componenta web și componenta mobile. Pachetul endpoint are în componența sa clase care se mapează pe diferitele metode oferite de API-ului xpressMenu și care sunt apelate de clienți folosind REST și JSON.
- *Business Logic* – este nivelul care înglobează întreaga logică de business a sistemului, fiind responsabil de procesarea și formarea răspunsului la cererile venite din partea aplicației Android și a celei web. Acest nivel are în componența sa următoarele pachete: entity, service și util. Responsabilitățile principale ale acestui nivel sunt modelarea entităților din cadrul sistemului și executarea proceselor de business ale sistemului.
- *Data Access* – nivelul de la baza arhitecturii componentei server este format din pachetul dao, care conține clase responsabile de accesarea și gestionarea datelor stocate în App Engine DataStore.

Dintre avantajele pe care această arhitectură le oferă sistemului enumerăm: uşurinţa modificării sau înlocuirii oricărui nivel fără a afecta funcţionalitatea celorlalte, posibilitatea testării separate a fiecărui nivel, facilitatea reutilizării codului, iar separarea oferită de aceste nivele îmbunătățește semnificativ flexibilitatea, mentenanța și scalabilitatea sistemului.

5.4.1. Modulele componentei server

1. Controller

Pachetul „Controller” face parte din nivelul de „Presentation” și conține toate controller-ele care sunt responsabile de preluarea și procesarea cererilor HTTP venite din partea clienților. Aplicația preia cererile și determină controller-ul care este responsabil de procesarea lor pe baza adonotărilor specificate pe fiecare clasă de controller. Această mapare a cererilor pe controllere se face în funcție de URL-ul sub care sunt formulate cererile.

Mai jos poate fi observat un fragment dintr-un controller, care este mapat la o cerere de adăugare a unui restaurant în baza de date. Cererea vine la următorul url: */restaurant/add.*

```
@Controller
@RequestMapping("/restaurant")
public class RestaurantController {
        @RequestMapping(value="/add", method = RequestMethod.POST)
        public ModelAndView add(HttpServletRequest request, ModelMap model)
        {
                // cod pentru procesarea cererii
        }
}
```

Se poate observa din fragmentul de cod anterior că se utilizează adnotarea @Controller pentru a specifica faptul ca clasa respectivă reprezintă un controller, iar cu ajutorul adnotărilor @RequestMapping se mapează acestă clasă și metodele sale aferente cu URL-ul pe care a venit cererea. Se mai poate observa ca adnotarea @RequestMapping specifică și metoda HTTP cu care trebuie invocată această funcție. La funcția de adăugare a unui restaurant este trimis și parametrul HttpServletRequest request care conține toate informațiile necesare realizării cu succes a cererii.

2. Endpoint

Pachetul „Endpoint”, care face parte din nivelul de „Presentation” reprezintă clasele care intră în componeța interfeței publice de tip REST (REST API), prin intermediul căreia componenta mobilă poate accesa serviciile oferite de componenta server.

Clasele din acest pachet și metodele acestora sunt mapate la posibilele cereri care pot veni din partea clienților (componenta mobilă în cazul nostru) în funcție de URL-ul cerut și metoda HTTP utilizată. Se utilizează 4 metode HTTP în funcție de tipul cererii:

- GET – pentru preluare de informații (exemplu: listarea restaurantelor)
- POST – introducerea unor informatii noi in baza de date (exemplu: adaugare restaurant)
- PUT – actualizarea informațiilor din baza de date (exemplu: modificarea informațiilor despre un utilizator)
- DELETE – pentru stergerea unor informații (exemplu: ștergerea unui produs din baza de date)

Pentru a identifica o clasă din pachetul Endpoint ca făcând parte din API-ul xpreeMenu și pentru a mapa metodele acesteia la diferitele cereri care pot veni din partea clienților se utilizează o serie de adnotări puse la dispoziție de platforma Google Cloud Endpoint. Pe baza acestor adnotări adaugate claselor, Cloud Endpoint mapează cererile venite din partea clienților la clasele și metodele responsabile de procesarea acestora. Mai jos este prezentat un fragment care exemplifică o clasă Endpoint și metoda acesteia responsabilă pentru cererea de citire a unui restaurant din baza de date:

```
@Api(name="restaurantendpoint",version="v1")
public class RestaurantEndpoint {
        @ApiMethod(name = "addRestaurant")
        public Restaurant getRestaurant(@Named(„id”) Long id) {
                // procesare cerere
                return restaurant;
        }
}
```

Pentru identificarea clasei RestaurantEndpoint ca fiind făcând parte din API am utilizat adnotarea @Api, specificând numele API-ului și versiunea acestuia. Cu aceste configurări minime clasa API devine funcțională, deoarece Google Cloud Endpoint va prelua automat toate metodele publice din clasa adnotată și le va expune ca metode publice, accesabile folosind HTTP REST. Pentru mai mult control asupra clasei API se

pot utiliza adnotările @ApiMethod pentru a specifica un nume metodei și @Named pentru a specifica parametrii pe care metoda API să îi ia in considerare împreună cu numele sub care clientul trimite acestei parametri.

Este de remarcat faptul că nu a fost precizată metoda HTTP care să fie utilizată pentru procesarea cererii, Google Cloud Endpoint determinând acest lucru pe baza semnăturii metodei. Dacă se dorește specificarea unei anumite metode HTTP pentru procesarea cererii, acest lucru se poate face cu ajutorul atributului httpMethod adăugat la adnotarea @ApiMethod.

3. Entity

Pachetul Enity conține toate entitățile care sunt persistate în Google DataStore folosind Java Persistance API (JPA v1.0). În descrierea acestor clase se utilizează diferite adnotări specifice JPA pentru a stabili entitățile și atributele care trebuie persistate, dar și pentru a descriere relațiile dintre entități.

În cadrul componentei server cele mai importante entități persistate în DataStore sunt următoarele: UserType, User, Restaurant, Menu, MenuItem, Order, OrderItem și Table.

