Universitatea Tehnică de Construcții București

Facultatea de Hidrotehnică Domeniul Ingineria Sistemelor

Specializarea Automatică și Informatică Aplicată

**PROIECT DE DIPLOMĂ**

**Aplicatie de mobile pentru identificarea si rutarea statiilor electrice**

Coordonator: Absolvent:

Șl. dr. ing. Ionela Halcu Brinzaru Marius-Andrei

-2020-

CUPRINS

[CAPITOLUL 1. INTRODUCERE 3](#_Toc43227730)

[1.1. Alegerea temei 6](#_Toc43227731)

[1.2.Obiectivele lucrarii 9](#_Toc43227732)

[CAPITOLUL 2. Sistemul de operare Android 11](#_Toc43227733)

[2.1Platforma Hardware 11](#_Toc43227734)

[2.2Arhitectura MCP 12](#_Toc43227735)

[2.3 Sistemul de operare 14](#_Toc43227736)

[2.4 Rezumat Android 14](#_Toc43227737)

[CAPITOLUL 3. Aspecte teoretice 15](#_Toc43227738)

[3.1. Perspectiva Server-Side C# .NET CORE 15](#_Toc43227739)

[3.1.1 Clase si Tablouri 25](#_Toc43227740)

[3.1.2 Drepturi de acces 25](#_Toc43227741)

[3.1.3 Constructori si Destructori 26](#_Toc43227742)

[3.1.4 Moştenire 27](#_Toc43227743)

[3.1.5 Metode 28](#_Toc43227744)

[3.1.6 Proprietăţi 28](#_Toc43227745)

[3.1.7 Dependency Injection 29](#_Toc43227746)

[3.2 Perspectiva interfata utilizator IONIC + Angular 29](#_Toc43227747)

[3.3. Autentificare Auth0 30](#_Toc43227748)

[3.4. Baza de date MonogoDB 31](#_Toc43227749)

[3.5. Algoritmul de rutare Google Directions API 33](#_Toc43227750)

[3.6. Tehnologii auxiliare 37](#_Toc43227751)

[3.6.1Postman 37](#_Toc43227752)

[3.6.2 Visual Studio 38](#_Toc43227753)

[3.6.3MongoDB Compass 39](#_Toc43227754)

[3.6.4 Visual Studio Code 41](#_Toc43227755)

[3.6.5 Swagger Swashbuckle 42](#_Toc43227756)

[3.7. Rezultate obtinute 43](#_Toc43227757)

[CAPITOLUL 4. CONCLUZII SI DEZVOLTARI ULTERIOARE 43](#_Toc43227758)

[BIBLIOGRAFIE 43](#_Toc43227759)

[ANEXE 44](#_Toc43227760)

# CAPITOLUL 1. INTRODUCERE

Știința prelucrării informațiilor (datelor) cu ajutorul computerelor se numește informatică. Istoria informaticii începe cu mult timp înainte ca disciplina modernă a informaticii să apară în secolul al XX-lea. Progresul, de la invenții mecanice și teorii matematice până la mașini și concepte moderne, a format un domeniu major academic și baza unei industrii uriașe la nivel mondial.

Cel mai vechi instrument cunoscut pentru utilizarea în calcul este abacul. Se crede că a fost inventat în Babilon circa 2400 î.Hr.. Prima oară era folosit prin trasarea unor linii în nisip cu pietricele. Acesta a fost probabil primul computer și cel mai avansat sistem de calcul cunoscut din aceea perioadă precedând metoda elenă cu 2000 ani. Abacul este o tăbliță dreptunghiulară, folosită de oameni în antichitate pentru efectuarea calculelor. Abacele cu un design modern sunt încă folosite astăzi ca instrumente de calcul.

Dispozitivele mecanice analogice de calcul au apărut din nou o mie de ani mai târziu, în lumea islamică medievală. Exemple de dispozitive din aceasta perioada includ: equatoriumul lui Arzachel (equatoriumul este un instrument astronomic de calcul a pozițiilor aștrilor) sau astrolabul lui Al-Biruni (astrolabul este un instrument astronomic istoric folosit de astronomi, navigatori, și astrologi).

În 1623, Wilhelm Schickard a proiectat o mașină de calcul, dar a abandonat proiectul, deoarece clădirea în care se afla prototipul a fost distrusă de un incendiu în 1624.

În jurul anului 1640, Blaise Pascal, un cunoscut matematician francez, a construit primul dispozitiv mecanic bazate pe un design descris de matematicianul grec Heron din Alexandria.

Apoi, în 1672, Gottfried Wilhelm Leibniz a inventat Socotitorul în trepte care a fost finalizat în 1694. Nici unul dintre aceste dispozitive timpurii de calcul nu au fost calculatoare în sensul modern al cuvântului, și a fost nevoie de un progres considerabil în matematică înainte ca primele calculatoare moderne să poată fi concepute.

În secolul al VII-lea, matematicianul indian Brahmagupta a dat prima explicație a sistemului de numerație hinduso-arab și utilizarea lui zero atât ca substituent cât și ca cifră zecimală. (Un substituent este un element care poate înlocui un alt element, cu care are proprietăți asemănătoare.)

În 1703, Gottfried Leibnitz a dezvoltat logica într-un sens matematic formal, în scrierile sale despre sistemul de numerație binar. În sistemul său, valorile unu și zero reprezintă valorile adevărat și fals (true și false) sau pornit/oprit (on/off). Dar a fost nevoie de mai bine de un secol pentru ca George Boole să publice algebra booleană în 1854 cu un sistem complet care permite proceselor de calcul să fie modelate matematic. (Algebra booleană este o algebră formată din: elementele {0,1}; două operații binare numite SAU și SI, notate simbolic cu + sau Ú și × sau U; și o operație unară numită NU (negație), notată simbolic 0 sau O.

În perioada 1833-1835, Charles Babbage a inventat „mașina analitică”, care se baza direct pe cartelele perforate pentru programare ale lui Joseph Marie Jacquard. Această mașină era alimentată de un motor cu aburi. Babbage nu a reușit să o construiască până la moartea sa, din cauza limitărilor tehnologice ale vremii. Cu toate acestea, o mașină construită în 1991 după schițele sale s-a dovedit a funcționa perfect.

Un computer numit în limba română și calculator, calculator electronic sau ordinator (din franceză) este o mașină care prelucrează informații (date) pe baza unui program (o listă de instrucțiuni).

Calculatoarele sau computerele pot administra, memoriza, transmite și prelucra o mare cantitate de date într-un timp scurt. Pentru efectuarea unor astfel de operații este necesară o interacțiune complexă între partea fizică sau echipament (în engleză hardware) și partea logică sau programele soft care reprezintă domeniile fundamentale de cercetare în Informatică.

Termenul informatică desemnează știința procesării sistematice a informației, în special a procesării cu ajutorul calculatoarelor.Istoric, informatica s-a dezvoltat ca știință din matematică, în timp ce dezvoltarea primelor calculatoare își are originea în electrotehnică și telecomunicații. Marele avantaj al sistemelor computaționale constă în capacitatea lor de a prelucra în mod schematic cantități enorme de informații la o viteză foarte mare. S-a încercat și implementarea capacităților perceptive ale omului în sistemele informatice, însă până în prezent cu un succes foarte limitat. Un exemplu în această direcție îl constituie sistemele de recunoaștere a chipului uman, sau/și de luare a deciziilor atunci când nu se dispune de toate datele necesare. Astfel de procese sunt studiate de o ramură specializată a informaticii, inteligența artificială. Astfel, în anumite discipline restrânse pot fi obținute deja rezultate remarcabile. Totuși nu se poate încă vorbi despre o modelizare a inteligenței umane.

De-a lungul timpului, formele de comunicare și mijloacele folosite pentru a comunica s-au dezvoltat în funcție de progresul tehnologiei. Modul de comunicare al oamenilor a fost revoluționat de apariția calculatorului, internetului și a telefoniei inteligente. Acesta a reușit să spargă bariera de timp și spațiu, ajutându-i să interschimbe intr-un mod eficient informații. Aparția telefoanelor inteligente ( smartphones) a marcat momentul în care tehnologia a reuțit să conecteze oamenii oriunde și oricând.

În prezent, informatica își găsește aplicații în toate domeniile vieții. Prezența ei este puternic amplificată de impactul pe care îl are internetul. Rețeaua la nivel mondial a revoluționat comunicarea dintre companii, logistica, mass-media, dar și viața privată a fiecărui individ. Mai puțin vizibil, dar totuși omniprezent, informatica și-a câștigat un loc stabil până și în aparatele casnice. Printre altele, telefoanele inteligente, care sunt adevărate calculatoare, au permis democratizarea utilizării informatice pe scară extrem de largă.

Oamenii au necesitatea de a depăși orice prag în privința interrelaționării umane, iar acest lucru este vizibil în evouția vânzărilor e smartphone-uri. Evoluția vânzărilor se poate observa în figura următoare de mai jos.

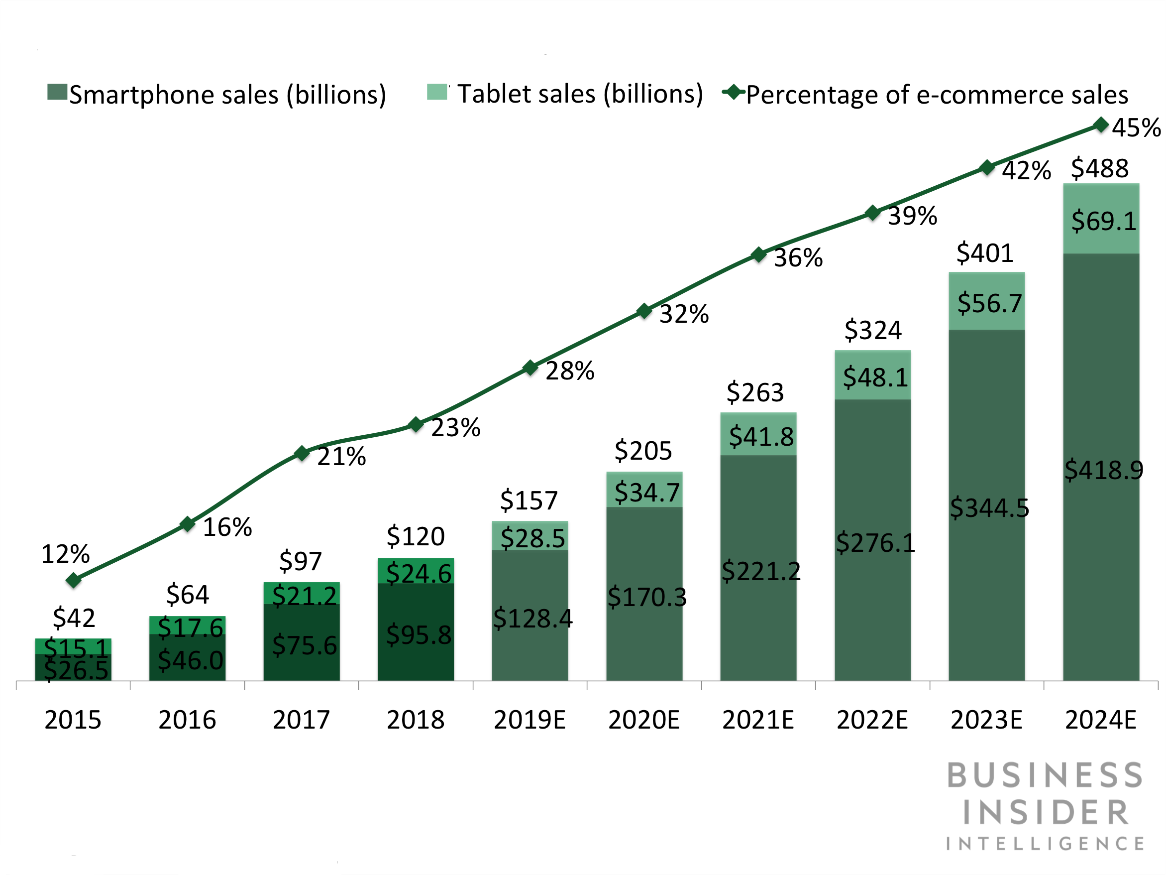


Figura 1.1. Evoluția vânzărilor de smartpone-uri. [1]

