Cuprins

[1 Abstract 2](#_Toc11516313)

[1.1 Inteligentă artificială 2](#_Toc11516314)

[1.2 Rețele neuronale 2](#_Toc11516315)

[1.3 Programare avansată 2](#_Toc11516316)

[1.4 Rețele de calculatoare 2](#_Toc11516317)

[2 Tehnologii folosite 2](#_Toc11516318)

[2.1 Android 2](#_Toc11516319)

[2.2 Google Vision 3](#_Toc11516320)

[2.3 Google Pay 3](#_Toc11516321)

[2.4 NodeJS 4](#_Toc11516322)

[2.5 MongoDB 5](#_Toc11516323)

[3 Introducere 6](#_Toc11516324)

[4 Obiectivul proiectului 8](#_Toc11516325)

[5 Metodologia folosită 9](#_Toc11516326)

[5.1 Aplicatia Android 9](#_Toc11516327)

[5.2 Serverul NodeJS 10](#_Toc11516328)

[6 Descrierea sumara a solutiei 11](#_Toc11516329)

[7 Descrierea problemei 12](#_Toc11516330)

[7.1 Necesitate 12](#_Toc11516331)

[7.2 Dificultatea problemei 12](#_Toc11516332)

[7.3 Susținerea inițiativei 13](#_Toc11516333)

[7.4 Alte implementări 13](#_Toc11516334)

[8 Descrierea solutiei 15](#_Toc11516335)

[8.1 Aplicația Android 15](#_Toc11516336)

[8.2 Server 22](#_Toc11516337)

[8.3 Baza de date 24](#_Toc11516338)

[9 Rezultate 25](#_Toc11516339)

[10 Viitoare îmbunătățiri 26](#_Toc11516340)

[11 Concluziile lucrării 27](#_Toc11516341)

[12 Bibliografie 27](#_Toc11516342)

# Abstract

## Inteligentă artificială

## Rețele neuronale

## Programare avansată

## Rețele de calculatoare

# Tehnologii folosite

## Android

Android este un sistem de operare dezvoltat de Google care are la bază o versiune modificată a kernelului de Linux. El a fost proiectat în special pentru dispozitivele mobile care au ecran tacticl cum ar fi telefoanele inteligente sau tabletele. De asemeni, Google, a dezvoltat versiuni ale Androidului care sunt specializate pentru televizoare, Android TV, pentru mașini, Android Auto, pentru ceasuri, Wear OS, și multe alte tipuri de electronice.

Inițial a fost dezvoltat de compania Android Inc. care a fost cumpărată de Google în anul 2005. Prima dată, a fost anunțat public în anul 2007, urmând ca în septembrie 2008 să apară primul telefon care rula Android și anume HTC Dream.

Ultima versiune stabilă este Android 9 „Pie” lansată în August 2018 dar până în prezent are lasată a treia versiune beta a Android Q pe telefoanele din gama Pixel.

Android este cel mai bine vândut sistem de operare pentru smartfone-uri la nivel global începând din anul 2011 și pentru tablete din anul 2013. De asemeni Android reprezintă 70% din cota de piață a device-urilor mobile având 2.7 miliarde device-uri active.

Aplicațiile, care extind funcționalitățile device-ului, sunt scrise folosind “Android software development kit”, care în prezent a ajuns la versiunea 28. Limbajele folosite pentru a scrie aceste aplicații sunt Java și Kotlin (din 2017). Dezvoltarea de aplicații android tinde să se încline spre noul limbaj Kotlin deoarece este un limbaj mai ușor de învățat, mai ușor de citit și fiind realizat în maniera în care este făcut Python, aceleși lucruri scrise în Java tind să ocupe mai puține linii de cod în Kotlin.

Codul sursă este lansat de Google sub o licentă open source, motiv pentru care Android a și adunat o comunitate foarte mare de developeri și entuziaști pentru a folosi codul ca baza în diverse proiecte dezvoltate de comunitate care livrează update-uri pentru device-urile mai vechi, adaugă feature-uri pentru utilizatori avansați sau pentru a aduce Android-ul și pe device-uri care de regulă nu sunt livrate cu acest sistem de operare.

## Google Vision

Google Vision este o un tool pentru recunoșterea pattern-urilor din imagini. El este găzduit de Google Cloud, unde se află majoritatea tool-urilor publice de la Google. Această unealtă poate fi folosită prin două moduri diferite AutoML Vision sau Vision API.

Pentru AutoML Vision, clientul, trebuie să dețină un set cât mai mare de date de antrenament care să conțină perechi de imagini și informații care să indice lucrurile pe care modelul să le învețe. Acestă unealtă este foarte utilă pentru acei clienți care dispun sau pot crea un set de date îndeajuns de mare încât modelul obținut în urma antrenamentului cu acel set de date sa obțină acuratețea dorită. De asemeni, aceast tool ofera posibiltatea a antrena un model pentru lucruri mai specifice cum ar fi în cazul nostru recunoașterea numelor de produse și al prețurilor de pe imaginile cu notele de plată.

Vision API pe de altă parte, oferă modele deja antrenate pentru recunoașterea unor categorii de caracteristici și care poate fi integrat în orice fel de aplicație Android, iOS sau web prin faptul ca poate fi accesat prin request-uri la un serviciu de tip REST. Modelele sunt deja antrenate sa poate eticheta conținutul imaginilor, să detecteze obiecte sau fețe și de asemeni poate detecta text printat sau scris de mână.

Google Vision Text API este serviciul parte a Google Vision API care este specializat pentru detectarea textului din imaginile oferite ca input. În urma procesării, se obțin blocuri, linii și elemente care conțin text. Pentru oricare dintre acestea există posibilitatea obținerii textului sau a drepunghiului care încadrează textul detectat. Aceste drepunghiuri sunt reprezentate prin coordonatele colțurilor stânga sus și dreapta jos, coordonate calculate în pixeli.

## Google Pay

Google Pay este un serviciu care îmbunătățește metodele de plată din aplicațiile android și web. Focus-ul acestui serviciu este să ofere plăți sigure, rapide și ușor de integrat în aplicații.

Siguranța din Google Pay este oferită prin faptul că datele sunt păstrate în contul google și utilizatorul nu este nevoit să introducă datele sale de plată cum ar fi cele ale cardului de credit în aplicații care nu sunt de încredere.

Viteza de plată este asigurată prin faptul că plata începe prin apăsarea butonului specific google pay, apoi utilizatorul va naviga într-o pagină pentru selectarea metodei de plată, adaugarea de informații adiționale dacă este cazul și checkout-ul. Metodele de plată sunt setate din cadrul contului google, datele necesare fiind introduse o singură dată iar apoi selectarea metodei respective este doar la distanță de o apăsare de buton.

Acest serviciu este ușor de integrat prin faptul că google oferă pachete cu clientul pentru google pay care poate fi atașat ușor la dependințele aplcației iar apoi prin câteva linii de cod poate fi trimisă cererea de realizare a plății.