Pentru a identifica o entitate și a o persista în DataStore se utilizează adnotarea @Entity, iar pentru a specifica atributul care reprezintă cheia primară a entității se folosește adontarea @Id. Dacă se dorește utilizarea unei chei primare care să fie unice și sa se genereze automat se poate utiliza adnotarea @GeneratedValue. În ceea ce privesc atributele care să fie stocate în baza de date acestea trebuie sa fie declarate ca un tip de date care este automat persistat sau să se specifice acest lucru implicit utilizând asupra atributului adnotarea @Basic.

De asemenea, relațiile dintre entități sunt specifice tot cu ajutorul adnotărilor. De exemplu, pentru specificarea unei relații de 1-1 se utilizează adnotarea @OneToOne, iar pentru specificarea unei relații de 1-N se utilizează adontarea @OneToMany. Aceste adnotări care reprezintă relații între entități se specifică pe atributele entităților care fac obiectul acestor relații.

4. Service

Clasele din pachetul „Service” sunt punctul central al nivelului de Business Logic, ele realizând legătura între nivelul de „Presentation” și nivelul de „Data Access”. Serviciile sunt responsabile de realizarea diferitelor operații de business solicitate din nivelul de „Presentation”, ele înglobând logica din spatele acestor cereri și cunoscând clasele DAO catre trebuie utilizate.

Fiecare serviciu este reprezentat de o interfață și de implementarea interfeței respective. În interfață se află antetul metodelor utilizate de fiecare serviciu, iar implementarea metodelor se află în clasele concrete care implementează interfețele respective. De asemenea, tot în clasele concrete de servicii se utilizează instanțe ale claselor DAO pentru a obține o legătură la baza de date, instanțe care se obțin cu ajutorul unei clase DAOFactory.

Principalele servicii ale componentei server sunt următoarele: UserService, UserTypeService, RestaurantService, MenuService, MenuItemService, OrderService, OrderItemService și TableService.

5. Util

Pachetul „Util” conține clase ajutătoare pentru serviciile componentei server. Aici se poate găsi mai multe clase de tip Factory, fișiere de configurare sau clase care furnizează funcționalități comune tuturor serviciilor sau altor procese de business.

Cele mai importante clase de tip Factory prezente în pachetul „Util” sunt:

- **EMF – Entity Manager Factory**, care utilizează design pattern-ul **Singleton** pentru a returna o instanță de tip **EntityManager**, care va fi utilizată pentru realizarea operațiilor asupra bazei de date.
- **DAOFactory –** clasă care implementează design pattern-ul factory pentru instanțierea diferitelor obiecte de tip DAO de către clasele de servicii. Prin utilizarea acestui design pattern, sistemul devine independent de tipul bazei de date utilizat pentru stocare, fiind foarte ușor de modificat acest factory astfel încât sistemul să funcționeze cu alt tip de bază de date fără a influența logica de business.

6. Dao

Clasele de tip DAO (Data Access Object) oferă serviciilor modalitatea de legătură cu baza de date, realizând conexiunea și interogările necesare. Fiecare entitate din cadrul proiectului are asociată o clasa DAO corespunzătoare, care este instanțiată în cadrul serviciilor utilizând un DAOFactory prezentat în pachetul Util.

Fiecare clasă DAO e reprezentată de o interfață și de implementarea interfeței respective care cuprinde operații executate folosind tehnologia JPA și specifice surse de date utilizate.

În proiectul curent cele mai importante clase DAO sunt: UserTypeDAO, UserDAO, RestaurantDAO, OrderDAO, OrderItemDAO, MenuDAO, MenuItemDAO, TableDAO.

Operațile de creare și editare pe fiecare dintre clasele DAO specificate se realizează cu metodele specifice: insert, update, delete. Aceste metode primesc ca și parametrii entitățile necesare care la rândul lor conțin informațiile care trebuie persistate în baza de date, excepție făcând metoda delete care primește ca și parametru cheia obiectului care trebuie șters.

Pentru operațiile de căutare în baza de date, pentru toate clasele DAO există două metode general valabile: getAll și findById. Metoda getAll returnează o listă cu toate înregistrările stocate în baza de date, de tipul entității din care se apeleaza metoda, iar findById primește ca parametru un obiect Key ce reprezintă cheia primară a entității salvate în tabelele în DataStore. Mai există metode specifice anumitor clase DAO cum ar fi: findByUsername sau findByEmail care sunt specifice clasei UserDAO. Metoda findByEmail primește ca parametru un obiect de tipul String ce reprezintă email-ul utilziatorului căutat, această metodă fiind utilizată la operațiunea de autentificare.

Pentru conectarea și executarea operațiilor pe baza de date, fecare clasă DAO folosește un obiect EntityManager ca se obține la instanțierea clasei DAO de către DAOFactory, care la rândul său utilizează clasa EMF (Entity Manager Factory) din pachetul „util”.

5.5. Componenta mobilă

Componenta mobilă a sistemului xpressMenu este reprezentată de aplicația client Android dedicată clienților restaurantelor.

Aplicația rulează pe dispozitivele Android și a fost implementată în limbajul Java, având la dispoziție tool-urile din Android SDK. Android SDK conține debugger, librării, emulatoare, documentație, exemple de cod și tutoriale. Ca și mediu de dezolvatre suportat oficial este Eclipse, care folosește plugin-ul ADT (Android Development Tool) pentru dezvoltare aplicațiilor Android.

În acest subcapitol vor fi prezentate diagrama de pachete și diagrama de clase a aplicației xpressMenu. De asemena, vor fi descrise o parte din clasele și metodele mai importante ale sistemului.