**1.1. Informații utile**

În programare, este adeseori dificil sau problematic în a alege o modalitate de implementare sau a gândi așezarea și mecanismele optime. Astfel apar diferite întrebări, decizii sau chiar conflicte în legătură cu alegerea făcută. Un dezvoltator trebuie să țină cont de foarte multe aspecte atât la nivel înalt, de exemplu proiectarea unei arhitecturi, cât și la nivelul unei decizii uzuale în elaborarea codului. Printre aceste puncte de interes, putem enumeră:

● Reuzabilitatea: ne ajută să identificăm conceptele de bază și să eliminăm redundanțele

● Intenția: ne permite să exprimăm intenția clară a componenței dezvoltate

● Testarea: se referă la modularizarea metodelor și la soluții optime pentru testare unitară

● Minimizarea codului: se referă la soluții optime pentru a atinge efectul dorit, reducând complexitatea

● Comunicarea: folosirea unor modalități sau metode ce pot fi ușor înțelese și preluate de un precursor al procesului de dezvoltare pe acelaș cod

● Mentenanță: impune folosirea unor construcții intuitive și deconectate

● Siguranță: impune încredere asupra modalitatatii de dezvoltare aleasă

Astfel, apare nevoia de refactorizare, confuzii, neînțelegeri și în general o perioada mai mare pentru dezvoltare și testare.

În întâmpinarea acestor probleme, cu timpul, au apărut diverse soluții ce rezolva aceste dificultăți în diferite contexte. Acestea, pot fi caracterizate mai degrabă că niște șabloane sau modalități de rezolvare decât că părți de cod predefinite sau algoritmi consacrați.

Denumirea sub care sunt cunoscute aceste practici este “Design Patterns” (în traducere “Șabloane de arhitectură”), fiind o unealtă importantă în programarea modernă. Ele ajută la îmbunătățirea procesului de dezvoltare prin a oferi soluții testate, documentate, de încredere și care respectă cerințele enumerate mai sus, pentru a servi în diverse contexte. Mai mult decât atât, sunt un limbaj comun pentru programatori, ajutând astfel la comunicarea și înțelegerea codului.

Pe lângă acestea, fiind un numitor comun între programatori în rezolvarea anumitor situații, aceste modalități sunt folosite, menținute și testate la nivel global de către dezvoltatori din toată lumea. Fiind atât de răspândite și utilizate, în timp, au fost aduse foarte multe îmbunătățiri și corecturi, în prezent ajungând la variante “șlefuite” și de încredere pentru a fi preluate. De asemenea, există că aspect negativ nivelul puțin mai ridicat de dificultate pe care unele dintre ele le prezintă. Deși este o problema reală, care poate încetinii puțin procesul de dezvoltare, o putem combate prin timpul salvat căutând alte soluții (care se pot dovedii ineficiente) dar chiar și prin caracterul “popular” pe care îl au, astfel există foarte multe resurse și explicații pe internet.

## 1.1. Alegerea temei

În prezent aplicațiile de telefon sunt în continuă ascensiune și vor domina piața pentru mult timp de acum încolo.Momentan nu exista nici o aplicație de telefon mobil, fie el Android sau iOS, care să iți permită vizualizarea unui traseu specific de parcurs pentru a ajunge exact la destinația în care se află stația de încarcare electrică.

Tema abordată în cadrul proiectului de diplomă se intitulează “Aplicație de mobil pentru identificarea și rutarea stațiilor electrice**”.**

Unul dintre motivele principale care a adus la alegerea ca temă de diplomă a fost crearea unui sistem informatic care să informeze utilizatorii aplicației de starea stațiilor electrice de încarcare a automobilelor aflate în împrejurimea utilizatorului.

Un alt motiv este determinat de studiul pe care l-am realizat asupra unor aplicații existente și care m-a determinat să creez o nouă aplicație care să înglobeze toate facilitățile de care are nevoie un utilizator.

Alt motiv ar fi acela ca industria auto se orientează din ce in ce mai mult spre modelele cu o propulsie hibridă sau doar electrică. Urmând o creștere accelerată a automobilelor și respective a stațiilor electrice ne confruntăm cu o problemă de indentificare a stațiilor disponibile din zona în care vrem să le accesăm.

O scurtă istorie a automobilelor hibride, adică a mașinilor care erau propulsate de motoare electrice ajutate de un motor cu ardere internă, ne duce până în zorii secolului al XX-lea, în anul 1901. În acel an, Ferdinand Porsche a proiectat automobilul Mixte, o versiune hibridă a mașinii electrice “System LohnerPorsche”, lansat cu un an înainte la Salonul Auto de la Paris. Mixte avea o autonomie de 50 de kilometri și atingea o viteză maximă de 50 km/h.

În figura următoare este prezentată prima mașină hibridă.



Figura 1.2. Prima mașină hibridă, Mixte anul 1901. [2]

În momentul de față există aplicații web care indică poziția stațiilor electrice numită “PlugShare”. Rețeaua de încărcare pentru mașini electrice din România se îndreaptă cu pași mari spre ieșirea din faza de pionierat. În toate orașele importante din țară există cel puțin câteva puncte de încărcare de putere medie, adică 22 Kw, și minimum o stație rapidă , 50 Kw. Prețul pentru 1 kwh este de 0,5 lei, în medie pentru un consum casnic. Timpul de încărcare pentru o mașină electrică a anului 2019, cu o capacitate de 40 kWh, poate dura între o oră și 12 ore, în funcție de capacitatea bateriei, puterea prizei/stației și puterea încărcătorului.

În 1917, compania Woods Motor Vehicle din Chicago a produs un automobil hibrid al cărui motor electric îl propulsa până la 25 km/h. Până în 1918 au ieșit de pe linia de fabricație 600 de astfel de produse, dar mașina nu a avut succes deoarece era prea lentă, scumpă și greu de reparat.Istoria maşinilor electrice este una fascinantă, iar evoluţia acestui tip de automobile a fost una spectaculoasă, cu multe urcuşuri şi coborâşuri în preferinţele publicului. Dacă în anii 1900 aceste maşini dominau piaţa, în 1935 reprezentau deja o raritate, iar producţia lor la scară largă s-a făcut abia 90 de ani mai târziu.Marii constructori de automobile, inclusiv General Motors, Volkswagen, Daimler AG, nu neglijează acest sector, în prezent fiind alocate fonduri consistente pentru a dezvolta vehicule electrice ce pot satisface chiar şi cele mai rafinate gusturi. Tesla s-a axat în special pe dezvoltarea bateriilor care alimentează autoturismele electrice. Primul automobil electric care a stârnit pasiuni a fost conceput la sfârşitul anilor 1800 şi începutul anilor 1900. În 899 şi 1900, vehiculele electrice surclasau toate celelalte tipuri de maşini. În fapt, potrivit datelor statistice, 28% din cele 4.192 de maşini produse în Statele Unite erau electrice.Îngrijorarea legată de poluare a fost parţial responsabilă pentru interesul privind dezvoltarea tehnologiei pentru automobile electrice. Toyota Prius a fost un alt model care a crescut interesul pentru maşinile cu consum redus de carburant. Primul model a fost produs în Japonia, în 1997. A devenit disponibil în întreaga lume în anul 2000. Prius a fost unul din primele vehicule hibrid produse în masă. Succesul a fost unul remarcabil, în primul an de la lansare compania vânzând 50.000 de astfel de maşini.Tayota Prius înregistrează recorduri peste recorduri în vânzări în SUA, Japonia și cresc constant în Europa de Vest. [3]

În următorii ani, modelele propuse de Nissan, General Motors şi Volkswagen au făcut ca numărul celor care aveau maşini electrice să fie de ordinul milioanelor, iar tendinţa este una de creştere. De altfel, în 2016 vor fi lansate pe piaţă nu mai puţin de şase modele noi de maşini electrice. Mai mult, un nou tip de motor electric, după cel oferit de Tesla, propus de Faraday Future, ar putea rula pe şoselele lumii până în 2020. Deşi compania nu a dat detalii despre gama exactă a vehiculelor sale, Nick Sampson - şeful deparamentului de dezvoltare - a precizat că bateriile vor avea o autonomie mult mai mare decât cele marca Tesla .

Așa-zisele „mașini verzi” sunt proiectate să producă niveluri reduse sau inexistente de gaze cu efect de sera, in special dioxid de carbon, metan și oxid nitros. Acest lucru se poate obține prin construirea unor mașini funcționale pe baza altor mijloace energetice decât combustibilii fosili, benzina sau motorina. Este vorba despre surse de energie alternativa si regenerabila, care, în privința mașinilor cel puțin, pot îmbracă forma de celule de combustie, bazate pe biocombustibil (etanol, butanol, metanol), hidrogen, alimentare fotovoltaica, baterii sau super capacitoare .

Primul punct de încărcare pentru mașini electrice din România a fost înaugurat în data de 22 iunie 2011, la sediul Transelectrica din Municiupiul București. Acest punct de încărcare era o priză de 230 V, 16 A, pozițioată pe stradă, loc de parcare dedicat și puteai să îți încarci un autovehicul electtic în acel loc.