## NodeJS

NodeJS este un “JavaScript run-time environment”, care poate fi utilizat pe orice platformă (Windows, Linus, MacOS), open-source, care rulează cod JavaScript înafara browser-ului. Scopul principal este crearea de scripturi pentru partea de server.

Are la bază o arhitectura bazată pe evenimente care este capabilă de operații de tip asyncron. Această alegere de proiectare are ca țintă optimizarea transferurilor de date și al scalabilitații aplicațiilor web cu multe operații de tip input/output.

Printre utilizatorii de software NodeJS se află si companii cum ar fi Microsoft, IBM, Netflix, GoDaddy, PayPal și multe altele.

Ideea pentru NodeJS a apărut de la limitările pe care le aveau cele mai populare servere web din 2009, și anume Apache HTTP Server, care nu puteau suporta prea bine foarte multe conexiuni concurente.

NodeJS a fost inaugurat în anul 2009 la conferința europeana JSConf, iar din anul 2010 a fost atașat si un package manager numit npm care avea ca scop ajutarea developerilor să livreze pachete ce aveau să ajute comunitatea să se dezvolte.

Versatilitatea acestui mediu este dată de existența unui număr mare de așa zise module folosite pentru a livra funcționalități de baza cum ar fi file system I/O, networking (DNS, HTTP, TCP, UDP, TLS/SSL), date binare, funcții criptografice, streaming de date și multe alte funcționalități. Aceste module sunt proiectate sub formă de API-uri care sa reducă complexitatea scrierii de aplicații server.

## MongoDB

MongoDB este o bază de date, bazată pe documente, care este clasificată ca fiind parte din bazele de date NoSQL. Folosește obiecte de forma JSON legate printr-o schemă ce descrie strctura bazei de date. Este dezvoltată de către compania MongoDB Inc. sub o licenta de tip Server Side Public License. Fiind un proiect de tip open-source codul poate fi studiat pe pagina lor de Github[[1]](#footnote-1). MongoDB este suportat pe majoritatea sistemelor de operare actuale Windows, Linux Debian și MacOS. De asemeni acest tip de baze de date are suport si pentru acces fară prezența unui server care sa intervină, operațiunile asupra bazei de date putând fi făcute chiar si de pe sisteme de operare mobile cum ar fi Android și iOS.

O înregistrare intr-o bază de date MongoDB este un document care este de fapt o structură de date compusă din perechi cheie valoare. Aceste tipuri de documente sunt similare cu cele de tip JSON doar că aici se folosesc Binary JSON (BSON) care acceptă mai multe tipuri de date. Cheile din aceste documente pot fi asemănate cu coloanele tabelelor di bazele de date relaționale dar pot conține mai multe tipuri de date, incluzâand alte documente, array-uri sau array-uri de documente.

Documentele, care conțin cate o cheie primara care îl identifică unic, este unitatea de baza a acestor baze de date. Colecțiile sunt mulțimi de astfel de documente care funcționează echivalent cu tabelele din bazele de date relaționale doar că sunt limitate de faptul că datele dintr-o colecție nu pot fi răspindite și prin alte baze de date.

MongoDB Atlas este o de bază de dată complet gestionate și stocate în cloud oferită ca un serviciu care dispune de clustere la nivel global, și posibilități cum ar fi backups, upgrades si monitorizare. Este construit cu scopul de a economisi timpul dezvoltatorilor de aplicații și pentru a scăpa de grija problemelor care apar în timpul dezvoltării unei baze de date și în timpul managementului ei.

Printre utilizatorii de MongoDB se află si companii cum ar fi Google (în google search), IBm, Orange, HSBC, eBay, Cisco, Uber și altele.

# Introducere

Pe parcursul anilor de facultate, am ajuns să cunosc foarte mulți oameni prin participarea la diverse activități organizate de asociații, facultate sau de universitate. Pentru a cunoaște mai bine acești oameni sau pentru a întări legăturile cu noile cunoștiințe cea mai ușoară metodă este ieșirea împreună în pauzele de masă sau în timpul liber în locații cum ar fi restaurante, terase sau localuri. Problemele apar când vine nota de plată, acestea fiind amplificate cu fiecare persoană în plus la masă. De cele mai multe ori, in situații de acest gen, există acea persoană care poate plăti doar cu cardul sau persoana care nu are suma exactă pe care trebuie să o plătească pentru ca apoi bonul să trebuiască să treacă din mână in mână pe la fiecare persoană pentru a-și calcula fiecare suma de plătit sau in cea mai rea situație, după strângerea banilor sa nu se atingă suma totală. Regăsindu-mă in situații de genul celor descrise mai sus m-am gândit că aceasta este o problemă care se cere rezolvată printr-o metoda cât mai simplă, rapidă si la îndemâna tuturor. De asemeni, deoarece in anul III am lucrat în cadrul materiei Intelingență artificială cu un tool de OCR[[2]](#footnote-2) pentru recunoașterea manuscriselor, în cadrul materiei Rețele neuronale am lucrat la o rețea neuronală ce făcea clasificarea caracterelor din imagini și în cadrul materiei Android am învățat cum să creez aplicații Android care pot fi la îndemana oricărei persoane ce deține un smartfon, am considerat că am capacitățile necesare pentru realizarea unei soluții pentru problema identificată.

La momentul realizării acestei lucrări, există numeroase aplicații care țintesc rezolvarea acestei probleme prin diverse modalități fiecare cu plusurile si minusurile sale. Voi descrie în câteva rânduri două modalități întâlnite în aplicația Play Store de pe Android care rezolvă problema descrisă.

Una dintre modalități este cea in care aplicația este folosită ca un carnet de notițe unde un utilizator introduce în aplicație datele înscrise pe bon, adaugă persoanele care au de plătit, se asociază plați pentru fiecare persoană iar apoi plata se realizeze înafara aplicației. Această metoda nu salveză foarte mult timp la momentul plății, deoarece datele trebuiesc introduse manual. Motiv pentru care consider ca aplicațiile care implementează această metodă nu sunt axate pe plata la momentul venirii notei de plată. De asemeni nu toate aplicațiile care au adoptat această modalitate oferă acces la detaliile de plată pentru toți cei aflați la masă. Consider ca această modalitate este utilă pentru a ține evidența datoriilor in urma unei vacanțe cu prietenii spre exemplu, unde ai nevoie sa urmărești calculele pe o perioadă de cateva zile de la mai multe ieșiri. (Splitwise, Splid)

O altă modalitate este asemănătoare cu cea descrisă mai sus doar că mai apare ca opțiune pentru introducerea informațiilor de pe bon utilizarea camerei foto si crearea unei imagini cu detaliile produselor de pe bon care este procesată pentru a se obține o listă de produse si prețuri. Aici viteza de realizare a plații ar fi îmbunătațită doar că în toate aplicațiile ce utilizează procesarea imaginii, persoanele adăugate la plată nu au acces la detalii, sunt trecute ca etichete la plătile de făcut. În final, dinnou, plata are loc tot înafara aplicației ramânând nerezolvată problema cu lipsa sumei fixe pentru plată din partea unora dintre cei aflați la masă. (Blitter, Snap & Split Bill)

Necesitatea unui acest gen de aplicație poate fi întâlnită la oricare ieșire de grup la care participă mai mult de 3-4 persoane și care probabil nu se întalnesc îndeajuns de des încât să își poată rezolva datoriile. De asemeni mai există și situația în care cineva de la masă nu are în posesia sa suma exactă de plată sau cardul pentru a-și realiza partea sa de plată motiv pentru care ar fi nevoide de o metodă de plată din interiorul aplicației.