5.5.1. Diagrama de pachete

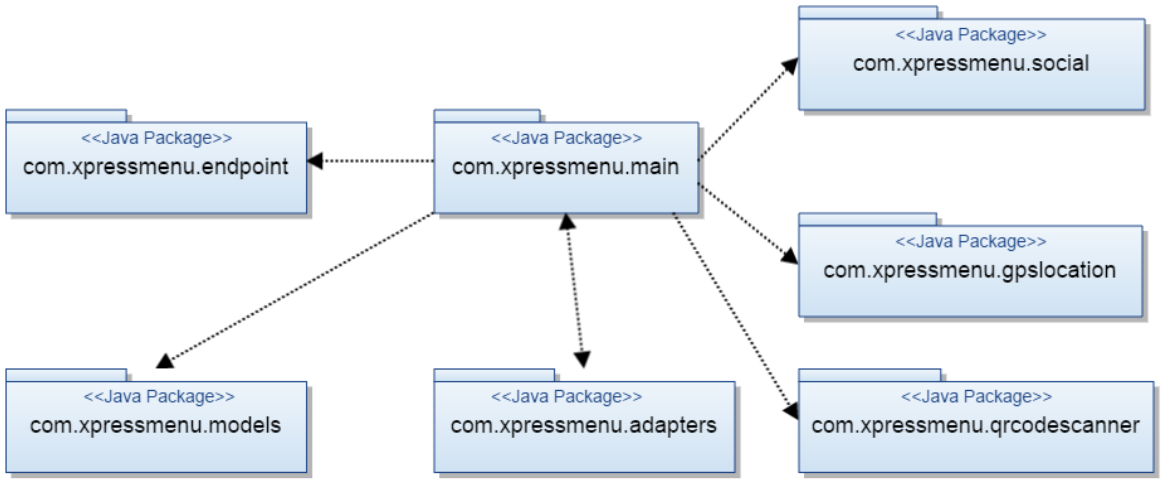


Figura 5.7 Diagram de pachete a componentei mobile

În **Figura 5.7** sunt ilustrate cele 7 pachete ale componentei mobile a sistemului xpressMenu împreună cu dependențele dintre ele.

Pachetul **com.xpressmenu.main** este punctul central al aplicației și depinde de toate celelalte 6 pachete. Acest pachet conține clasele care extind superclasa Activity și fiecare are asociat un layout și un meniu definte sub forma de fișiere XML. Aceste layout-uri reprezintă View-urile aplicației, iar clasele de tip Activity reprezintă Controller-ele.

Pachetul **com.xpressmenu.adapters** este resposabil de generarea listelor afișate de către aplicație (de exemplu lista restaurantelor) și realizează legătura între listele de

obiecte și View-urile pe care aceste liste se vor mapa. Fiecare clasă din acest pachet extinde clasa de bază BaseAdapter.

Pachetul **com.xpressmenu.endpoint** conține clase de tip servicii care sunt responsabile de interacțiune cu componente server. Aceste clase utilizează protocolul REST HTTP și interfața publică de acces expusa de componenta server pentru a realiza cereri și a interpreta răspunsurile primite.

Pachetul **com.xpressmenu.gpslocation** conține o singură clasă care se ocupă determinarea locației curente a utilizatorului. Această clasă extinde superclasa Service și implementează interfața LocationListener.

Pachetul **com.xpressmenu.models** conține clase care modelează diferitele entități utilizate în logica de business. Clasele din care este compus pachetul se ocupă de gestionarea comenzilor, a produselor, a categoriilor de produse, a listei de restaurante și a diferitelor informatii specifice utilizatorului.

Pachetul **com.xpressmenu.social** conține clasele responsabile de interacțiunea cu serviciile de socializare și facilitează implementarea funcționalităților de autentificare utilizând rețele de socializare. Aceste clase extind clasa de bază Activity, dar și alte clase sau extind interfețe specifice serviciilor de socializare pe care le implementează. De exemplu în cazul clasei care implementează autentificarea utilizând serviciile furnizate de Google+ va implementa interfața GooglePlayServicesClient.

Pachetul **com.xpressmenu.qrcodescanner** conține clasele care implementează funcționalitate de scanare a unui cod QR. Aceste clase extind clasa Activity.

5.5.2. Diagrama de clase

În continuarea acestui subcapitol va fi prezentată diagrama tuturor activităților care intră în componența aplicației mobile xpressMenu în **Figura 5.8**. Diagrama ilustrează activitățile propriu-zise împreună cu relațiile de dependență dintre acestea. Rolul acestor activități este de a răspunde și a procesa evenimentelor care apar pe View-urile asociate lor. Fiecare activitate extinde clasa de bază Activity sau ListActivity în funcție de rolul pe care îl indeplinește.

Fiecare activitate are asociat un meniu și un layout, care se definesc utilizând XML. Layout-urile care reprezintă liste (ListView) sunt compuse din elementele de listă care la rândul lor reprezintă layout-uri separate care sunt definite tot cu ajutorul XML. Aceste layout-uri separate ne permit să realizăm elemente de listă complexe, cum este de exemplu în cazul activității care afisează lista de produse dintr-o anumită categorie a meniului. Fiecare element din această listă conține o imagine a produsului, două elemente TextView pentru afișarea numelui și a prețului produsului.

Pentru a face legătura între listele de obiecte și lista asociată unui anumit View se utilizează așa numite adaptoare care au rolul de a mapa obiectele pe un element al listei. Mai jos se poate observa o listă a unora dintre perechile de forma Activity – Adapter - Model, reprezentând activități care au asociate liste, care la rândul lor au asociat un adaptor care mapează conținutul modelului pe un anumit view:

- MenuCategoryActivity – MenuCategoryAdapter – MenuCategory
- RestaurantsListActivity – RestaurantsListAdapter – Restaurant