În anul 2015 industria mașinilor electrice a avut un nou început, începând cu instalarea primei stații de mare putere, 50 de Kw, la sediul Porsche România din Pipera. În anul 2016 a urmat o altă stație similară, tot de 50 Kw, la sediul BMW Group și tot atunci deveneau funcționale și primele stații de 50 Kw plasate în parcările marilor centre comerciale. S-au făcur remarcați Kaufland/e-charge Renovatio (la nivel național) și Lidl (pe București).

Anii 2017 și 2018 au marcat dezvoltarea rețelei de încărcare pe bazele deja puse anterior, plus puncte de încărcare noi. În această perioadă au început să apară noi nume precum Polyfazer, Rotric, Romstal, Electromagnetica, Penny, Mega Image, EVconnect, OMV + Electrica, Rompetrol, Mol, ETU, MCM Petrol, Enel, E.ON, engie, Uber (Green), BCR (eGO), practic încep să se facă remarcate companii noi pe subiectul electromobilității, iar marile benzinării diversifică tipurile de carburant oferite, adăugând și stațiile de curent electric pe lângă cele de benzină și disel.

În anul 2019 au fost așteptate primele stații de putere foarte mare, adică 150 Kw, iar concomitent vor apărea primele mașini electrice ce pot fi încărcate la acceastă putere.

În cele mai multe locuri este gratuit să încarci o mașină electrică, iar cel mai scump preț pe unitate (1 unitate = 1 kWh) este de 1,95 Lei. Termenul de unitate apare ca echivalent pentru kWh din motive de legislație. Furnizorii de electricitate nu pot vinde curent electric către un automobil (adică spre un bun care se deplasează), ci doar către un punct fizic fix (clădire, stație). Așa că furnizorii s-au adaptat și au botezat kWh în unitate, să fie cât mai simplu de înțeles de către utilizatori. [4]

## 1.2.Obiectivele lucrarii

Lucrarea are ca obiectiv realizarea unei aplicații de mobile (Android) care sa pună la dispoziția utilizatorilor un serviciu de indetificare a stațiilor electrice pe o rază de aproximativ 400-500 m a utilizatorului, precum și o rută care să te conducă cu succes la stația cea mai apropiată și potrivită din punct de vedere al configurației vehiculului pe care utilizatorul dorește să îl încarce.

Android este un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google.

Android permite dezvoltatorilor să scrie un cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google. Lansarea platformei Android la 5 noiembrie 2007 a fost anunțată prin fondarea Open Handset Alliance, un consorțiu de 48 de companii de hardware, software și de telecomunicații, consacrat dezvoltării de standarde deschise pentru dispozitive mobile. Google a lansat cea mai mare parte a codului Android sub licența Apache, o licență de tip free-software și open source. Fiind o platformă destinată dezvoltării de aplicaţii pentru telefoane mobile, în realizarea ei s-a avut în vedere conservarea energiei acumulatorului precum şi gestionarea în mod eficient a resurselor de memorie existente.[5]

În figura următoare este prezentată evoluția pieței de telefonie mobilă între anii 2017-2023.

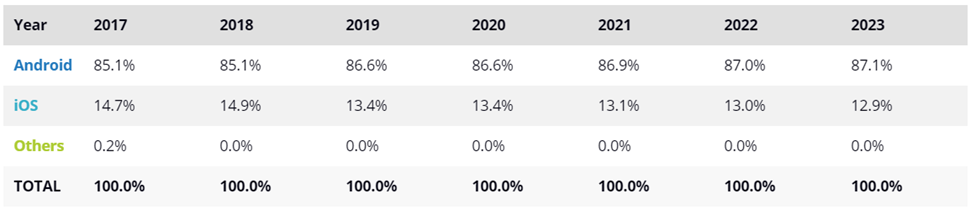


Figura 1.3. Evolutia pieței de telefoane mobile în următorii ani [6]

Android deține în prezent o pondere a pieței de 86% care este în continuă ascensiune pe piața telefoanelor și tabletelor. O analiză facută de IDC Corporate prezice urmatoarea evoluție a pieței de telefoane smart și tablete.

Problema abordată este aceea de a îndetifica stațiile electrice într-un mod cât mai accesibil si la îndemana oricui. Aproape oricine deține un smartphone care îl ajută în activitățile de zi cu zi de aceea o aplicație care să te ajute să îndetifici stațiile electrice mai ales dacă deții un automobil cu propulsie electrică.

Avantajul acestei metode de indentificare este faptul că este ușor accesibilă nu necesită cunoștințe avansate de utilizare a smartphone-urilor.În scopul evitării oricărei aglomerări asupra uneia sau mai multe stații.

Dezavantajul este că dacă nu este folosit de o mare majoritate de utilizatori pot apărea interferențe în cea ce privește starea actuală a stației.

Pentru realizarea lucrării de diplomă se va realiza o aplicație mobile pentru identificarea stațiilor de încărcare, pentru vehiculele electrice si hibride.

Un vehicul hibrid este un vehicul care are mai multe sisteme de propulsie, spre deosebire de vehiculele convenționale (cu motor cu ardere internă), cele electrice sau cele cu pile de combustie, care au câte un singur sistem de propulsie. Adesea, în special pentru autoturisme, termenul se folosește în sensul de vehicul electric hibrid adică pentru vehiculele echipate cu motoare cu ardere internă și cu motoare electrice. Însă sunt posibile și alte combinații: hidraulic hibrid (cu motor cu ardere internă și pompă/motor hidraulic), sau propulsat muscular și electric, de exemplu bicicletele electrice. În această lucrare se studiază istoria dezvoltării mijloacelor de transport hibrid.

În ceea ce privește numărul de mașini electrice pe șoselele din întreaga lume, această a crescut până la aproximativ 2 milioane în 2016, potrivit Agenției Internaționale pentru Energie. Guvernul Britanic s-a angajat că aproape toate autoturismele și autoutilitarele să fie cu emisii zero până în 2050.Că urmare, se estimează că cererea pentru automobile electrice va accelera dramatic în următoarele decenii. OPEC și-a revizuit prognoza la 266 milioane de mașini electrice până în 2040, aproape de șase ori mai mult decât estimarea anterioară de 46 milioane.[7]

Avantajele majore ale automobilului electric sunt consumul redus, poluare aproape de zero în afară producției, eficiență/randament foarte mare, accelerare instantă, zgomot foarte redus, întreținere ușoară prin eliminarea unei mari părți a motorului.

Dat fiind faptul că autoturismele electrice sunt în continuă ascensiune asta arată graficul făcut de cercetătorii de la IENE(Institutul de energie pentru sud-estul Europei).

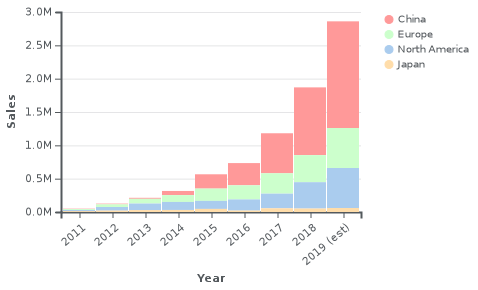


Figura 1.3. Evoluția vânzării de mașini electrice în următorii ani [8]

Ascensiunea statiilor electrice este direct proporțională cu piața autovehiculelor cu propulsie electrică și hibird. Au apărut mai multe standarde de putere electrică și mufe de încarcare specifice fiecărei mărci de autovehicule.

# CAPITOLUL 2. Sistemul de operare Android

Următoarele două subcapitole oferă o privire de ansamblu asupra platformei Android, atât din punct de vedere hardware, cât şi software şi au fost realizate utilizăndu-se ca referinţe: lucrarea Android Forensics - Investigation, Analysis, and Mobile Security for Google Android scrisă de Andrew Hoog şi publicată la editura Elsevier în 2011 [9], precum şi pagina web destinată programatorilor Android. Ultimul subcapitol descrie componenţa unei aplicaţii Android şi a fost elaborată consultând materialul Professional Android 2 Application Development scris de Reto Meier şi publicat la editura Wiley Publishing în 2010 [10].

## 2.1Platforma Hardware

Android a fost proiectat pentru a fi compatibil cu o gamă largă de componente hardware. Aceasta constituie o caracteristică importantă, întrucât permite producătorilor să conceapă şi să integreze elementele necesare dispozitivului Android, făcându-l atractiv atât în rândul programatorilor cât şi al consumatorilor. Fiind o platformă destinată dezvoltării de aplicaţii pentru telefoane mobile, în realizarea ei s-a avut în vedere conservarea energiei acumulatorului precum şi gestionarea în mod eficient a resurselor de memorie existente. Aşa cum apare în [9], majoritatea dispozitivelor Android au utilizat încă de la început procesoare ARM, cunoscute a fi destul de puternice, raportat la consumul mic de energie. Cu toate acestea, unele corporaţiile care au portat Android pe alte platforme au ales să folosească propriile lor CPUuri (ex: Intel a utilizat propriul lor procesor Atom). Pentru a nu suprasolicita procesorul, designerii de celulare au realizat o componentă separată care să gestioneze complexitatea comunicării mobile.

Modem-ul baseband şi radioul sunt sisteme hardware şi software care furnizează dispozitivelor Android o conexiune la telefonia mobilă, lucru ce permite atât transfer de voce, cât şi de date. După cum se poate vedea in Figura 2.1, la telefoanele inteligente întâlnim două tipuri de memorie: memorie cu acces aleator (RAM) şi memorie nevolatilă flash NAND. Conform [9], memoria RAM este folosită de sistem pentru încărcarea, executarea şi manipularea diferitelor părţi ale sistemului 12 de operare, iar cea flash pentru stocarea sistemului de operare precum şi a datelor dorite de utilizator. Datorită limitării impuse de spaţiul restrâns, cele două tipuri de memorie sunt combinate într-o singură componentă, cunoscută sub numele de pachet multicip (MCP).