Soluția propusă de mine are ca scop oferirea unei modalități complete de plată a notelor din restaurante sau localuri cuprinzând partea de recunoaștere a detaliilor de pe bonuri, adăugarea de persoane la plata notei, selectarea produselor de plătit de către fiecare persoană de la masă iar apoi plata la un dispozitiv POS[[3]](#footnote-3) folosind telefonul persoanei care a inițiat plata si care a facut poză bonului. De asemeni, înainte ca inițiatorul plăți să realizeze plata către restaurant, cei aflați la masă vor accepta sa transfere către inițiator partea lor de plată. Astfel consider că procesul de plată este ușurat semnificativ fiind evitat momentul cand fiecare își face calcule pentru a vedea cum face ca să își plătească partea sa din nota de plată.

Această soluție ar aduce un plus din partea faptului ca fiecare om de la masă are acces la lista de produse pentru a-și selecta produsele consumate de el și poate realiza de unul singur partea lui de plată către inițiatorul comenzii care ulterior va realiza plata intregii comenzi prin intermediul plații la un dispozitiv POS oferit de restaurantul sau localul unde a avut loc consumația.

# Obiectivul proiectului

Cu acest proiect mi-am setat ca obiectiv îmbunătățirea procesului de rezolvare a notelor de plată care apar la ieșirile in grupuri mai mari de 3-4 persoane, în restaurante sau localuri, prin realizarea unei soluții ce implică o citire rapidă a informațiilor de pe bon, o adaugare mai rapida a persoanelor la plata notei utilizand un cod unic al facturii, implicarea activă a persoanelor de la masă in selectarea produselor consumate direct de pe device-ul lor si realizarea transferurilor pentru plata fiecărei parți din nota plată din interiorul aplicației.

În primă fază, introducerea datelor trebuie facută utilizănd o metodă automată care procesează o imagine a notei de plată si care este capabilă sa extragă numele produselor si prețul acestora. Astfel apare o îmbunătățire a vitezei eliminându-se timpul de introducere manuală a datelor de pe bon și fiind înlocuit cu timpul necesar fotografierii notei de plată și așteptarea rezultatelor procesării imaginii.

Pentru partea de adaugare a persoanelor la plată, pentru îmbunătățirea vitezei si pentru implicarea persoanelor activ la selectarea produselor, după trimiterea pozei cu nota de plată si primirea listei de produse si prețuri, să fie generat un cod unic al notei de plată cu care fiecare persoană de a masă să se poată alatura plații din interiorul aplicației de pe device-ul lor.

Pentru selectarea produselor obiectivul este ca fiecare persoana de la masă sa poată participa activ la selecție utilizănd propiul dispozitiv pentru a grăbi procesul de plată, după ce a fost introdus codul facturii sa fie afișată lista de produse si prețuri din care utilizatorul sa iși selecteze produsele consumate. De asemeni, fiecare persoană având lista cu produsele selectate de el la dispoziție, odată cu confimarea selecției se va realiza si plata către persoana care va plati nota la pasul final.

La finalul procesului se află inițiatorul, care urmează să realizeze plata utilizând device-ul Android dotat cu NFC[[4]](#footnote-4) pentru a face transferul final către restaurantul sau localul unde a avut loc consumația după ce a primit confirmarea de plată din parte celorlalți participanți la plată.

# Metodologia folosită

Soluția oferită de mine este împărțită in două componente: aplicația mobilă dezvoltată pe platforma Android, legată la serviciile Google Vision și Google Pay, și un server realizat in NodeJS care este legat la o bază de date realizată în MongoDB prin serviciul Mongo Atlas.

## Aplicatia Android

Am creat o aplicație Android în care este afișată interfața grafică din care utilizatorul poate realiza operațiunile de conectare cu un nume de utilizator, crearea unui plați noi, alăturarea la o plată deja creata, vizualizarea unui istoric cu plățile inițiate de acea persoană și vizualizarea unui istoric cu plațile pe care persoana le-a facut către alte persoane.

Pentru fiecare dintre aceste operațiuni există câte o activitate, spre care se poate naviga dintr-o bara de navigare situată în partea de jos a aplicației, și care se ocupă de afișarea informațiilor necesare realizării operațiunii. Pentru workflow-ul principal în care se introduce o factură noua, se realizează mai multe activități înlănțuite care fiecare adaugă un plus de informații facturii, motiv pentru care am creat un obiect special pentru factură care este pasat de la o activitate la alta și pe rând, fiecare adaugă informațiile obținute.

Pentru crearea clientului REST în aplicație, am folosit Retrofit. Acesta folosește o interfață pentru descrierea operațiunilor posibile în comunicarea cu serviciul, un set de clase ce descriu obiecte care vor fi trimise in cadrul requesturilor sau vor fi primite ca raspuns si un builder pentru realizarea conexiunii proproi-zise cu backend-ul.

Pe parcursul dezvoltării aplicației am încercat dezvoltarea unui sistem independent care sa recunoască textul din imagini dar de fiecare dată am ajung în impasul in care nu aveam destule date de antrenament si testare pentru rețeaua neuronală care sa facă recunoașterea textului din imagini. Din acest motiv am optat pentru Serviciul Google Vision. El este folosit în cadrul aplicației pentru prelucrarea imaginilor ce conțin bonurile si oferă ca răspuns textul din imagini separat pe linii de text și coordonatele în imagine, unde poate fi gasită acea bucată de text. Odată cu obținerea textul va fi rulat un algoritm care pe baza poziționării textului în pagină va asigna produsele la prețuri iar apoi va oferi o listă de prețuri si produse care va fi afisată utilizatorilor ce vor participa la acea factură.

Serviciul Google Pay va fi folosit odată de către cei ce s-au alăturat plații facturii pentru a face viramentul către utilizatorul care a inițiat plata facturii, iar apoi de către utilizatorul inițiator pentru a realiza plata finală a notei. În ambele situții, va fi afișat un buton pentru realizarea plații care după apăsare, va deschide o nouă pagina unde utilizatorul va putea vizualiza detaliile plații cum ar fi suma totală și beneficiarul plații cu un nume și o adresă de mail. Din această pagina utilizatorul poate confirma plată urmată de realizarea transferului din contul atașat in Google Pay catre inițiatorul facturii.