- OrderActivity – OrderListAdapter – OrderItem

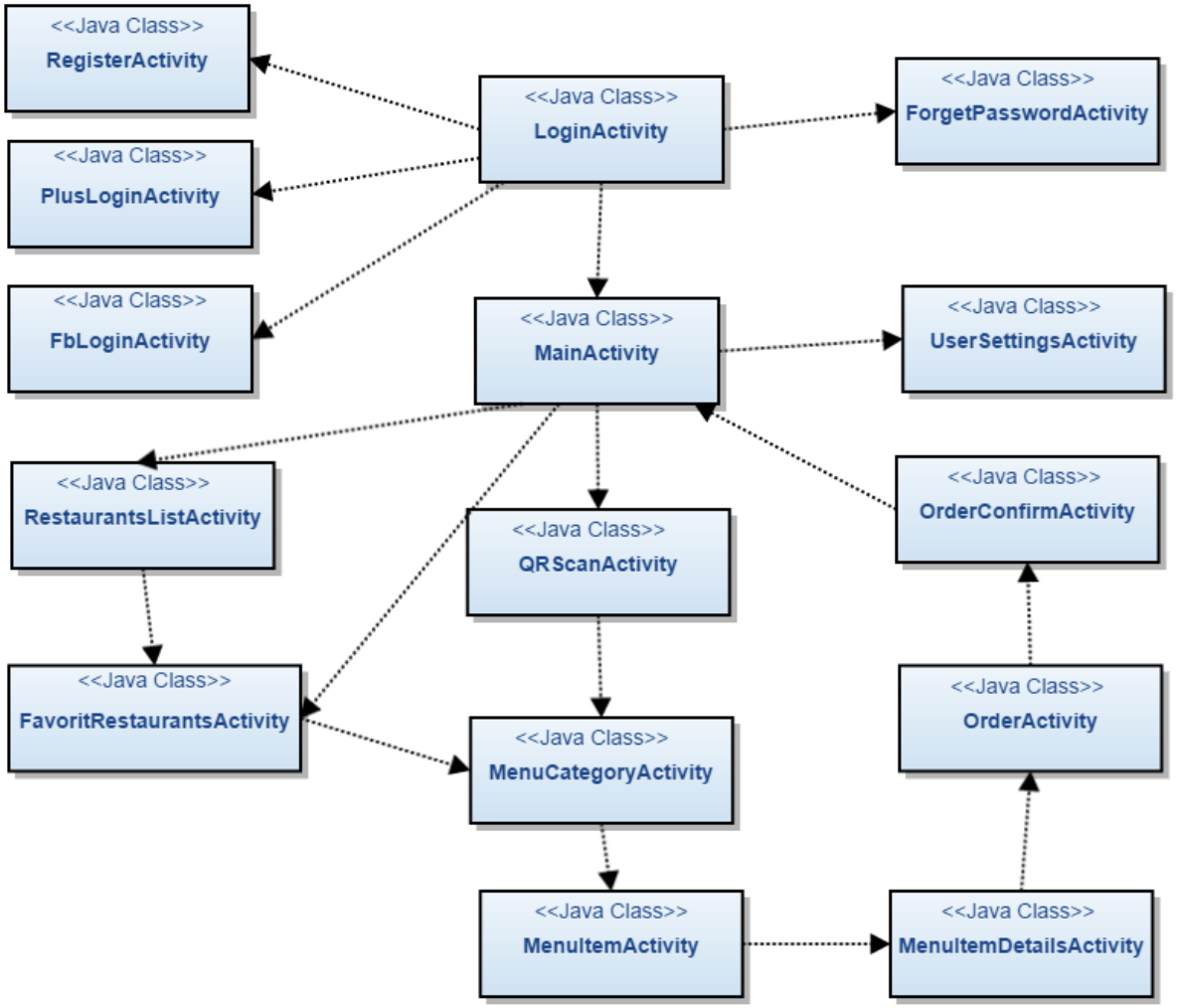


Figura 5.8 Diagram de clase a componentei mobile

5.5.3. Descrierea claselor importante

Acest subcapitol are rolul de a prezenta detalii de implementare despre celor mai importante clase ale aplicației xpressMenu. Va fi detaliată clasa Order care se ocupă de gestionarea comenzii unui utilizator și clasa EndpointAsyncTask care este responsabilă de comunicarea asincronă cu componenta server.

5.5.3.1. Clasa Order

Clasa Order este responsabilă de gestiunea comenzii unui client și conține toate atributele și metodele necesare pentru efectuarea operațiilor asupra conținutului acesteia.

În continuare sunt descrie pe scurt atributele principale ale acestei clase:

- **orderItems** : ArrayList<OrderItem> - reprezintă produsele din comanda propriu-zisa a clientului, reprezentate sub forma unor obiecte de clasă OrderItem;

- **totalPrice** : Float – valoarea totala a comenzii;

Următoarea listă prezintă principalele metode ale acestei clase împreună cu o scurtă descriere:

- **getOrderContent()** : ArrayList<OrderItem> - returnează toate produsele din comandă;
- **addNewItem(newItem:OrderItem)** : void – adaugă un produs la comanda curentă;
- **removeItem(position:int)** : void - ștergerea un produs din comandă;
- **containsItem(itemId:String)** : boolean - verificarea dacă un anumit produs se află deja în comandă;
- **clearOrder()** : void – șterge tot conținutul comenzii;
- **computeTotalPrice()** : Float – calculează valoarea totală a produselor din comandă;

5.5.3.2. Clasa EndpointAsynkTask

Clasa EndpointAsynkTask este subclasă privată a tuturor activităților care comunică cu componenta server. Deoarece aceste clase realizează conexiuni la alte servicii externe componentei mobile și depind de timpul de răspuns al acestora, ele trebuie să ruleze într-un thred separat fața de cel principal pentru a nu bloca aplicația Android. În consecință fiecare această subclasă extinde clasa AsyncTask<String, Void, String> care permite efectuarea de operații în background prin intermediul thread-urilor, fiind o clasă Helper pentru clasele Thread și Handler.

EndpointAsynkTask suprascrie trei dintre metodele clasei AsyncTask:

- **onPreExecute()** : void – aici se specifică cod care se execută înainte de a se porni task-ul asincron. De obicei este utilizată pentru a afișa un ProgressDialog pe perioada în care se procesează cererea către componenta server;
- **doInBackground(params:String[])** : String – aici se specifică codul care să ruleze în noul thread creeat. În cazul aplicației noastre efectuează un request de tip GET sau POST, în funcție de necesitate, cu ajutorul unui obiect de tip HttpClient și HttpGet sau HttpPost. Params reprezintă parametrii care sunt trimiși la componenta server.
- **onPostExecute(result:String)** : void – aici se scrie codul care va rula după finalizarea task-ului asincron. Este apelată în momentul în care ajunge răspunsul de la componenta server și primește ca parametru conținutul răspunsului respectiv.