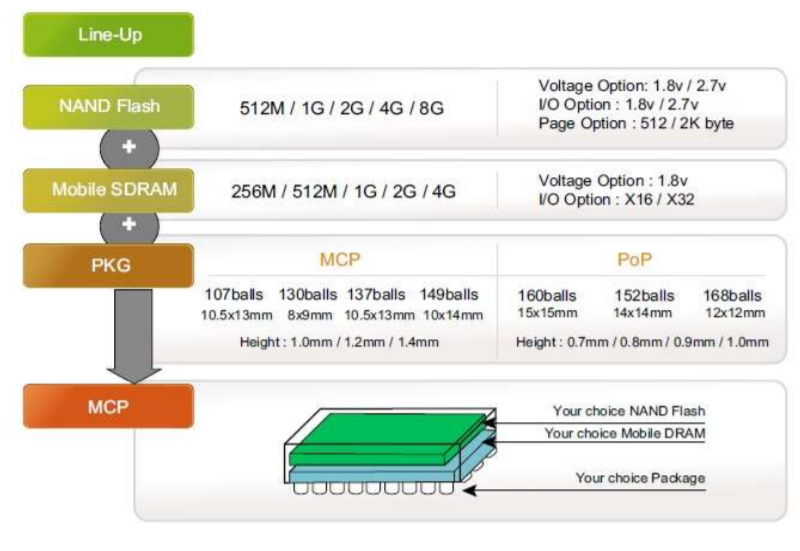


Figura 2.1. Arhitectura MCP [9]

## 2.2Arhitectura MCP

Arhitectura MCP Una dintre cele mai interesante componente hardware este sistemul de poziţionare globală (GPS) care este oferit gratis în pachetul de bază. Acesta nu identifică doar locaţia telefonului, folosindu-se de sateliţii GPS, dar oferă şi aplicaţiilor posibilitatea de a accesa şi de a se folosi de aceste informaţii. Pe lângă reţeaua de telefonie mobilă existentă, Android permite conectarea la internet prin activarea opţiunii Wireless, cât şi conectarea la alte telefoane sau dispozitive externe, prin activarea opţiunii Bluetooth. Majoritatea telefoanelor prezintă un card de memorie detaşabil care este cunoscut prin numele de card SD (Secure Digital). La fel ca memoria flash, cardurile SD sunt tipuri de memorie nevolatilă ce utilizează tehnologia flash NAND.

Fiind concepute ca elemente portabile, acestea aderă la anumite specificaţii fizice, făcând posibil interoperabilitatea cu celelalte dispozitive. Acest aspect constituie una dintre notele distinctive vis-a-vis de telefoanele Apple, care nu prezintă card SD, ci doar memorie flash cu o capacitate de la 4 GB la 32 GB, după cum este relatat în [9]. Această alegere făcută de producătorii Android nu este doar o soluţie mai ieftină ci şi practică, acest card putând fi utilizat şi de către alte telefoane.

Ecranul este interfaţa sistemului de operare pentru interacţiunea cu utilizatorul, fiind important nu doar pentru impactul pe care îl produce din punct de vedere vizual, cât şi pentru răspunsul pe care îl oferă la atingere. Acesta este într-o continuă dezvoltare urmărindu-se creşterea rezoluţiei, a luminozităţii, precum şi capacităţii de răspuns la stimulii externi. Camera telefonului şi-a depăşit atribuţiile cunoscute de a face poze sau de a filma, utilizând celelalte componente hardware pentru a-şi lărgi aria de utilitate. Majoritatea dispozitivelor combină funcţionalitatea acesteia pentru a înregistra pe lângă data şi timpul la care s-a realizat o poză sau un video şi coordonatele GPS la momentul utilizării acesteia.

Datele multimedia pot fi cu uşurinţă încărcate pe web şi partajate cu alte persoane. Interesantă este utilizarea acestei comoponente în domeniul realităţii augmentate, camera putând fi folosită pentru a identifica un produs pe baza codului de bare sau a unei clădiri pe baza coordonatelor geografice sau a unor etichete. Tastatura a fost şi ea înlocuită cu una ce încorporează tehnologia touch screen. Aceasta se poate adapta după poziţia telefonului şi poate suporta mai multe limbi, existând şi posibilitatea utilizării unor sisteme ce au rolul de a prezice cuvintele ce urmează a fi scrise. Durata de viaţă a bateriei constituie încă o problemă datorită aplicaţiilor existente ce sunt mari consumatoare de energie. Se doreşte depăşirea acestui neajuns, existând cercetări în acest sens, după cum apare în [9], ce urmăresc realizarea unor baterii din ce în ce mai puternice sau încărcarea bateriei fără utilizarea vreunui cablu, un exemplu fiind încărcarea în timpul mişcării. Majoritatea dispozitivelor Android oferă suport pentru USB (Universal Serial Bus) cu ajutorul căruia telefoanele pot fi conectate la un calculator. Prin cablu se poate realiza atât încărcarea bateriei, cât şi accesarea memoriei SD sau a consolei Linux de pe dispozitiv.

Android poate detecta şi schimba poziţia ecranului, în funcţie de felul în care telefonul este ţinut sau rotit, acest lucru fiind posibil datorită existenţei unui accelerometru care detectează aceste schimbări. Noile versiuni suportă şi un giroscop, care este mult mai senzitiv decât un accelerometru, aspect ce a dus la dezvoltarea jocurilor pe această platformă. Ultimele componente hardware sunt constituite de difuzorul şi microfonul telefonului, care la fel ca şi celelalte părţi sunt perfecţionate de la o versiune la alta. Unele dispozitive prezintă două sau trei microfoane, care combinate cu software-ul Android au abilitatea de a detecta şi elimina zgomotul de fundal pentru a oferi o calitate mai bună a sunetului.

## 2.3 Sistemul de operare

Sistemul de operare este bazat pe nucleul Linux 2.6 care oferă software-ul necesar pentru a coordona atât partea hardware, cât şi aplicaţiile Android. Aşa cum este ilustrat în Figura 1.2, funcţiile low-level includ administrarea energiei, a conexiunii wireless, a ecranului, a driverelor audio şi a memoriei flash. Peste kernel este construit un set de librarii ce oferă funcţionalităţile principale de care au nevoie atât dezvoltatorii, cât şi utilizatorii. Acestea includ librării precum cele ce gestionează lucrul cu bazele de date (SQLite), cel ce furnizează utilizarea de socket-uri securizate bazate pe protocolul SSL, dar şi librăria standard C.

Conform, librăriile principale sunt unite cu o maşină virtuală customizată Java, ce furnizează mediul de rulare pentru aplicaţiile Android. Dezvoltatorii scriu şi compilează programele cu kit-ul de dezvoltare Java oferit de companie Sun (Sun’s Java Development Kit) şi byte code-ul rezultat este transformat în fişiere de tip .dex ce sunt rulate apoi pe maşina virtuală Dalvik.

Fiecare program rulează în procesul său propria lui instanţă a maşinii virtuale Dalvik, care este concepută astfel încât mai multe multe instanţe ale acesteia să poată rula pe acelaşi dispozitiv într-un mod eficient.

Utilizarea acestor resurse este realizată prin intermediul unor API-uri şi a unui application framework, cu care programatorii interacţionează şi care oferă acces acestora la metodele necesare în crearea aplicaţiilor dorite.

## 2.4 Rezumat Android

Acest capitol oferă o privire de ansamblu asupra platformei Android atât din punct de vedere hardware, cât şi software. Deşi unele componente variază de la un producător la altul, structura este comună pentru fiecare dintre aceste dispozitive.

Telefonul poate fi privit asemenea unui mini computer, prezentând un procesor de tip ARM şi doua tipuri de memorie, combinate într-o singură componentă, numită pachet multicip (MCP). Conexiunea la reţeaua mobilă este realizată de către modem-ul baseband şi radioul existent, conectarea la internet prin activarea opţiunii Wireless, iar la alte telefoane sau dispozitive externe prin activarea opţiunii Bluetooth sau USB. Pe lângă acestea întâlnim o cameră performantă, un ecran touch screen şi un sistem de localizare GPS.

Sistemul de operare este bazat pe nucleul Linux 2.6 şi oferă software-ul necesar pentru a coordona atât partea hardware, cât şi aplicaţiile Android. Librăriile sunt accesate de programatori prin intermediul API-ului existent, iar rularea programelor se realizează pe maşinile virtuale Dalvik.

Din punct de vedere structural, o aplicaţie Android este compusă din două elemente principale: clasele ce descriu funţionalitatea programului şi resursele constituite din aspecte, imagini, şiruri de caractere sau culori ce descriu interfaţa programului. Definirea structurii şi metadatei aplicaţiei, componentele şi cerinţele acesteia sunt precizate în fişierul manifest al aplicaţiei, fişier fără de care o aplicaţie nu poate rula.

# CAPITOLUL 3. Aspecte teoretice

În acest subcapitol, principalele obiective vor avea drept scop înțelegerea evenimentelor petrecute în aplicația curentă, familializarea cu unele concepte folosite şi oferirea unor detalii legate de implementarea acestora.

## 3.1. Perspectiva Server-Side C# .NET CORE

În această lucrare, propun o metodă de creare a unei aplicații în Visual Studio 2019 Professional, într-un limbaj de programare de nivel înalt, C#.Motivul este ca, desi trebuie mentionat că motivatia nu este este valabilă pentru orice context, este un limbaj foarte usor de invatat si folosit, dupa cum am mentionat având o sintaxă foarte asemanatoare cu limbajele clasice, este foarte popular, fiind cel mai folosit dintre cele enumerate mai sus si printre cele mai folosite din lume si există o comunitate foarte dezvoltata (deci totodata si resurse disponibile), folosind o arhitectură REST(Representational State Transfer), însoțită de algoritmi eficienți de ultimă generație pentru procesarea unui traseu optim.

Codul C# este compilat ca şi cod de bază, ceea ce înseamnă că beneficiază de serviciile CLR. Aceste servicii includ interoperabilitea limbajelor, garbage collection, securitate ridicată şi suportul versiunilor imbunătăţite.

În .NET (şi, implicit, în C#), tipurile de date se împart în două categorii principale: tipuri valoare şi tipuri referinţă. Diferenţa dintre ele este că variabilele de tip referinţă conţin referinţe (pointeri) spre datele propriu-zise, care se află în heap, pe când variabilele de tip valoare conţin valorile efective. Această deosebire se observă, de exemplu, la atribuiri sau la apeluri de funcţii.

La o atribuire care implică tipuri referinţă, referinţa spre un obiect din memorie este duplicată, dar obiectul în sine este unul singur (are loc fenomenul de aliasing - mai multe nume pentru acelaşi obiect). La o atribuire care implică tipuri valoare, conţinutul variabilei este duplicat în variabilă destinaţie. Tipurile valoare sunt structurile (struct) şi enumerările (enum). Tipuri referinţă sunt clasele (class), interfeţele (interface), tablourile şi delegările (delegate).

Poate părea surprinzător, dar unele din tipurile "primitive" cu care suntem obişnuiţi din C (int, de exemplu), sunt tot structuri. Altele, cum ar fi string, sunt clase. Singurul lucru care le face mai speciale în C# este faptul că, la compilare, textul din codul sursă este identificat şi convertit automat în instanţe ale acestor tipuri. În plus, pentru aceste tipuri, există cuvinte cheie C# prin care sunt descrise, dar şi clase .NET care reprezintă implementarea propriu-zisă.