## Serverul NodeJS

Am avut nevoie să creez un server care sa realizeze operații de tip CRUD asupra bazei de date unde sunt stocate informații cum ar fi numele utilizatorilor, detaliile facturii (titlul, data, și un id unic), detalii despre plăți care vor avea asignate câte un id al unei persoane și un id al unei facturi urmat de numele produsului si prețul acestuia. Baza de date este realizată in MongoDB si este accesată prin intermediul serviciului cloud Mongo Atlass. Această bază de date este descrisă pe baza unor obiecte folosite drept modele în cadrul serverului NodeJS folosind librăria Mongoose.

Serverul este unul realizat in maniera REST unde sunt folosite URL-uri ce conțin rute care împreună cu tipul cererii HTTP sunt folosite pentru a face diverse operațiuni asupra bazei de date cum ar fi crearea de intrări noi, citirea si actualizarea celor deja existente și ștergerea acelor obiecte care nu mai sunt necesare aplicației.

Rutele alese de mine pentru serverul NodeJS sunt /persons și /bills peste care se pot realiza requesturi de tip GET, POST, UPDATE și DELETE pentru managementul obiectelor salvate in baza de date. La realizarea unui request de tip POST pentru ruta /persons este nevoie de un nume de utilizator de tip string care sa fie unic la nivelul bazei de date pentru a putea fi confirmată postarea. Pentru ruta /bills, este nevoie ca request-ul sa fie insoțit si de un nume de utilizator (cel al inițiatorului), un titlu, si o listă de produse si prețuri care sunt salvate in baza de date sub forma de payment-uri, separat de bill-uri, dar care sunt legate de factură printr-un id al bill-ului.

# Descrierea sumara a solutiei

Punctul central al proiectului este aplicația mobilă, motiv pentru care voi pleca de la lucruri care se întâmplă in aceasta si voi explica cum sunt folosite restul componentelor.

Workflow-ul principal al aplicației este cel în care utilizatorul adaugă o factură noua. Considerând că acesta este deja autentificat, din Home Page acesta va putea începe prin apăsarea butonului “Add new bill”. După apăsarea lui, utilizatorul este trecut printr-un proces de selectare sau creare a unei poze care să conțină datele înscrise pe nota de plată. În urma obținerii pozei, aceasta este trimisă către API-ul Google Vision prin clientul acestui serviciu din cadrul aplicației. Când este primit răspunsul de la serviciu, acesta este procesat pentru a se obține lista de produse și prețuri care va fi și afișată în pagina Edit Results. Această pagină are drept scop confimarea din partea inițiatorului plații că datele produselor sunt în conformitate cu ce este printat și pe bonul fizic. În urma confirmării datelor și după introducerea unui titlu sugestiv pentru această plată, o noua intrare va fi creată în baza de date prin accesarea serviciului aplicației pentru a stoca datele facturii. De asemeni, inițiatorul este trimis către o noua pagină unde îi va fi afișat un “id” al facturii generat unic și o listă cu persoanele care s-au alăturat plații prin introducerea id-ului din aplicația de pe device-ul lor. După confirmarea persoanelor, toți cei alăturați, vor primi de la serviciul aplicației lista de produse din care își pot selecta produsele consumate de ei și pot finaliza selectarea cu plata totalului, prin intermediul serviciului Google Pay, către inițiatorul plații. După ce inițiatorul termină de selectat produsele sale, va naviga către o pagina în care îi este afișat statusul pe care îl au ceilalți utilizatori. După ce toate persoanele au confirmat plata către inițiator, acesta va putea apăsa butonul de Google Pay din pagina de status care în urma contactului cu un dispozitiv POS oferit de personalul localului va realiza plata facturii.

|  |
| --- |
| Fig 1 – Diagramă master și workflow |

# Descrierea problemei

Problema pe care încerc să o rezolv odată cu realizarea acestui proiect este lipsa unei aplicații complete care sa realizeze plata notelor din restaurante sau localuri de la citirea bonului până la selectarea produselor pentru fiecare persoana și plata proproiu-zisă a consumației.

## Necesitate

Pentru a afla mai mult informații despre părera potențialilor utilizatori ai aplicației am relizat un chestionar care a ajuns la cca. 100 de persoane cuprinse între vârsta de 18 si 22 de ani, majoritatea fiind studenți. În Fig 2 poate fi observată frecventa ieșirilor, în decurs de o săptămână, cu cel puțin 4 persoane, acesta fiind genul de ieșiri pe care le targetez cu acest proiect.

|  |
| --- |
| Fig 2 - Statistica numarului de ieșiri în decurs de o săptămână cu cel puțin 4 persoane. |
|  |

## Dificultatea problemei

De asemeni în Fig 3 se poate observa cum părerile despre dificultatea momentului de calculare si adunare a sumelor de bani pentru plata notei sunt foarte împrăștiate dealungul intervalului de notare media fiind în jurul valorii de 5.

Considerând ca persoanele nu sunt neapărat deranjate de procedeul standard dar nici nu sunt încântate, se poate specula că o soluție care se ridică la nivelul așteptărilor ar fi bine venită în rândul acestora.

|  |
| --- |
| Fig 3 - Statistica dificultații (pe sacara 1-10) calculării si adunării sumelor de bani pentru plată. |

## Susținerea inițiativei

În Fig 4 se pot observa rezultatele notării pe o scară de la 1 la 10 a interesului în utilizarea unei aplicații care rezolvă problema lipsei unei aplicații care să gazduiască întregul proces de la primirea notei până la plata ei. Probabil numărul mare de note de 10 poate fi explicat pe fondul faptului ca majoritatea celor chestionați sunt studenți care ar fi mai deschiși spre a încerca orice fel de aplicație noua care are la bază rezolvarea unei probleme cu care se întâlnesc săptămânal.

|  |
| --- |
| Fig 4 - Statistica interesului în utilizarea aplicației |

## Alte implementări

Din cercetările făcute prin Google Play Store, am reușit să realizez un top de 5 aplicații care încearcă sa rezolve aceași problemă cu cea pe care încerc eu să o rezolv cu proiectul meu. Ordinea aplicațiilor în funcție de numărul total al descărcărilor este Splitwise [[5]](#footnote-5)(5M+), Revolut [[6]](#footnote-6)(1M+), Splid [[7]](#footnote-7)(100K+), Blitter [[8]](#footnote-8)(10K+) și Snap & Split [[9]](#footnote-9)(5K+).

De asemeni, dintre abordările problemei alese să fie implementate în aceste aplicații se disting 3 modalități. Voi vorbi despre ele în funcție de cât de sus în top se află aplicațiile ce implementează acea modalitate.