Un caz în care se utilizează această clasa privată este în momentul în care utilizatorul scanează codul QR aflat pe masa restaurantului și se trimite o cerere la server prin care se solicită meniul restaurantului respectiv. Clasa EndpointAsynkTask este apelată folosind următoarea linie de cod:

```
String[] params = {restaurantId};
new EndpointAsyncTask(QRScanActivity.this).execute(params);
```

Primul parametru reprezintă activitatea din care se instanțiază această clasă privată, iar apoi se apelează metoda *execute()* care are rolul de a porni task-ul asincron și la care se trimite parametrul *params* care conține Id-ul restaurantului pentru care se cere meniul.

Dacă acest task asincron este realizat cu succes atunci utilizatorul este direcționat către meniul restaurantului, în caz contrar va primi un mesaj de eroare.

5.6. Diagrama de deployment

Diagrama de deployment din **Figura 5.9** prezintă componentele hardware pe care rulează componentele software și modul în care aceste componente sunt interconectate. Aceste componente mai sunt denumite și artefacte ale sistemului. Scopul acestei diagrame este de arăta structura hardware a sistemului.

Utilizatorii sunt cei care beneficiază de pe urma întregului sistem. Clientul restaurantului utilizează dispozitivul mobil Android pe care este instalată componenta mobilă a sistemului xpressMenu, reprezentată de aplicația Android. Aplicația comunică cu componenta server a sistemului reprezentată de platforma Google App Engine care furnizează servicii web folosind protocolul HTTP.

Ospătarul sau administratorul restaurantului este un alt beneficiar al întregului sistem. Aceștia utilizează un calculator personal care cu ajutorul unui client web accesează componenta web a sistemului xpressMenu și realizează operațiuni pe baza de date a sistemului utilizând serviciile puse la dispoziție de componenta web. Și în acest caz comunicarea între componenta web și componenta server a sistemului se realizează folosind protocolul HTTP.

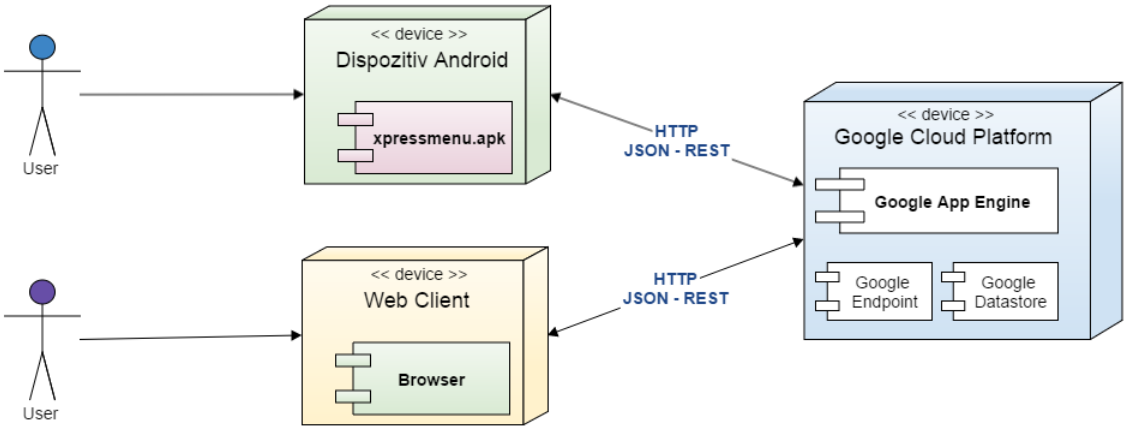


Figura 5.9 Diagrama de deployment a sistemului xpressMenu

Capitolul 6. Testare şi Validare

Acest capitol are rolul de a prezenta procesele de testare şi validare realizate asupra sistemului implementat. Testarea nu garantează funcționarea corectă a sistemului în orice condiții, dar identifică cât mai multe cazuri în care sistemul nu funcționează corect pentru a putea fi găsite soluții. În ceea ce privește procesul de validarea, acesta are scopul de a confirma dacă aplicația îndeplinește cerințele funcționale și non-funcționale și cazurile de utilizare realizate la proiectarea aplicației.

În ultimii ani device-urile mobile au devenit tot mai complexe din punct de vedere al funcționalităților și au căștigat un procent tot mai mare din piața de gadget-uri. Multe aplicații, care au fost dezvoltate inițial ca și aplicații desktop au fost converite treptat spre aplicații web, iar acum migrează spre aplicații mobile. Încă de la apariția primelor dispozitive mobile era cunoscut faptul ca acestea au resurse limitate, ceea ce pune un accent mai mare pe calitatea și fiabilitatea acestor aplicații.

Metoda principală de a măsura calitatea unor produse și de a decide gradul lor de fiabilitate, este testare. Acest capitol prezintă principalele metode utilizate pentru testarea sistemului dezvoltat. Testarea și validarea a fost realizată atât pentru aplicația mobile, cât și pentru componenta de server a sistemului xpressMenu.

6.1. Testarea manuală a aplicației mobile xpressMenu

Testarea manuală a aplicației xpressMenu s-a realizat treptat după implementarea unei noi funcționalități, astfel s-au putut determina din timp a anumitor bug-uri care apoi au fost rezolvate. Acest proces iterativ de rezolvare treptată a bug-urilor identificate pe parcursul dezvoltării aplicației are mai puține costuri în ceea ce privește timpul și complexitatea soluției decât în cazul în care aceste probleme ar fi abordate la sfârșitul implementării.

În procesul de dezvoltare a aplicației mobile xpressMenu s-au utilizat câteva tehnici de testare manuală care sunt specifice testării dispozitivelor tradiționale, însă se pot aplica cu ușurință și pe dispozitivele mobile. Dintre acestea se poate aminti: Black Box Testing și White Box Testing.