In implementarea .NET este folosit CLI (common language infrastructure), ceea ce înseamnă, printre alte lucruri, ca dezvoltarea in acest framework oferă interoperabilitate si un runtime independent de limbaj. Astfel putem enumera:

* C# care este cel mai simplu limbaj din ecosistemul Microsoft. Find orientat pe obiecte si “type-safe” (specificând tipul de date pentru fiecare variabilă), este foarte asemanator cu limbajele consacrate C si c++.
* F# este un limbaj functional cross-platform care suporta si modul traditional de programare orientat pe obiecte.
* Visual Basic care este un limbaj mai vechi dar inca mentinut si dezvoltat de catre Microsoft.

Fiind precursorul .NET Framework, .NET CORE este o colectie de librări si tehnologi independente de platforma, ce este intretinută si dezvoltată de catre Microsoft si comunitatea de programatori. Avand un caracter open-source si free to use, putem gasi codul sursă pentru toate componentele framework-ului si le putem folosi in scopuri comerciale fară costuri suplimentare sau impartirea dretpurilor intelectuale.

.Net CORE suportă diferite tipuri de aplicații cum ar fi: Console Applications, Web Appications, Class Libaries, Unit Tests .etc. Deasemenea fiind publicată recent de catre Microsoft, are o foarte bună integrare cu alte domenii auxiliare ca: Mobile Aplications, Cloud Computing si in general cu platforme bazate pe comunicarea pe internet.

Managementul memoriei

.NET foloseste un sistem de management al memoriei numit “Garbage collection” (in traducere “colector de gunoi”). Acesta rulează in spate, asigurandu-se ca programul este memory safe, adică foloseste doar zone de memorie alocate in prealabil.

**Tipurile de date**

C# este un limbaj “strongly-typed”, ceea ce inseamnă ca fiecare instanță are un anumit tip. Acest lucru se rasfrange si asupra proprietatilor si metodelor, unde un obiect nu are acces decât la cele proprii.

Fiind un limbaj orientat pe obiecte, conține si partea de ierarhizare si de clase derivate. Asfel cast-urile pot face referința doar la un tip din aceeasi ierarhie

**Ex:**

Pisica pisica = New Pisica();

Animal animal = (Animal) pisica // acest cast va funcționa pentru ca pisica este un animal

Casa casa = (Casa) pisica //aceast cast va rezulta intr-o eroare pentru ca nu fac parte din aceeasi ierarhie.

**Template-uri si delegați**

Deasemenea, intr-un limbaj atat de raspandit si complex nu puteau lipsii construcții ca: funcții lambda, delegați sau funcții generice. Primele doua, sunt foarte asemanatoare cu omoloagele lor in C++ si reprezentand pointeri de funcții ce au un anumit tip. Funcțiile generice permit folosirea unui parametru de tip “TYPE” care permite creerea unor “funcții-sablon” ce pot fi apelate pentru diverse obiecte.

Acestea sunt, in mare, “uneltele” sau cele mai importante aspecte din framework-ul .NET pe care le vom utiliza, dar pentru intelegerea completă si pentru caracterul stiințific al lucrarii mai amintim ca aspecte ale acestei tehnologii:

* O inclinare mare asupra programării asincrone, oferind diverse modalitati/metode pentru a facilita lucrul cu fire de executie sau thread-uri.
* Propriul limbaj de interogare (numit LINQ - Language Integrated Query)
* Propriul sistem de hosting, prin caracterul de “self-hosting” al tehnologiei

**Microsoft Azure**

“Cloud Computing” este o paradigmă tehnologică modernă ce foloseste puterea calculului, procesarii si stocarii distribuite pentru a rezolva diverse probleme intr-un mod mai rapid, mai eficient si mai ieftin. Astfel, in prezent exista diferite companii ce ofera astfel de servicii atat pentru utilizatori independenti cat si pentru companii.

Ca avantaj principal, cloud-ul permite utilizatorului sa se concentreze exclusiv pe logică si funcționalitățile produsului, preluând toate responsabilitatile ce țin de infrastructura, mentenanța, securitate si in general de gestionarea fizică a mediilor computaționale.

În timp, după diferite modificări și prin evoluația firească a conceptului, s-au delimitat câteva tipuri principale:

● Cloud privat - care conține o infrastructură, administrată intern sau nu, dediacata unei singure entități.

● Cloud public - cel mai răspândit tip, reprezintă un mediu, gratis sau contra cost, unde oricine poate beneficia de infrastructură distribuită ( exemple: Azure, AWS, , Google..etc)

● Cloud hibrid - este o modalitata mai rară de a lucra cu această paradigmă, dar poate fi foarte folositoare în anumite situații. Acest tip de Cloud permite extiderea uni cloud public cu unul privat. Astfel o companie poate folosi un mediu distribuit accesibil tuturor pentru anumite resurse pe care le poate conecta cu un mediu proprietar. (exemplu: Azure Stack)

● Alte tipuri: Aici puten enumeră: Comunity Cloud, Distributed Cloud, Mulți Cluod, Big Dată Cloud, HPC Cloud

Microsoft a intrat pe această piață în 2010 prin “Microsoft Azure” care este o platforma de cloud computing folosită pentru dezvoltare, testare și publicare de aplicații personale dar și pentru numeroasele servicii și resurse folosite. Avantajul competițional este în principal multitudinea de tehnologii și limbaje compatibile și acceptate dar și prin apartenența la ecosistemul .NET.

Se definesc 3 tipuri principale de servicii oferite:

● SaaS (Software aș a service) - Reprezintă un mod modern de livrare și comercializare de soluții software, sub formă de serviciu. Fiind numit și “on demand” (la cerere), este un model bazat pe centralizarea resursei, această fiind consumată de către clienți prin apeluri în contextul unei subscriptii. Cuantificarea prețului fiind de obieci făcută prin numărul de cereri. Acest tip de livrare are diverse avantaje că

○ Posibilități ridicate de configurare și particularizare

○ Timp de livrare crescut (o versiune nouă fiind publicată doar printr-un “deploy”)

○ Integrare cu alte servicii Azure

○ Comunicare și colaborare sporită prin caracterul unificat

● PaaS (Platform aș a service) - Această categorie de servicii cloud se referă la comercializarea (sau chiar și oferirea într-un mod gratuit) a unei platforme, unde clienții pot dezvoltă, rula, menține și monitoriza aplicații, eliminând responsabilitățile și cheltuielile ce țîn de infrastructură și securitate.

● IaaS (Infrastructure aș a service) - Reprezintă o categorie de servicii online care oferă API-ri și unelte high end, pentru a permite utilizatorului să își dezvolte soluția în așa fel încât să aibă acces la detalii foarte specifice (low-level) că locație, partiționare de date, securitate..etc.

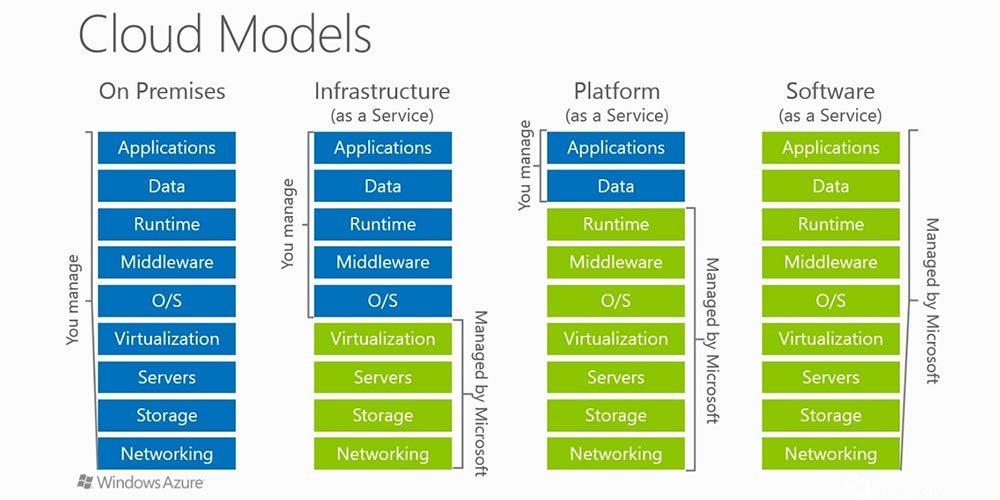


Fig.3.1. Cele mai utilizate servicii cloud [11]

În prezentarea și documentația oficială, sunt listate peste 600 de servicii diferite oferite de platforma Azure. Acestea sunt folosite atât de utilizatori independenți cât și de companii mici, medii și mari. Într-o clasificare mai generală, bazată pe scop, putem să le definim pe acestea astfel:

* Servicii mobile
  + Servicii de tip REST, optimizate pentru a fi folosite de catre aplicatii mobile
  + Unelte pentru analiza si monitorizare
  + HokeyApp pentru a dezvolta si distribuii aplicatii
* Servicii de stocare
  + Kit de dezvoltare pentru accesul la date
  + Produse ca: Table Service, Blob Service, Queue Service si File Service
* Management de date
  + Baze de date relationale oferite ca serviciu (ex. Azure SQL)
  + Baze de date non-relationale oferite ca serviciu (ex. Cosmos DB, Table Storage)
  + Redis Cache
  + SQL Data Warehouse
  + Azure Data Lake
  + Alte sisteme de gestionare de date
* Putere computationala
  + Masini virtuale oferite ca serviciu
  + Medii de tip “PaaS”
  + Medii distribuite pentru hostare de aplicatii
* Comunicare
  + Event Hubs
  + Queues (cozi de comunicare)
  + Topics (se ocupa de comunicarea intr-o singura directie)
  + Relays (comunicare in ambele sensuri)
* Servicii media
  + Se refera la modalitati de criptare, streaming sau protectie de date.
* Macine learning
  + Microsoft Azure Machine Learning care este parte din suita “Coratana Intelligence Suite”
  + Microsoft SMA
* IoT
  + Azure Sphere, care reprezinta primul produs Microsoft in directia “internet of things”

HTTP ete un acronim de la “Hypertext Transfer Protocol” si reprezintă un mijloc de a comunica intre sisteme pe internet. HTTP este un protocol ce funcționează pe sistemul “request-response”. Clientul poate fi o aplicație consumatoare ca un website, un browser sau o aplicație mobilă iar serverul poate fi reprezentat printr-un server ce posedă anumite functionalitați.

Astfel, clientul trimite un apel HTTP catre server, care trimite înapoi resursele ce au fost cerute. Acestea pot fi fișiere HTML, XML, JSON sau alte tipuri, deasemenea un astfel de apel poate declansa acțiuni.

Comunicarea se face folosind un link sau o adresp web care îndrumă cererea catre destinația corectă folosindu-se protocoloul DNS pentru convertire.