Prima modalitate de a rezolva această problemă și care se pare că are cel mai mare success este întâlnită în aplicațiile Splitwise și Splid cu mici diferențe. Aceste aplicații funcționează ca niște carnete de notițe în care se introduc manual sumele de plătit care apoi sunt asignate persoanelor. Diferența dintre aplicații este că în aplicația Splitwise toți cei implicați în plată au acces la informații, pe când în Splid numai inițiatorul poate vizualiza informațiile.

A doua modalitate este cea implementată de cei de la Revolut, o aplicație bancară, care adaugă la prima modalitate realizarea propriu-zisă a transferurilor din cadrul aplicației către persoana inițiatoare care poate plăti dupa ce primește suma de plată de la fiecare. Ceea ce îi lipsește este faptul că calculele trebuiesc făcute de către utilizatori si inserate sumele pentru transfer în aplicație deoarece aplicația nu are nici o metoda de recunoaștere a informațiilor de pe bonuri.

A treia metodă folosită este cea în care aplicațiile au la începutul workflow-ului pasul în care se realizează o fotografie a bonului pentru ca apoi să fie procesată și să se obțină o listă cu produsele si prețurile de pe bon. Aplicațiile care implementează aceasta metodă sunt Blitter, Snap & Split Bill. Folosind acest pas adițional la început este salvat timpul care pentru restul aplicațiilor este irosit pentru calcularea sumelor și introducerea lor în aplicație. Ceea ce le lipsește acestor aplicații este faptul ca nu au o metodă de plată a notei din interiorul aplicației motiv pentru care ele se diferențiază de prima metodă doar prin procesul de introducere a datelor prin fotografiere.

# Descrierea solutiei

În acest capitol vă voi prezenta și explica în detaliu cum a fost realizat proiectul luând pe rând fiecare componentă aplicație, server, baza de date și mondul în care acceste componente comunică între ele.

## Aplicația Android

**Interfața aplicație** am încercat sa o păstrez cât mai simplistă pentru a fi cât mai intuitivă. Pentru paginile principale (Home, Bills, Payments și Settings) am implementat o bară de navigare ce ajută la trecerea dintr-o pagină in alta printr-un mod customizat pentru aplicația noastră.

Pagina home, prezentată în Fig 5, este implementată prin clasa MainActivity care afisează un layout compus din bara de navigare în partea de jos și doua butoane în centru. Aceste butoane ajută la pornirea unuia dintre următoarele workflow-uri: cel în care utilizatorul este inițiatorul unei plăți noi (prin intermediul butonului “Add new bill”) și cel în care utilizatorul nostru vrea să se atașeze unei plăți deja inițiate (prin intermediul butonului “Join bill”) de una dintre persoanele de la masă.

|  |
| --- |
| Fig 5 - Home page |

Workflow-ul pentru alăturarea la o plată deja inițiată de un alt utilizator al aplicației, care poate fi observat în Fig 6, se face din pagina Home prin apăsarea butonului “Join bill”. După apăsarea butonului utilizatorul este trimis către o pagină unde va fi rugat sa introducă id-ul facturii generat și afișat imediat după confirmarea corectitudinii listei de produse, de către inițiatorul plații, pe dispozitivul acestuia. După apăsarea butonului “Join” va fi realizată o verificare a id-ului iar după confirmarea acestuia utilizatorul va fi directionat către o noua pagină în care va avea afișată lista de produse dintre care va urma sa le selecteze pe cele consumate de el prin atingerea numelui sau prețului și schimbarea culorii field-ului cu aceste informații în culoarea verde. În timp real, în partea de jos a acestei pagini poate fi observat un contor al sumei totale pe care utilizatorul va urma să o platească, care variază în funcție de ce produse din listă au fost selectate. Când utilizatorul consideră că a terminat de selectat produsele sale, în urma apasării butonului pentru Google Pay aflat în partea dreaptă a sumei totale de plată, acesta va fi redirecționat către o pagină în care poate vizualiza datele pentru plata finală cum ar fi suma totală și beneficiarul. După apăsarea butonului de confirmare a plății se va realiza o cerere de transfer de la utilizator către inițiator și va fi afițat daca a avut loc cu succes sau daca nu a fi putut realizată tranzacția. Dacă tranzacția are succes, treaba acestui utilizator se termină aici și va fi direcționat către pagina “Home”. Mai multe informații despre plată vor putea fi vizualizate în pagina “Payments”.

|  |
| --- |
| Fig 6- Join bill workflow |

Principalul workflow, cel în care unul dintre utlizatori joacă rolul de inițiator al plății, începe tot de pe pagina “Home”, după cum poate fi observat si în Fig 7. După apăsarea butonului “Add new bill” de pe această pagină, utilizatorul este direcționat către o pagină unde va avea pus la dispoziție o listă de opțiuni prin care poate crea sau selecta o poză care să conțină nota de plată (aplicații de cameră sau galerii). După selectarea uneia dintre opțiuni, respectiva aplicație va fi deschisă iar utlizatorul va trebuii să facă o poză bonului (în cazul în care a ales să folosească o aplicație pentru cameră) sau să o selecteze (în cazul în care a ales o aplicație galerie). După selectarea pozei, următorul pas este ca utilizatorul să decupeze spațiul din poză unde sunt numele și prețurile produselor. După decuparea pozei, aceasta este trimisă spre procesare la serviciul Google Vision care va răspunde cu textul din poză. Apoi textul din poză este procesat pe dispozitiv pentru asocierea numelor de produse cu prețurile acetora. Cât timp au loc toate aceste procesări, utilizatorul va aștepta cu aplicația adusă intr-o stare de “loading”.

|  |
| --- |
| Fig 7 - Take photo step |

După ce se termină procesarea datelor primite de la serviciul Google Vision, lista de produse este afișată în pagina “Edit results”, care poate fi observata în Fig 8. Acestă listă de produse și prețuri nu este definitivă datorită faptului că acuratețea de recunoaștere a textului din imagine nu este 100%. Din acest motiv utilizatorul are posibiltatea să modifice numele și prețurile produselor acolo unde este cazul. De asemeni utilizatorul, care este și inițiatorul plații, va trebuii să introducă și un nume sugestiv pentru acestă factură, nume care va fi folosit ulterior pentru identificarea unică și pentru generarea unui cod care să fie folosit de către restul persoanelor de la masă pentru a se alătura plații. După ce inițiatorul termină de corectat lista de nume si prețuri ale produselor, va naviga către urmatoare pagină “Add persons page” unde este afișat codul pe care îl vor folosi celelalte persoane pentru a se alătura. Aici este afișată și o listă în care apar persoanele care s-au alăturat la acea plată până în momentul de față. De aici se poate naviga către pagina unde inițiatorul își va selecta produsele consumate de el, dar spre deosebire de restul persoanelor, el nu va avea afișat un buton de Google Pay, ci va putea naviga către o altă pagină de unde poate vedea statusul persoanelor care sunt alăturate plații. Cei care au terminat de selectat, și au realizat transferul către acesta, vor apărea cu un tic verde în partea dreaptă al username-ului lor iar restul vor apărea cu o iconiță cum că sunt în așteptarea finalizării selecției sau a transferului.