Black Box Testing

Black Box Testing mai este cunoscută sub denumirea de testare funcțională și are rolul de a determina dacă cerințele funcționale de bază stabilite inițial au fost implementate. Această metodă de testare se concentrează doar pe testarea aplicației fără ca tester-ul să dețină acces la codul sursă sau să cunoască detaliile interne ale aplicației.

În acest proces de testare se urmăresc mai multe aspecte pornind de la instalarea aplicației, interfața cu utilizatorul și până la testarea API-urilor sau a elementelor ce țin de securitatea aplicației.

În cazul fiecărei funcționalități implementate s-a testat dacă aplicația mobilă se conectează cu succes la componenta server, dacă comunică corect prin intermediul API-ului și dacă informațiile transmise sunt corect salvate în baza de date. De asemenea, s-a urmărit permanent dacă interfața răspunde corect la interacțiunea cu utilizatorul.

White Box Testing

După cum și numele sugerează, această metodă de testare este opusul metodei Black Box Testing. Daca în cazul testării funcționale tester-ul nu are acces la detaliile interne ale aplicației, în cazul white box testing se presupune ca acesta are acces la sursa aplicației. În consecință, tester-ul cunoaște structura sistemului, modul de implementare și de lucru, putând astfel să urmărească fiecare metodă a programului dacă se execută cum a fost intenționată.

După implemntarea fiecărei funcționalități din cadrul API-ului xpressMenu, aceasta a fost testată mai întâi în cadrul browser-ului web folosind tool-ul pus la dispoziție de către Google App Engine. După comunicarea cu succes între browser-ul web și componenta server, s-a trecut la testarea comunicării între componenta mobilă și componenta server a sistemului. De fiecare dată s-a urmărit descoperirea și rezolvarea din timp a diferitelor bug-uri.

6.2. Testarea componentei server de pe Google App Engine

În testarea componentei server s-a urmărit în primul rând funcționarea corectă a tuturor cererilor transmise către API-ul implementat cât și comportamentul acestuia la încărcare în cazul cererilor simultane.

Testarea componentei server s-a realizat utilizând aplicația Postman (www.getpostman.com). Postman este un tool de testare care suportă protocolul HTTP și permite trimiterea de cereri la un serviciu web, care pot să cuprindă conținut XML sau JSON.

Pentru testarea componentei server au fost trimise cereri multiple și variate către fiecare dintre URI-urile suportate de acesta. Pentru fiecare dintre funcționalitățile componentei server s-au urmărit atât cazuile în care cererea și conținutul acesteia erau valide, cât și cazurile când acestea nu erau valide.

Comportamentul sistemului la încărcare în cazul cererilor simultane a fost testat tot cu ajutorul Postman, prin trimiterea mai multor cereri identice la componenta server. Cu ajutorul consolei puse la dispoziție de Google AppEngine-ului se pot vizualiza informații despre starea serviciului nostru. Aceste informații includ media timpului de răspuns la o cerere, numărul cererilor făcute la fiecare URI, numărul erorilor, mărimea datelor transmise. Folosind consola App Engiene s-au urmărit și baza de date pentru a verifica finalitatea cu succes a cererior transmise către server.

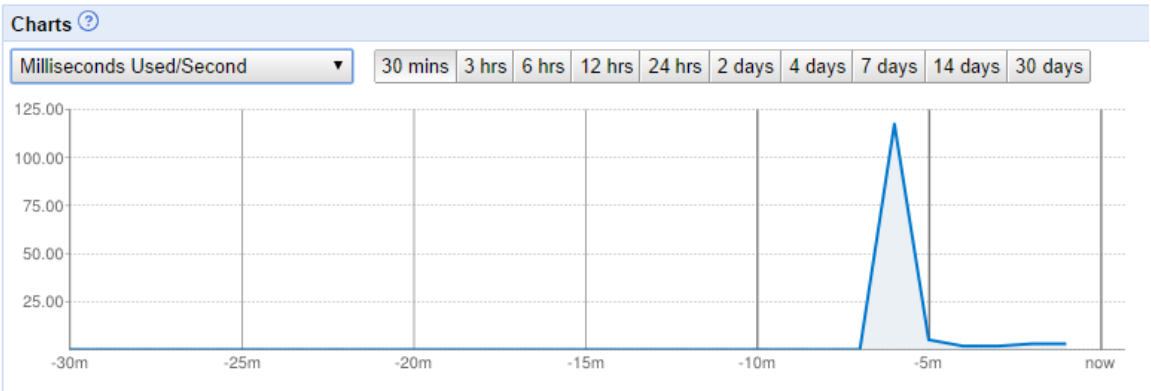


Figura 6.1 Timpul (miliseconde/sec) necesar aplicației pentru a prelua o cerere

În **Figura 6.1** se poate vizualiza timpul necesar componentei server pentru a prelua o cerere fără a include timpul necesare procesării acesteia. Din analiza graficului se poate constata o creștere a timpului necesar preluării primei cereri adresate componetei server, de aproximativ 120ms, după care media nu depășește 5ms.

Capitolul 7. Manual de Instalare si Utilizare

7.1. Instalarea aplicației mobile xpressMenu

7.1.1. Cerințele hardware

- Un telefon inteligent sau tabletă pe care să fie instalat SO Android;
- Cel puțin 15 MB de memorie fizică;
- Conexiune la Internet;

7.1.2. Cerințele software

- Dispozitivul mobil trebuie să aibă instalată minim versiunea Android 3.0;

7.1.3. Pașii de instalare

Pentru instalarea aplicației mobile xpressMenu folosind fișierul .apk e nevoie de:

- Fișierul .apk al aplicației;
- Un cablu de date USB pentru conexiunea dintre smartphone și calculator;

Pașii necesari instalării aplicației sunt:

1. Conectați dispozitivul mobil la calculatorul pe care se află xpressMenu.apk folosind cablul de date USB
2. Copiați fișierul .apk de pe calculator pe dispozitivul mobil
3. Din meniul de Settings al telefonului, navigați spre Applications, și bifați „Unknown sources" care va permite instalarea de aplicații din alte surse, altele decât Google Play Marketplace.
4. Căutați pe dispoztiv fișierul xpressMenu.apk și se apăsați butonul Install pentru instalarea aplicației;

7.2. Manual de utilizare

7.2.1. Înregistrare, Autentificare, Recuperare parolă

După efectuarea pașilor de instalare a aplicației se poate trece la utilizarea ei. Pentru început căutăm aplicația xpressMenu pe telefon și o rulăm. La început utilizatorul este întâmpinat de ecranul de autentificare din **Figura 7.2**. Dacă utilizatorul deține un cont valid acesta poate să se autentifice în aplicație, iar în caz contrar poate să aleagă opțiunea de înregistrare („Register") care îl va duce la ecranul din **Figura 7.1** unde își poate creea un cont.