Elemente componente:

* Tipul requestului: Un apel de tip HTTP poate reprezenta mai multe acțiuni, cele mai comune fiind : GET (pentru a aduce date), PUT (pentru a actualiza date), POST ( pentru a insera sau creea date noi) și DELETE (pentru a șterge date)
* Headers: Odata cu cererea, se pot specifica diferite informații auxiliare folosind aceste câmpuri. Astfel putem comunica serverului lucruri ca: formatul in care vom trimite datele, formatul in care dorim sa primim datele, identitatea noastra, detalii cu privire la autorizare..etc
* URL-ul: Fiind la baza o adresa catre care se va duce apelul, acesta poate fi deasemenea folosit pentru a transmite anumite informatii. ex. un apel GET catre
* <https://ego-station.azurewebsites.net/api>/station/poate intoarce o lista cu toate statiile, iar un apel GET catre <https://ego-station.azurewebsites.net/api>/station/id poate intoarce doar detaliile pentru aceea satie specifica bazata pe un id
* Request: Reprezintă totalitaea componentelor definite anterior, fiind compus dintr-o adresa, o colectie de headere, un mesaj si verbul corect ce reflecta actiunea dorită.
* Status code: Este primit de catre client sau consumator sub forma unei colecții standardizate de numere intregi care are ca semnificație interpretarea starii raspunsului. Astfel, doar analizând acest numar întreg obținem informația referitoare la ce s-a întamplat cu cererea noastră. (Ex. codul 200 - “OK”, codul 400 - “Bad Request”, 302 - “Found”, 403 - “Forbidden”)
* Response: Reprezinta răspunsul primit in urma apelului, fiind in general compus dintr-un mesaj si un status code.

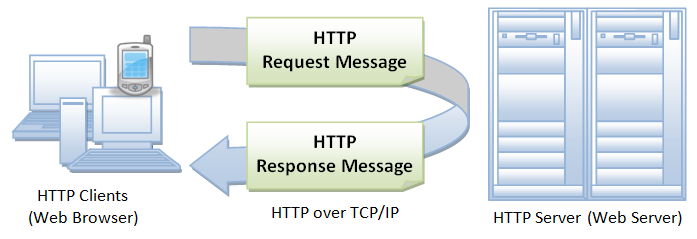


Fig 3.2 Interacțiunea printr-un protocol HTTP [12]

Țînând cont că protocolul HTTP se ocupă de transmitere și operații cu date, apare în mod firesc nevoia de securitate. Astfel, vine în întâmpinarea acestui lucru, “extensia” acestuia numită HTTPS (Secure Hyper Text Transfer Protocol).

Această îl înglobează pe predecesorul lui și reprezintă încapsularea lui într-un flux SSL/TSL ce deservește defapt la criptarea informațiilor pentru procesul de transport.

Pentru a avea un context clar, trebuie menționat că protocolul HTTP trimite mesajul de la sursă către destinație în mod clar, astfel, o posibilă interceptare pe traseu poate duce la expunerea mesajului, creeand astfel o breșă de securitate. În cazul HTTPS, datele sunt criptate pe server folosind o cheie publică, transmise pe internet și apoi decriptate pe client folosind o cheie privată. Astfel un posibil atacator, ce ar putea obține mesajul, nu are suficientă informații pentru a îl interpreta.

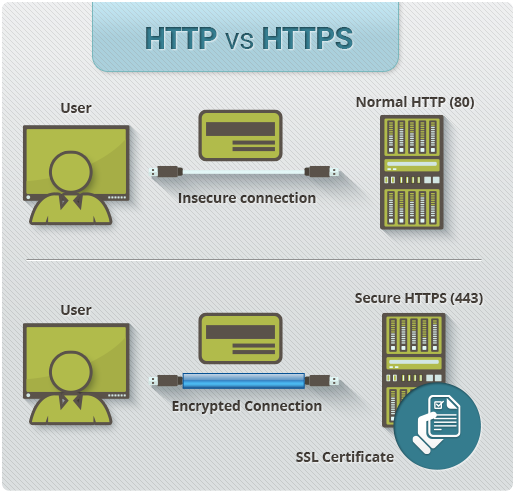


Fig 3.2 Diferența dintre HTTP și HTTPS [12]

**Newtonsoft JSON**

Newtonsoft JSON este o librărie pentru un ecosistem .NET Framework sau .NET Core, ce vizează lucrul cu tipul de date JSON, specific javascript.

Deși nu este o tehnologie atât de complexă, pentru corectitudinea acestei lucrări este nevoie menționarea ei pentru că transmiterea, serializarea și deserializarea mesajelor este un lucru esențial în dezvoltarea unei arhitecturi bazate pe schimb intens de mesaje.

Newtonsoft JSON este un așa-zis “NuGet”, cum este denumit în lumea .NET, și reprezintă o colecție mult mai complexă față de cea de baza. Printre funcționalități putem enumeră: serializare/deserializare îmbunătățită, integrare cu libajul pentru interogare specific .NET, LINQ, o sintaxa simplificată, disponibilitate mărită (poate fi folosit și pentru aplicațiile mobile de exemplu), este open source și suportă deasemenea și operații cu XML.

## 3.1.1 Clase si Tablouri

O clasă este o structură de date care poate conţine date (variabile şi constante) şi cod (metode, proprietăţi, evenimente, operatori, constructori şi destructori). Clasele pot fi derivate din alte clase, prin acest mecanism clasele derivate putând extinde, specializa sau modifică comportamentul clasei de bază.

Clasele reprezintă piatră de temelie a programării pe obiecte. Acest concept porneşte de la aserţiunea că datele sunt strâns legate de codul care le prelucrează şi că gruparea lor într-o structură comună (spre deosebire de limbajele procedurale, în care datele sunt separate de cod) duce la o proiectare, implementare, testare şi întreţinere mai simplă.

Clasele sunt tipuri referinţă, ceea ce înseamnă că datele membru ale instanţelor se află în heap şi programul are acces la ele prin referinţe. Referinţele sunt păstrate corespunzător de runtime şi în cazul în care nu mai există referinţe spre o instanţă, memoria folosită de această este recuperată automat.

Sunt eliminate astfel o serie întreagă de defecte software legate de dealocarea memoriei - în C++ pot apărea situaţii în care memoria nu este dealocată şi este pierdută pâna la oprirea programului sau situaţii în care memoria este dealocată de o componentă dar altă componentă încearcă ulterior să scrie la acea adresă pentru că are un pointer pe care îl crede valid.

Membrii unei clase pot fi statici; în acest caz, există o singură instanţă a lor care este accesibilă din toate instanţele clasei respective; în situaţia în care membri statici sunt şi publici, ei vor fi accesibili din alte clase fără a avea nevoie de o instanţă a clasei respective. Mai mult, ei nu pot fi accesaţi prin intermediul unor instanţe, ci numai folosind numele clasei.

De asemenea, pot fi definite metode statice. În mod similar, acestea aparţîn clasei şi nu instanţelor acesteia şi pot fi apelate folosind numele clasei. În general, acţionează asupra membrilor statici sau efectuează operaţii pentru care nu este necesară o instanţă a clasei.

Un tablou este o structură de date care conţine un număr de variabile numite elemente, accesate prin intermediul unuia sau mai multor indici întregi. Tablourile sunt tipuri referinţă, iar pentru a crea un tablou trebuie alocat explicit spaţiu cu operatorul new. În acest caz, un tablou este efectiv creat, iar valorile elementelor sunt iniţializate cu valoarea implicită a tipului. Pentru a crea un tablou de obiecte (tipuri referinţă), trebuie creat întâi tabloul, elementele acestuia vor fi iniţializate implicit cu null, după care tabloul trebuie parcurs şi elementele sale iniţializate explicit de către programator cu valori diferite de null.

## 3.1.2 Drepturi de acces

Încapsularea reprezintă un aspect esenţial al programării pe obiecte şi înseamnă, pe scurt, posibilitatea de a separă implementarea unui obiect de funcţionalitatea pe care acesta o pune la dispoziţie utilizatorilor săi. Prin implementarea unui obiect ne referim, în general, la variabilele membru ale obiectului respectiv şi la metodele interne, care contribuie la realizarea funcţionalităţii obiectului, dar care nu sunt necesare utilizatorilor obiectului respectiv.Dacă specificatorul de acces lipseşte, membrii unei clase sunt implicit privaţi.Drepturile de acces reprezintă modalitatea prin care putem ascunde de utilizatorii clasei detaliile de implementare ale acesteia. Cei cinci specificatori de acces în C# şi semnificaţia lor sunt:

**private** - Membrii privaţi sunt accesibili numai din clasa în care sunt declaraţi.

**internal** - Membrii interni sunt accesibili numai din clase care fac parte din acelaşi assembly cu clasa în care sunt declaraţi.

**protected** - Membrii protejaţi sunt accesibili din clasa în care sunt declaraţi şi din clasele derivate din aceasta.

**protected internal** - Membrii interni şi protejaţi sunt accesibili din clasa în care sunt declaraţi, din clasele care fac parte din acelaşi assembly şi din clasele derivate, indiferent în ce assembly sunt acestea.

**public** - Membrii publici sunt accesibili din orice clasă.

O clasă este alcatuită din membrii moşteniţi din clasa de bază şi din membrii declaraţi în corpul clasei. Datele pot fi variabile, constante, sau variabile read-only.Diferența dintre constante și variabile read-only este că valoarea constantelor poate fi calculată la compilare și plasată în fișierul executabil că atare, pe când valoarea variabilelor read-only este calculată la rulare. Implicit, pot fi declarate doar constante apartinând tipurilor recunoscute de compilator că "primitive". Variabilele read-only pot fi de orice tip și pot avea valori care nu sunt cunoscute la compilare.

## 3.1.3 Constructori si Destructori

Constructorul este o metodă care implementează acţiunile care au loc la crearea şi iniţializarea unei instanţe a clasei. Are acelaşi nume că şi clasa din care face parte şi nu returnează nici o valoare, nici măcar void. Poate apela un alt constructor al aceleiaşi clase sau un constructor al clasei de baza. Poate avea orice nivel de acces.

Dacă într-o clasă nu este implementat nici un constructor, compilatorul inserează automat un constructor public, fără argumente, care iniţializează automat toate variabilele cu valorile lor implicite.

În .NET, memoria ocupată de obiecte este automat recuperată de un garbage collector în momentul când nu mai este folosită. Uneori, însă, un obiect este asociat cu resurse care nu depind de .NET şi care trebuie dealocate explicit, într-un fel sau altul. Un exemplu de astfel de resurse îl reprezintă conexiunile TCP/IP, care nu pot fi lăsate la voia întâmplarii.

De obicei, este bine să se elibereze astfel de resurse în momentul când nu mai sunt necesare şi vom vedea puţîn mai încolo cum se face acest lucru. Există însă şi o plasă de siguranţă oferită de compilator, reprezentată de destructori.