|  |
| --- |
| Fig 8 - Edit results and add persons steps |

În această pagină există două opțiuni pentru finalizarea plătii: cash sau Google Pay. După ce toate persoanele adăugate confirmă plata către inițiator, acesta va putea selecta opțiunea de plata prin Google pay, pana atunci acest buton fiind dezactivat. Opțiune de plată prin Google Play implică activarea funcției NFC-ului (în cazul în care dipozitivul este dotat cu așa ceva) iar apoi transferul va fi realizat la apropierea telefonului de un aparat POS oferit de către personalul localului. Dacă transferul are loc cu succes va fi afișat un răspuns de confirmare iar apoi utlizatorul va fi direcționat către pagina "Home". Mai multe detalii despre plata facturii vor putea fi vizualizate in pagina "Bills".

|  |
| --- |
| Fig 9 - Log in and log out pages |

Pentru realizarea celor două workflow-uri descrise, utilizatorul va trebuii să fie logat cu un nume de utilizator. Acest nume trebuie să fie unic pentru a nu fi utlizatori cu același nume în detaliile facturilor și să apară confuzii. El este folosit pentru a stoca datele specifice fiecarui utilizator iar atunci când utilizatorul se deloghează informațiile sale vor fi șterse. După cum se poate observa și din Fig 9, logarea cu un nume nou de utlizator se poate face din pagina de settings unde este afițsat un spațiu, în care poate fi introdus un nume de utilizator, și un buton “Log in” care după apasare va trimite o cerere de postare a noului nume de utlizator la server. Acesta va trimite un răspuns care dacă este afirmativ atunci va fi afișat un mesaj precum logarea a avut loc cu success, dacă este unul negativ va fi afișat un mesaj precum utilizatorul trebuie sa încerce un alt nume de utlizator.

După cum am specificat de câteva ori mai sus, în bara de navigare există și opțiunea de navigare către pagina “Bills”. În această pagină, după cum poate fi observat și în FIGURA BILLS, utilizatorul poate vizualiza lista de facturi adăugate de el cu câteva detalii cum ar fi titlul, data la care a fost realizată factura și numarul de persoane. Acestă listă de facturi este interactivă și la apăsarea unei linii din listă utilizatorul va fi navigat către o noua pagină unde pot fi vizualizate mai multe detalii cum ar fi, lista de persoane (unde fiecare persoană are asociată câte o culoare), lista de produse (colorate în funcție de ce persoana a selectat produsul) și statusul plaților catre utilizatorul curent care este și inițiatorul plații. Dacă nota nu a fost finalizată, în partea de jos va apărea și butonul de finalizare a plații.

|  |
| --- |
| Fig 10 - Bills page |

Pagina de “Payments”, care de asemeni poate fi gasită în bara de navigare, este asemănătoare cu cea de “Bills” doar că aici pot fi vizualizate plățile către alți utilizatori ai aplicației în urma alăturării la o factura creată de aceaștia. După cum se observă și în FIGURA PAYMENTS, dinnou avem o listă de plați realizate sau care așteaptă să fie finalizate, pentru care afișăm titlul facturii, data la care a fost creată și numele utilizatorul care a inițiat plata. După ce una dintre payment-uri este selectată, utilizatorul este trimis către o pagină unde pot fi vizualizate o listă de persoane și o listă de produse cu prețuri. În cazul în care payment-ul selectat nu a fost finalizat, utiliazatorul va avea posibilitatea să selecteze produsele consumate de el iar apoi să realizeze plata către inițiatorul facturii prin apasarea butonului Google Pay.

|  |
| --- |
| Fig 11 - Payments page |

**Comunicarea cu serverul** este realizată prin intermediul pachetului ServerAPI din aplicația android care se folosește de diverse unelte ale librăriei Retrofit. În principal este vorba despre o interfață care descrie toate tipurile de requesturi care pot fi trimise către serviciu, sub formă de metode care au adnotat deasupra lor tipul requestului (GET, POST, UPDATE DELETE) urmat de un string ce descrie path-ul sub format REST la care să aibe loc requestul. Ca răspuns au un obiect de tip Call (din libraria Retrofit) care conține obiecte sau colecții de obiecte care urmează a fi primite ca răspuns. De asemeni metoda poate primi și parametri care pot fi simpli sau pot fi adăugați în interiorul path-ului adnotat. Toate aceste detalii pot fi vizualizate în Fig 10.

|  |
| --- |
| Fig 12 - Descrierea requesturilor în interfața clientului |

Această interfață este folosită în clasa “Client Builder” care pune în practică pattern-ul singleton și instanțiază o clasă care implementează interfața serviciului și are grijă ca doar o instanță a clientului să fie creată. După crearea unei instanțe, în momentul când este nevoie să se realize un anumit tip de request se va crea o instanță a uneia dintre clasele care construiesc acel tip request, cum ar fi pentru postarea, ștergerea și verificarea existenței unei persoane, sau pentru postarea de facturi impreuna cu selecțiile de produse realizate din aplicație.

Pentru a putea fi incluse în requesturi sau în raspunsuri, am creat un set de obiecte care descriu persoanele, facturile și selecțiile. Acestea sunt construite din aplicație și trimise către serviciu sau sunt folosite pentru a despacheta Json-urile primite ca răspuns.

**Serviciile externe** folosite sunt Google Vision și Google Pay. Pentru comunicarea cu aceste servicii am creat câte o clasă pentru fiecare dintre ele și le-am inclus în pachetul services.

După ce utilizatorul care inițiază plata a făcut poză bonului și a decupat spațiul din poza unde sunt regăsite produsele și prețurile acestora, poza este stocată sub forma unui bitmap și transmisă către o metodă a clasei Google Vision, care se ocupă de serviciul cu același nume. Această metoda va instanția un detector de text căruia ii va pasa bitmapul imaginii și va iniția procesul de recunoaștere a textului. Astfel imaginea este trimisă către cloud-ul google unde are loc procesarea imaginii și se construieste un răspuns din care putem extrage blocurile, liniile și elementele individuale din text. Pentru fiecare dintre acestea avem la dispoziție și coordonatele punctelor ce încadrează elementele într-un drepunghi.

Când sosește răspunsul de la serviciul cloud va fi pasat către o rețea care clasifică textul și apoi pe baza clasificarii elimină liniile care nu ne interesează, cum ar fi cele în care sunt menționate detalii despre cantitate și numărul de produse, și păstrează doar liniile în care avem numele produsului și liniile în care avem prețul acestuia. Apoi, folosind produsele si prețurile rămase vom rula un algoritm de asignare a acestora folosind informațiile despre coordonatele punctelor ce încadrează textul în imagine. Realizăm acest lucru prin sortarea liniilor de text iar apoi dacă primul element este un nume de produs vom selecta pentru acesta prețul aflat între numele curent și următorul nume și viceversa daca primul element este un preț. Astfel se poate obține o listă de produse si prețuri asignate care sunt foarte aproape de ce este afișat pe bon. Datoriă faptului că nu putem fi 100% siguri că rezultatele coincid cu realitatea este nevoie de o intervenție umană care să verifice corectitudinea rezultatelor obținute.