Dacă utilizatorul deține un cont valid care nu își mai amintește parola, acesta poate accesa pagina de resetare a parolei prin apăsarea butonului cu numele „Forgot password". Această acțiune îl duce la ecranul din **Figura 7.3** unde acesta își introduce adresa de email unde va primi noua parolă.

De asemena, utilizatorul are posibilitatea de a se autentifica sau înregistra utilizând contul său de Google+.

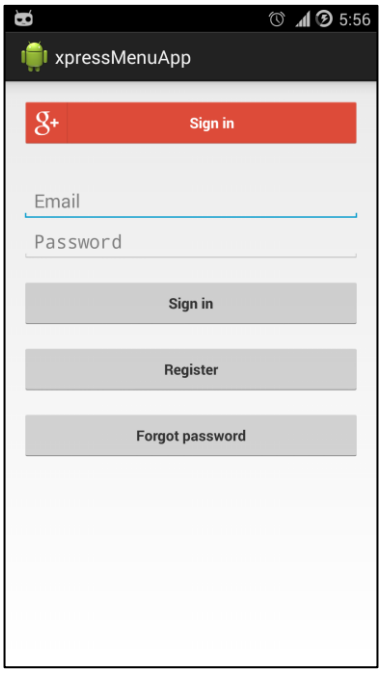


Figura 7.2 Pagina de autentificare

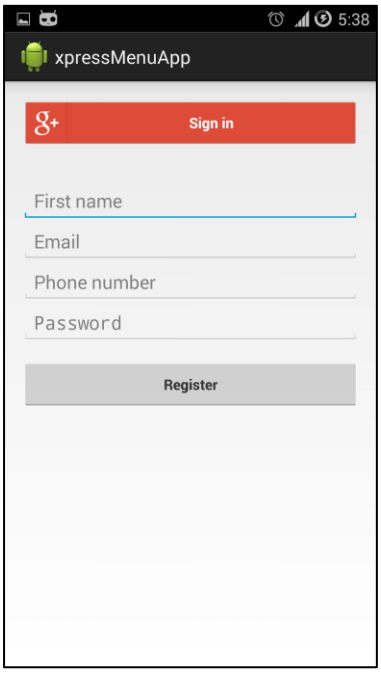


Figura 7.1 Pagina de inregistrare

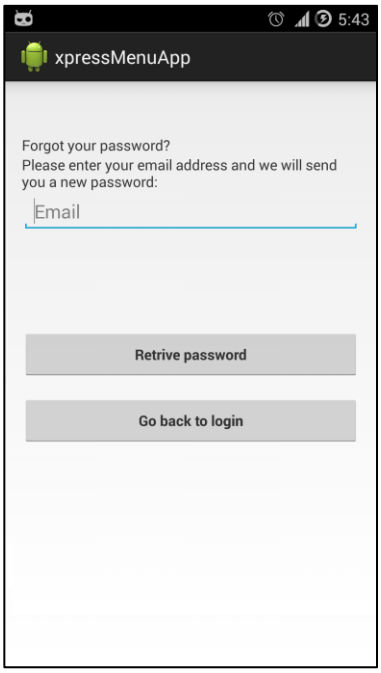


Figura 7.3 Pagina de resetare a parolei

Capitolul 8. Concluzii

În acest ultim capitol sunt prezentate rezultatele obținute și măsura în care obiectivele propuse au fost atinse. De asemenea, în finalul capitolului sunt enumerate câteva devoltări ulterioare posibile pentru îmbunătățirea sistemului.

8.1. Realizări

Sistemul xpressMenu a fost dezvoltat pentru a gestiona mult mai facil activitatea dintr-un restaurant și pentru a oferi clienților o nouă experiență, prin oferirea unei aplicații mobile care să le permită vizualizarea digitală a meniului unui restaurant și plasarea unei comenzi fără a mai fi nevoie de intervenția chelnerului. Realizarea acestui principal obiectiv a adus cu sine și dezvoltarea unor funcționalități secundare care să vină în ajutorul atât al restaurantelor cât și al clienților acestora.

Astfel, aplicația vă permite să vizualizați cu succes meniul unui restaurant și să plasați o comandă când vă aflați în incinta acestuia fără a mai fi nevoiți să asteptați venirea ospătarului. Acesta din urmă primește notificarea comenzii trimise pe aplicația web care o utilizează pentru gestionarea comenzilor. Acest principal obiectiv al proiectului, este destul de important deoarece sistemele similare nu au această funcționalitate.

În legătură cu celelalte obiective secundare propuse putem afirma ca au fost realizate aproape în totalitate. Am dezvoltat un sistem de autentificare în aplicație care este necesar când utilizatorul dorește să utilizeze aplicația și în același timp am dezvoltat o soluție care să identifice dacă utilizatorul se află în incinta restaurantului fără a fi nevoiți să utilizăm funcționalitatea de GPS. De asemenea, utilizatorul are și posibilitatea de a se autentifica utilizând conturile sale de pe rețelele de socializare, iar în momentul în care s-a autentificat în aplicație nu mai trebuie să se autentifice din nou la redeschiderea acesteia.

Dintre obiectivele secundare realizate se numără și posibilitatea vizualizării unei liste cu restaurante aflate în proximitatea utilizatorului, dar și o listă cu restaurantele favorite. De asemenea, s-a reușit implementarea și a funcționalităților care permit utilizatorului să localizeze pe hartă un restaurant sau să contacteze telefonic restaurantul direct din aplicație, fără ca utilizatorul să fie nevoit să folosească alte aplicații secundare.