Destructorii sunt metode care au acelaşi nume cu clasă din care fac parte, precedat de semnul ~. Nu au drepturi de acces, nu au argumente şi nu permit nici un fel de specificatori (static, virtual șamd). Nu pot fi invocaţi explicit, ci numai de librăriile .NET specializate pe recuperarea memoriei. Ordinea şi momentul în care sunt apelaţi sunt nedefinite, că şi firul de execuţie în care sunt executaţi. Este bine că în aceste metode să dealocaţi numai obiectele care nu pot fi dealocate automat de .NET şi să nu faceţi nici un fel de alte operaţii.

## 3.1.4 Moştenire

Prin moştenire se înţelege crearea unei clase derivate care conţine implicit toţi membrii (mai puţîn constructorii, constructorul static şi destructorul) unei alte clase, numite de bază. Moştenirea se mai numeşte şi derivare sau, mai rar, specializare.

În C#, moştenirea se realizează punând ":" după numele clasei, urmat de numele clasei de bază. O clasă poate avea o singură clasă de bază. Dacă o clasă nu este derivată explicit din nici o clasă, compilatorul o face implicit să fie derivată din object. Object este rădăcina ierarhiei de clase din .NET.

Moştenirea este tranzitivă, în sensul că dacă A este derivată din B şi B este derivată din C, implicit A va conţine şi membrii lui C (şi, evident, pe cei ai lui B). Prin moştenire, o clasă derivată extinde clasa de bază. Clasa derivată poate adăuga noi membri, dar nu îi poate elimina pe cei existenţi.

Deşi clasa derivată conţine implicit toţi membrii clasei de bază, acest fapt nu înseamnă că îi şi poate accesa. Membrii privaţi ai clasei de baza există şi în clasa derivată, dar nu pot fi accesaţi. În acest fel, clasa de bază îşi poate schimbă la nevoie implementarea internă fără a distruge funcţionalitatea claselor derivate existente.

O referinţă la clasa derivată poate fi tratată că o referintă la clasa de bază. Această conversie se numeşte upcast, din cauză că în reprezentările ierarhiilor de clase, clasele de bază se pun deasupra, cele derivate dedesubtul lor, că într-un arbore generalizat. Prin upcast se urcă în arbore.

Conversia inversă, de la clasa de bază la cea derivată, se numeşte downcast şi trebuie făcută explicit, deoarece nu ştim dacă referinţa indică spre un obiect din clasa de bază, spre un obiect din clasa derivată la care încercăm să facem conversia sau spre un obiect al altei clase derivate din clasa noastră de bază.

Accesibilitatea trebuie să fie consistentă şi în cazul în care încercăm să derivăm o clasă din altă. Clasa de bază trebuie să fie cel puţîn la fel de accesibilă că şi clasa derivată din ea.

O clasă derivată poate ascunde membri ai clasei de bază, declarând membri cu aceeaşi semnătură. Prin această, membrii clasei de bază nu sunt eliminaţi, ci devin inaccesibili prin referinţe la clasa derivată. Ascunderea membrilor se face folosind cuvântul cheie new. Acest cuvânt cheie are rolul de a-l obligă pe programator să-şi declare explicit intenţiile şi face, în acest fel, codul mai lizibil.

## 3.1.5 Metode

În C#, este posibil, ca şi în C++ sau Java să avem mai multe metode cu acelaşi nume în cadrul aceleiaşi clase. În C#, la supraîncarcare se ţine cont de numele metodei, de tipul argumentelor, de categoria lor (de intrare, intrare-iesire, iesire), de numărul şi de ordinea lor. Semnătura unei metode nu include tipul returnat, numele argumentelor sau argumentele variabile de la sfârşitul listei de argumente.

Supraîncărcarea metodelor şi operatorilor se mai numeşte şi polimorfism static, deoarece putem obţine comportamente diferite prin acelaşi apel de funcţie, dar comportamentul este totuşi bine determinat la compilare.

În C#, este posibil ca o metoda să primească un număr variabil de argumente. Argumentele variabile sunt transmise într-un tablou, iar numele acestui tablou trebuie precedat în declaraţia funcţiei de cuvântul cheie params.

Polimorfismul este a treia piatră de temelie a programării pe obiecte. Dacă încapsularea afectează în mod direct uşurinţa de implementare şi utilizare a claselor prin separarea implementării de funcţionalitate, iar moştenirea reprezintă o modalitate simplă de reutilizare a codului, polimorfismul reprezintă componenta esenţiala când vine vorba despre extensibilitate.

O metodă virtuală este o metodă care poate fi suprascrisă într-o clasă derivată.

## 3.1.6 Proprietăţi

Proprietăţile sunt membri care oferă acces la atributele unui obiect. Exemple de proprietăţi includ lungimea unui şir de caractere, lungimea unui tablou, mărimea unui font, titlul unei ferestre şamd. Proprietăţile pot părea sinonime cu variabilele membru ale unei clase au nume, au tip şi sunt folosite în mod similar de utilizatorii clasei, cu aceeaşi sintaxă. Cu toate acestea, proprietăţile diferă de variabilele membru prin faptul că nu reprezintă spaţii de memorie, ci funcţii de acces.

Aceste funcţii de acces sunt executate când utilizatorii clasei încearcă să obţină sau să modifice aceste atribute. Ele sunt similare cu modelul get/set din alte limbaje de programare (aşa şi sunt implementate intern), dar oferă o modalitate mai coerentă de lucru împreună cu aceleaşi avantaje precum posibilitatea de a valida argumentele furnizate la modificarea atributelor sau de a anunţa alte obiecte despre modificările apărute.

## 3.1.7 Dependency Injection

Dependency Injection este un șablon ce se bazează pe clase slab cuplate. Astfel, ideea din spate este de a oferi un serviciu catre client in loc a lasa clientul sî își creeze serviciul dorit. Mai concret, codul clientului trebuie sa ramană neschimbat chiar daca funcționalitatea apelată se schimba. În acest sens, Dependency Injection este o forma de Inversion of control, sursă delegând responsabilitatea de a primi implementarea de care este dependența catre o entitate externî numită injector. Ca imagine de ansamblu, o implementare de acest gen presupune faptul ca obiectul nu cunoaste nimic despre logica de care este dependent si in schimb este folosita o interfata pentru a definii metodele accesibile.

Folosind acest principiu, obținem un grad de abstractizare, folosindu-ne de interfețe, reducand astfel locațiile si modalitațile de scriere de cod. Practic putem avea referință catre o interfață cu mai multe implementari definite la runtime.

In cazul concret al proiectului nostru, folosim dependency injection pentru testare. Asfel, folosim interfete care au o implementare reală pentru funcționalitatea principală a aplicației si o logică diferită pentru contextele de testare unitară.

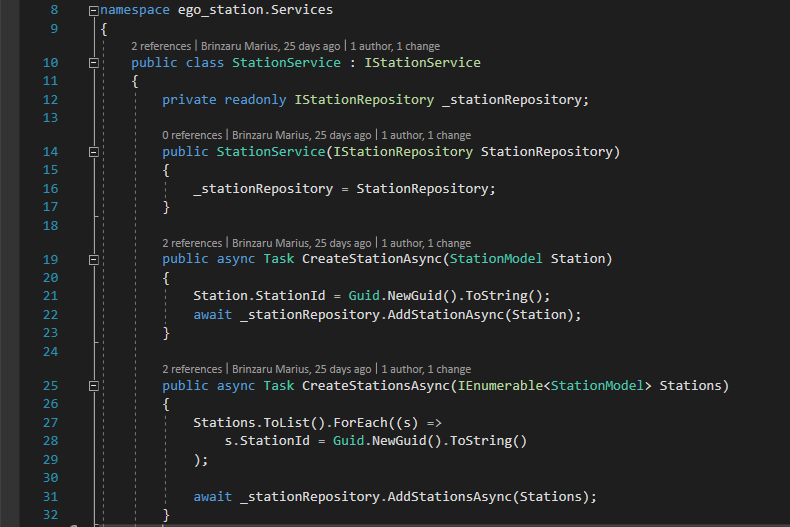


Fig.3.3 Folosirea șablounului Dependency Injection si a interfețelor

## 3.2 Perspectiva interfata utilizator IONIC + Angular

IONIC este un SDK(software development kit) mai exact un set de instrumente disponibile intr-un pachet destul de generos plin de instrumente si componente care este destinat crearii aplicațiilor de mobile hybrid, desktop si aplicatii web progresive.

Acest SDK se bazează pe un ansablu modern de tehnologii precum CSS, HTML5 si SASS. Aplicatiile pot fi construite ca o vizualizare web oferind portabilitate pentru orice fel de platformă si sitem de operare.Oferă servicii pentru dezovltatori pentru a crea applicații in cloud , build-uri automate si un IDE pentru a face ca dezovltarea sa fie cât mai accesibilă si un IDE(Integrated development environment) de folosit numit Ionic Studio.Un instrument important pe care acest SDK îl aduce este linia de comandă specifică pentru generearea proiectului, adăugarea de componente, servicii si pachete , instalarea de plugin-uri etc.

IONIC foloseste Cordova pentru a avea acces la toate funcțiile telefonului precum Camera, GPS, Lanterna, Gesturi .etc.Folosind componente vizuale web, metodologiile se bazează pe paradigmele din aria dezvoltării web.

Penru Android suportă o versiune de software minima de 4.4 sau mai mare, pentur iOS versiunea 10 sau mai mult.

In aplicația de Android , accesul la resurse se face prin protocoalele HTTP si HTTPS, explicate într-un capitol anterior, sub forma unui apel catre URL-ul gateway-ului.

Folosind instanța globală a clasei “SharedData”, unde este stocat token-ul de acces generat la login, se creează o cerere cu acestă in lista de headere care este trimis la adresa corespunzatoare.

Toate aceste apeluri sunt definite sub forma unor metode in clasa “AndroidRepository” folosind definiții generice pentru a putea fi refolosite pentru orice tip de date in activitatile respective.

## 3.3. Autentificare Auth0

Autentifiacarea aplicației se bazează pe un provider autorizat de autentificare si anume Auth0.Este unul dintre liderii de pe piață serviciilor de autentificare renumit pentru securitatea sporită pentru care o oferă,accesibilitate in funcție de orice tehnologie si mai ales faptul că este bazat pe un standar renumit JWT.

**JSON Web Token** (JWT) este un standard de internet pentru crearea de date cu semnătură opțională și / sau criptare opțională a cărei sarcină utilă deține JSON care afirmă un număr de revendicări.Jetoanele sunt semnate fie folosind un secret privat, fie o cheie publică / privată. De exemplu, un server ar putea genera un jeton care are revendicarea „conectat ca admin” și care îl poate oferi unui client.