Serviciul de Google Pay va fi folosit in doua momente din workflow-ul aplicației pentru a realiza transferuri în siguranță și cu expunere minimă de date.

Prima dată, serviciul, va fi folosit de către cei care se alătură la plata notei. După ce aceștia își vor selecta produsele consumate și doresc să finalizeze partea lor de plată, vor putea utiliza butonul de Google Pay, care este afișat in partea dreapta jos cu sigla specifică pe el după cum poate fi observat și în Fig 6, care la momentul apăsării va declanșa un transfer din contul bancar atașat la contul de google către contul bancar al inițiatorului notei de plată.

Cel de al doilea moment când va fi folosit serviciul de plată de la google este atunci când inițiatorul plății primește confirmarea de plată de la toți cei care s-au alăturat plații și îi revine responsabilitatea de a face plata finală către localul în care a avut loc consumația. Pentru acest lucru, inițiatorul va trebuii să activeze funcția NFC a dispozitivului, să apese pe butonul Google Pay din pagina de status al notei de plată și să apropie spatele dispozitivului său (acesta fiind locul unde este montată funcția NFC de obicei) de un dispozitiv POS oferit de personalul localului. După ce este finalizată tranzacția va fi afișat un pop-up care să confirme starea finală a tranzacției.

## Server

Pentru realizarea serverului am ales NodeJS datorită faptului că este scalabil și fiindcă poate suporta foarte multe cereri asincron. De asemeni, NodeJS se bucură de o comunitate foarte mare de dezvoltatori de servere dintre care majoritatea le implementează în maniera REST (printre care mă numar și eu).

Serverul este hostat prin intermediul modulului ngrok[[10]](#footnote-10). Acesta facilitează procesul de comunicare cu dispozitivele mobile prin faptul că ai nevoie doar de lucruri pentru a te conecta la un serviciu hostat cu ngrok și anume URL-ul serviciului (generat random la fiecare schimbare în codul serviciului) și access la internet orice sursă de internet fiind foarte aproape de eliberarea unui server în producție.

Pentru a putea implemeta serviciul în maniera REST am folosit modulul express[[11]](#footnote-11). Și anume, prin intermediul modulului se creează un obiect ce va reprezenta serviciul și va încărca toate rutele care pot fi accesate, în cazul nostru rutele persons, bills și payments, după cum poate fi observat și în Fig 11.

|  |
| --- |
| Fig 13 - Routes loading |

De asemeni fiecare dintre aceste rute, conține sub-rute care reprezintă diverse operații (CRUD) care pot fi realizate asupra obiectelor de tip persons/bills. Aceste sub-rute pot chiar primi parametri cum ar fi numele unei persoane, sau id-uri ale bill-urilor sau payment-urilor.

În comunicarea serverului cu baza de date, acesta are doua roluri, primul rol este acela de a descrie obiectele folosite în baza de date pentru a avea un model după care să se realizeze trimiterea interogărilor către Mongo Atlas. Cel de al doilea rol este cel de a trimite interogări de tip CRUD, pentru a crea, citi, modifica și șterge intrările din baza de date. Pentru ambele roluri am folosit modulul mongoose [[12]](#footnote-12) care facilitează conexiunea, modelarea și realizarea de operațiuni asupra bazelor de date de tip MongoDB. Astfel, conexiunea a fost realizată prin autentificarea cu un tocken oferit la crearea nodului în cloudul Mongo Atlas, au fost create obiecte care descriu înregistrările din tabele (person, bill, payment) și se realizează operațiuni asupra datelor prin intermediul funcțiilor predefinite care primesc ca parametrii obiecte de tip JSON. În Fig 12 putem observa un model realizat pentru intrările în tabela Bills.

|  |
| --- |
| Fig 14 - Bill model |

Pentru transmiterea datelor între server și clienți, au fost folosite obiecte de tip JSON. Din acest motiv, pentru parsarea cererilor, am folosit modulul body-parser[[13]](#footnote-13) care funcționează ca un middleware între intrarea în server și funcțiile de procesare al cererilor.

## Baza de date

Pentru realizarea bazei de date am ales MongoDB și pentru postarea ei în cloud am ales soluția Mongo Atlas. Structura acestei baze de date poate fi observată în Fig 11, este formată din trei tabele Persons, Bills și Payments.

|  |
| --- |
| Fig 15 - Database diagram |

Tabela Persons este o tabelă de sine stătătoare deoarece ea nu are nevoie de legături cu alte tabele pentru a descrie înregistrările sale. Ea seste folosită pentru a stoca informațiile persoanelor ce utilizează aplicația. În starea curentă, această tabelă stochează numele de utlizator pe care persoanele conectate îl folosesc pentru a se identifica la împărțirea produselor.

Tabela Bills este folosită pentru stocarea metadatelor legate de notele de plată. Ea are ca și câmpuri un id generat automat care să identifice unic respectiva notă de plată, un titlu sub formă de string care este dat sugestiv de către inițiatorul plații și de asemeni data la care a fost creată factură. Cum pentru o factură exită doar un singur inițiator, acesta este identificat în cadrul tabelei prin cheia străina “initiator\_id” care reprezintă unul dintre id-urile stocate în tabela Persons. Aceste metadate sunt folosite pentru o mai ușoară identificare a facturii în cadrul paginii Bills din aplicația android, asfel putând fi vazut un preview cu titlul și data plații.

Tabela Payments este folosită pentru stocarea plaților individuale pentru fiecare produs în parte. Astfel, în această tabelă avem un field “bill\_id” care leagă o plată a unui produs de o factură. Pentru o factură unica, evident, pot exista mai multe intrări în tabela Payments care să conțină același “bill\_id”. De asemeni, mai există și un câmp care leagă plata unui produs de o anumită persoană prin cheia străină “person\_id”, acest id reprezentând persoana care a realizat plata produsului descris în câmpurile “name” și “price”. Câmpul “person\_id” poate fi și setat ca null dar doar în intervalul de timp în care persoanele de la masă își împart produsele. Odată cu confirmarea plății facturii, toate campurile din această tabelă care sunt legate de factură trebuie să aibe câte o persoana asociată. Practic, scopul acestei tabele este să eficientizeze baza de date prin nerepetarea informațiilor aferente facturii (metadate) pentru fiecare produs care face parte din ea. Astfel, inregistrările din tabela payments pot fi considerate ca fiind obiecte componente ale obiectului mare ce reprezintă factura (o înregistrare în tabela Bills).