Un alt obiectiv secundar care aduce un beneficiu utilizatorilor este posibilitatea de a realiza o rezervare la unul dintre restaurantele preferate din doar câțiva pași.

Printre obiectivele secundare care nu au reușit să fie implementate în aplicația mobilă se numără posibilitatea de a evalua serviciile din cadrul unui restaurant și posibilitatea de a vizualiza un istoric al comenzilor făcute.

Din punctul de vedere al chelnerilor și al administratorilor de restaurante, obiectivele principale de realizare a unor meniuri digitale și de gestionare a comenzilor venit din partea clienților folosind aplicația mobila xpressMenu au fost implementate cu succes. Chelnerii au posibilitatea de a vizualiza în aplicația web destinată acestora mesele care sunt ocupate din restaurant și comenzile primite din partea clienților.

Administratorii restaurantelor au posibilitatea de a gestiona produsele din meniu și personalul din restaurant care are acces la platforma web. De asemenea, aceștia au

posibilitatea de a vizualiza un istoric al comenzilor din restaurant și rapoarte cu privire la evoluția comenzilor și a valorilor acestora pe o anumita perioada.

Din obiectivele secundare nerealizate în ceea ce privește componenta web dedicată restaurantelor se numără posibilitatea ospătarilor de edita comenzile primite și posibilitatea administratorilor de a promova reduceri în rândul clienților fideli.

Din punct de vedere al realizărilor tehnice personale se numără aprofundarea limbajului Java în ceea ce privește dezvoltarea aplicațiilor Android și învățarea dezvoltării de aplicații și servicii pentru platforma Google App Engine.

8.2. Dezvoltări ulterioare

Sistemul xpressMenu poate fi îmbunătățit prin dezvoltarea obiectivelor nerealizate, prin îmbunătățirea funcționalităților realizate, dar și prin adăugarea de noi funcționalități ca răspuns la nevoile utilizatorilor. Criteriile de performanță și îmbunătățirea interfeței utilizator sunt aspecte care pot fi avute în vedere la dezvoltările ulterioare ale aplicației. În lista următoare pot fi găsite câteva propuneri pentru noi funcționalități care ar putea fi adăugate în viitor sistemului:

- Identificarea clientului în restaurant pe baza tehnologiei NFC
- Disponibilitatea aplicației în mai multe limbi de circulație internațională
- Posibilitatea de a interacționa cu alți utilizatori ai aplicație mobile xpressMenu
- Posibilitatea de a adauga evaluari scrise în cazul fiecărui restaurant și al produselor individuale
- Posibilitatea de a achita nota de plată utilizând aplicația mobilă
- Posibilitatea de a invita prieteni atunci când se realizează o rezervare
- Oferirea de informații despre o dietă sănătoasă în funcție de produsele comandate
- Posibilitatea de a conecta componenta web dedicată ospătarilor la o imprimantă pentru tipărirea notei de plată și a bonului de casă
- Posibilitatea generării din cadrul aplicației a documentelor contabile

Pentru a îmbunătăți funcționalitățile actuale și pentru a descoperi posibile noi funcționalități pentru sistemul xpressMenu ar fi indicat realizarea unui test de câteva săptămâni în cadrul unui restaurant pentru a observa comportamentul clienților și al personalului restaurantului.

În concluzie, aplicația xpressMenu poate fi utilizată cu succes, deoarece obiectivele de bază au fost realizate, iar dezvoltările ulterioare sunt menite doar să ofere utilizatorilor alte funcționalități ajutătoare sau să le îmbunătățească pe cele actuale.

Bibliografie

[1] *Designing for usability: key principles and what designers.* **John D. Gould, Clayton Lewis.** 1985.

[2] **Vardy, Mike.** Resistant to the inevitable: How technology is changing the restaurant industry. *The Next Web.* [Interactiv] http://thenextweb.com/insider/2012/09/22/how-technology-changing-restaurant-industry/.

[3] *Technology's Effect on Hotels and Restaurants: Building a Strategic.* **Koutroumanis, Dean A.** 1, Tampa : Journal of Applied Business and Economics , 2011, Vol. 12.

[4] Software Gestiune Restaurant. [Interactiv] Sunsys Software. http://www.sunsys.ro/programe-gestiune/restaurant.

[5] OpenTable - Free Reservations. *www.play.google.com.* [Interactiv] Google. https://play.google.com/store/apps/details?id=com.opentable.

[6] **William Voorsluys, James Broberg, and Rakkumar Buyya.** *Cloud Computing: Principles and Paradigms.* s.l. : John Wiley & Sons, Inc., 2011.

[7] **Scott, Kendall.** *The Unified Process Explained.* s.l. : Addison-Wesley Professional., 2001.

[8] **Dan Pilone, Neil Pitman.** *UML 2.0 in a Nutshell.* s.l. : O'Reilly Media, 2005.

[9] Prezentare JSON. *JSON - ECMA-404 The JSON Data Interchange Standard.* [Interactiv] http://json.org/json-ro.html.

[10] **Bas, L., Clements, P., Kazman, R.** *Software Architecture in Practice.* s.l. : Adison-Wesley, 1998.

[11] *ECMA-404 The JSON Data Interchange Format.* Geneva : ECMA International, 2013.

[12] Google Cloud Computing, Hosting Services & Cloud Suport. *Google Cloud Platform.* [Interactiv] Google. https://cloud.google.com/.

[13] Overview of Google Cloud Endpoints. *Google Developers.* [Interactiv] Google. https://developers.google.com/appengine/docs/java/endpoints/.

[14] OpenTable Integrates with Apple Pay to Enable Convenient Mobile Payments for Restaurant Goers. *CNN Money.* [Interactiv] http://money.cnn.com/news/newsfeeds/articles/prnewswire/NY08120.htm.

Anexa 1 – Lista de figuri

Anexa 2 – Lista de tabele