# Rezultate

Funcționalitățile aplicației au fost testate folosind emulatorul de device-uri android din IDE-ul Android Studio și utilizând două dispozitive fizice, un LG G6 cu versiunea de android 8.0 (API level 26) iar celălalt un Samsung S5 cu versiunea de android 6.0 (API level 21).

Acum voi descrie maniera prin care au avut loc testele. Datorită faptului că worflow-ul principal (cel prin care se adaugă o noua factura) conține o sumă de 5 pagini diferite care au nevoie de informații de la pagina anterioară, în activitatea principală (MainActivity) inițializam obiectele de care respectiva pagină avea nevoie pentru a funcționa și mutam implementarea acelei pagini în pagina pricipală pentru ca apoi la rularea aplicației aceasta să apară prima dată. Astfel funcționalitatea paginii este testă intr-un mediu izolat, analizându-se rezultatele obținute pentru input-ul creat ca și cum ar fi primit de la pagina anterioară din workflow. Dacă în urma interacționării cu pagina, montată în acest fel, observam că obțineam comporamentul așteptat, consideram ca acea pagină a trecut testul de funcționalitate. Pentru a verifica diferite cazuri cum ar fi cele în care nu avem elemente de afișat în listă, sau nu s-au primit produsele și prețurile, etc, am pregătit mai multe seturi de date de intrare pentru pagini care să acopere și aceste cazuri.

După ce funcționalitățile individuale ale paginilor au fost testate, pasul următor a fost să tesetez integrarea lor. Astfel acestea au fost montate secvențial, fiecare în workflow-ul din care face parte, pentru a se putea observa dacă fiecare pagină este capabilă sa preia într-un mod corect informațiile pasate de pagina anterioară și dacă informațiile generate de pagina curentă pot fi transmise mai departe către următoarea pagină din workflow.

Pentru testarea serviciului Google Vision, am folosit un set de bonuri strânse pe parcursul dezvoltării aplicației. Din pagina Home selectam adaugarea unei facturi noi, iar când apărea selectarea unui opțiuni alegeam camera si făceam poze din diferite unghiuri și la diferite surse de lumină. Apoi pozele erau trimise la Google Vision API și așteptam rezultatele care urmau a fi afișate inițial în consolă. Cele mai bune rezultate apăreau atunci când bonul era relativ întins pe o suprafață dreaptă, poza era realizată dintr-o poziție cât mai aproape de planul paralele cu bonul astfel încat spațiul cu lista de produse să fie încadrat în totalitate iar poza să fie realizată cu blitz-ul telefonului activat. Apoi la pasul de decuparea a pozei să rămână doar lista de produse și fundalul alb al bonului.

# POZA GOOGLE VISION – IN STANGA POZA SI IN DREAPTA REZULTATELE

După obținearea rezultatelor de la Google Vision, aceastea sunt trecute printr-un algoritm de sortarea a liniilor pe baza pozițiilor lor în poză pentru ca apoi să fie clasificate pe linii de nume de produse și linii de prețuri pentru ca în final prețurile să fie asignate produselor pe baza modului în care sunt intercalate prețurile și numele de produse pe bon. Aici cu cât bonul urmează un afișaj mai comun cu atât sunt șanse cât mai mari să obținem rezultate bune. Din testele realizate cu bonurile adunate de mine, rezultate bune se obțin pe acele bonuri care au separate prețurile și numele în coloane diferite care au un spațiu între ele pentru ca Google Vision să le poate identifica ca linii diferite sau cand au prețurile trecute pe linii diferite, înainte sau dupa linia cu numele produsului.

Clientul pentru serviciul aplicației, care este creat cu Retrofit, a fost testat pentru a se observa daca rezultatele cererilor coincid cu ce există deja în baza de date. În prima fază urmaream datele din baza de date, apoi construiam cererea din aplicația Postman. Dacă cererea constuită obținea datele dorite le implementam în clientul aplicației folosind libraria Retrofit. După câteva rulări ale respectivei cereri, în diverse situații, din care rezultau numai răspunsuri corecte consideram că acea metodă a clientului pentru realizarea cererii a fost realizată corect.

# IMAGINE REQUEST POSTMAN

Serviciul aplicației a fost testat pentru verificarea a trei funcționalități: realizarea de operații de tip CRUD asupra bazei de date și corectitudinea datelor trimise ca răspuns la cererile venite de la client.

Testarea operațiilor CRUD asupra bazei de date a fost realizată prin trimiterea mai multor feluri de cereri get, put, post delete care aveau un comportament așteptat. După trimiterea cererilor verificam din panoul de control al bazei de date Mongo Atlas dacă datele s-au schimbat în modul așteptat. Când rezultatele se conformau cu așteptările noastre consideram că operațiile aveau loc cu succes și lucrează corect.

Pentru fiecare tip de obiect din baza de date, serviciul implementează rute de tip REST. Pe baza acestor rute, serviciul cunoaște asupra căror obiecte se vor realiza operații. După hostarea și verificarea accesului asupra rutelor serviciului a urmat verificare faptului că răspunsurile formate după procesarea datelor obținute din baza de date are loc corect și se conformează așteptărilor. Spre exemplu, la cererea de tip GET a undei facturi pentru o anumită persoană, se extrag date din tabela BILLS, ca apoi cu id-ul facturii să se obțină selecțiile de produse făcute, din tabela PAYMENTS, iar obiectul json trimis ca răspuns este format din combinarea datelor obținute din cele doua tabele. Dacă obiectul este creat corect și structura lui este cea așteptată ca răspuns în clientul aplicației, se considera că obiectul este creat cu succes și este corect.

# Viitoare îmbunătățiri

# **Concluziile lucrării**

# Bibliografie

gIGEL. (fără an). *GOOGLEPEDIA*.

1. Github MongoDB - https://github.com/mongodb/mongo [↑](#footnote-ref-1)
2. OCR – Optical character recognition [↑](#footnote-ref-2)
3. POS -  Point of sale terminal [↑](#footnote-ref-3)
4. NFC – Neer field communication [↑](#footnote-ref-4)
5. Splitwise - <https://play.google.com/store/apps/details?id=com.Splitwise.SplitwiseMobile> [↑](#footnote-ref-5)
6. Revolut - <https://play.google.com/store/apps/details?id=com.revolut.revolut> [↑](#footnote-ref-6)
7. Splid - <https://play.google.com/store/apps/details?id=splid.teamturtle.com.splid> [↑](#footnote-ref-7)
8. Blitter - <https://play.google.com/store/apps/details?id=es.soutullo.blitter> [↑](#footnote-ref-8)
9. Snap & Split - <https://play.google.com/store/apps/details?id=com.astepanov.mobile.splitcheck> [↑](#footnote-ref-9)
10. Ngrok - https://ngrok.com [↑](#footnote-ref-10)
11. Express - https://expressjs.com [↑](#footnote-ref-11)
12. Mongoose - https://mongoosejs.com [↑](#footnote-ref-12)
13. Body-parser - https://www.npmjs.com/package/body-parser [↑](#footnote-ref-13)