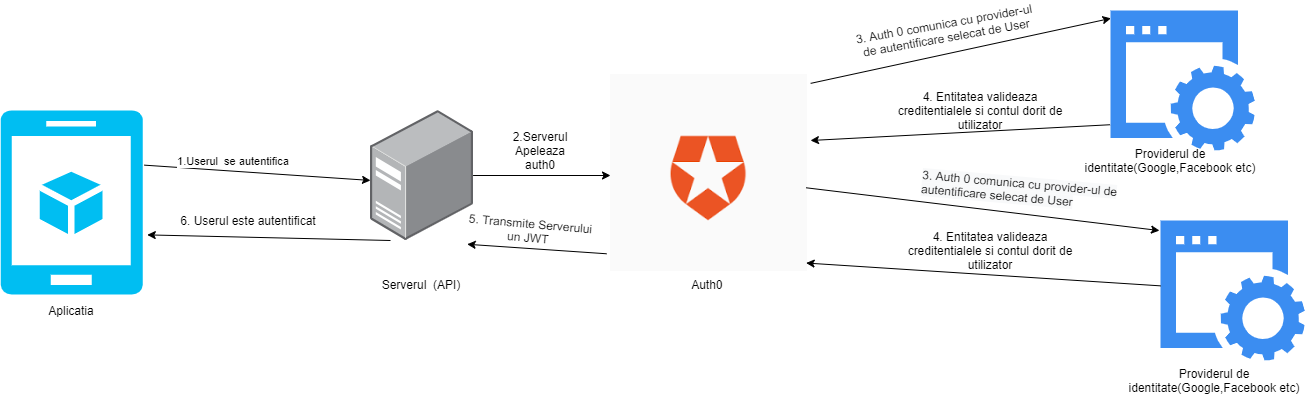
Clientul ar putea apoi să folosească acel jeton pentru a demonstra că este conectat ca administrator. Jetoanele pot fi semnate de cheia privată a unei părți (de obicei serverul), astfel încât partea care poate verifica ulterior simbolul este legitim. Dacă cealaltă parte, prin anumite mijloace adecvate și de încredere, deține cheia publică corespunzătoare, ei sunt capabili să verifice legitimitatea jetonului. Token-urile sunt concepute pentru a fi compacte, URL-safe și pot fi utilizate mai ales într-un context de conectare unică al browserului web. Revendicările JWT pot fi utilizate în mod obișnuit pentru a trece identitatea utilizatorilor autentificați între un furnizor de identitate și un furnizor de servicii sau orice alt tip de revendicări, după cum este necesar în procesele de afaceri.

Auth0 este o soluție flexibilă, pentru a adăuga servicii de autentificare și autorizare aplicațiilor. Aplicatia poate evita timpul și riscul care vin cu construirea propriei soluții de autentificare și autorizare a utilizatorilor.

Auth0 poate conecta orice aplicație (scrisă în orice limbă sau pe orice stivă) la Auth0 și să definiți furnizorii de identitate pe care doriți să-l utilizați (cum doriți ca utilizatorii dvs. să se autentifice ex: Google,Facebook etc).

Pe baza tehnologiei aplicației apelează-ne la API- lui Auth0 și se conecteaza la aplicatie. Acum de fiecare dată când un utilizator încearcă să se autentifice, Auth0 își va verifica identitatea și va trimite informațiile necesare înapoi la aplicație.

Figura 3.4 reprezintă arhitectura pe care am implementat-o în acest proiect precum și modul de funcționare.

Fig.3.4 Diagrama de functionare a autentificarii

## 3.4. Baza de date MonogoDB

In acest proiect am optat pentru o baza de date de tipul MongoDB pentru urmatoarele caractestici enumerate mai jos precum si hostarea bazei de date in cloud prin Serviciul Atlas.MongoDB Atlas este serviciul global de baze de date cloud pentru aplicații moderne care implementat complet gestionat în Azure, aceste doua tehnologii ofera împreună cele mai bune tehnici de automatizare și dovedite practici ce garantează disponibilitatea, scalabilitatea și respectarea celor mai exigente standarde de securitate și confidențialitate a datelor.

Diversele baze de date NoSQL existente azi pe piaţă prezintă diferite abordari. Ceea ce au în comun este faptul că nu sunt relaţionale. Principalul avantaj este acela că permit lucrul eficient cu date nestructurate precum e-mail, multimedia, procesoare de text. În prezent există multe companii care au dezvoltat propriile baze de date NoSQL. Cele mai populare sunt cele dezvoltate de către companiile mari Web 2.0, precum Amazon and Google, din nevoia de a procesa cantităţi mari de date.

Avantajele folosirii sistemelor NoSQL constau în portabilitate, iar serverele acestora utilizează modelul DHT (Distributed Hash Table), astfel încât manipularea datelor obiect se realizează prin furnizarea de către utilizator a cheii primare a obiectului. DHT (Distributed Hash Table) fiind accesibil prin intermediul API Hashtable. Funcţia hash este un algoritm matematic ce stă la baza arhitecturii NoSQL. Acest algoritm are la intrare o lungime variabilă şi produce o lungime fixă la ieşire. Fiecare cheie este partiţionată, iar valoarea hash rezultantă este folosită pentru a se trimite la un server de baze de date NoSQL. Înregistrarea este memorată pentru căutări ulterioare. NoSQL permite accesul la date doar într-un model îngust de acces predefinit. Astfel modelul de acces la date este unul fiabil. Printre sistemele NoSQL din ziua de azi pot fi menţionate cele open-source, precum MongoDB.

Din punctul de vedere al arhitecturii bazelor de date NoSQL există probleme legate de interfeţe şi interoperabilitate.Modelul DHT (Distributed Hash Table) conţine interfeţele sale încă standardizate, însă nu conţine o semantică intrinsecă pentru indicii. Interoperabilitatea este importantă în momentul în care datele trebuie accesate de mai multe servicii, moment în care se poate pierde din performanţă. Spre deosebire de bazele de date relaţionale care s-au consacrat că fiind stabile, sistemele NoSQL apar pe piaţă ca o alternativă ce poate fi pusă în aplicare doar cu precauţie, deoarece încă nu au ajuns la un nivel de maturitate cel puţin egal cu RDBMS-urile. De asemenea, în cazul apariţiei unor probleme, suportul sistemelor NoSQL este încă limitat, acestea fiind în mare proiecte open-source, iar companiile care oferă suport pentru bazele de date NoSQL nu oferă credibilitatea companiilor globale.Sistemele NoSQL se potrivesc foarte bine şi în tehnologia Cloud, care se bazează pe virtualizare. Există şi un punct slab al virtualizării ce ţine de performanţa I/O, limitările CPU-ului şi ale memoriei fiind de altfel în strânsă legătură. Bazele de date NoSQL folosesc proporţia cea mai mare pe memoria de disc, aceasta fiind locaţia principală de scriere, însă datorită scalării orizontale sunt capabile să memoreze datele eficient. Bazele de date NoSQL prezintă un dezavantaj din punctul de vedere al administrării, pentru că necesită un anumit efort pentru a fi menţinute şi cunoştiinţe solide pentru instalare. Persoanele calificate pentru bazele de date NoSQL sunt mai puţine decât cele cu experienţă pentru RDBMS. De aceea preluarea lor de către firme trebuie să fie făcută în mod corespunzător, întrucât beneficiile reale aduse de acestea pot fi însoţite uneori şi de anumite probleme.[13]

MongoDB este o bază de date NoSQL open-source orientată pe documente. MongoDB face parte din familia de sistemelor de baze de date NoSQL. Diferența principală constă în faptul că stocarea datelor nu se face folosind tabele precum într-o bază de date relațională, MongoDB stochează datele sub formă de documente JSON cu scheme dinamice. Dezvoltarea MongoDB a fot începută de 10gen în 2007, când compania dezvolta un produs platform as a service similar cu Windows Azure sau Google App Engine. În 2009 MongoDB a fost transformat într-un produs open-source.

MongoDB este o bază de date open-source NoSQL scrisă în C++. Aceasta poate conține mai multe baze de date, colecții și indecși. În unele cazuri (baze de date și colecții ) aceste obiecte pot fi create implicit.Colecțiile conțin documente (BSON). Aceste documente conțin la rândul lor mai multe câmpuri.

În MongoDB nu există câmpuri predefinite spre deosebire de bazele de date relaționale, unde există coloanele care sunt definite în momentul în care tabelele sunt create. Nu există schemă pentru câmpurile dintr-un document, acestea precum și tipurile lor pot varia. Astfel nu există operația de „alter table” pentru adăugare de coloane. În practică este obișnuit ca o colecție să aibă o structură omogenă, deși nu este o cerință, colecțiile putând avea structuri diferite. Această flexibilitate presupune ușurință în migrarea și modificarea imaginii de ansamblu asupra datelor.

MongoDB lucrează cu date nestructurate şi organizează aceste date în format document. Implementarea acestei baze de date este mai uşoară decat un RDBMS deoarece ea urmăreşte modelul cheie valoare şi nu are nevoie de o schemă predefinită a datelor. Conceptele acesteia pornesc de la concepte tradiţionale, de aceea întelegerea filosofiei acestei baze de date este ceva uşor de realizat.

**Caracteristicile MongoDB**

**Stocarea datelor sub formă de documente.** Baza de date MongoDB stochează obiecte (documente). Aceste documente reduc nevoia de join.

**Prezintă support pentru indexare.** Indexarea pe fiecare din atribute se face în modul tradițional (RDBMS) asupra cheilor de regăsire ale documentelor. Disponibilitatea datelor este asigurată printr-un proces automat de failover.

**Auto-Sharding.** Shardingul sau partiționarea datelor pe orizontală se face automat. Citirile și scrierile sunt distribuite pe partiții. Lipsa joinurilor face ca interogările distribuite să fie rapide.

**Limbajul de interogare**Limbajul de interogare este îmbunatatit și păstreaza principii SQL si C++.

**Modificări rapide.**MongoDB suportă operații de actualizare atomice căt și pe cele tradiționale. Operatiile de mai jos demonstrează flexibilitatea limbajului NoSQL.

**Map Reduce.** MongoDB folosește map/reduce pentru agregare și procesare batch.

**GridFS.**Poate stoca fișiere de dimensiuni mari fără a complica stiva de date.

**Suport.**Dispune de o comunitate care te poate ajuta treci peste anumte impedimente

**Paralelă MySQL versus MongoDB**

|  |  |
| --- | --- |
| **Termeni MySQL** | **Concepte MongoDB** |
| Bază de date | Bază de date |
| Tabelă | Colecție |
| Index | Index |
| Rând | Document BSON |
| Coloană | Câmp BSON |
| Join | Încapsulare și legatură |
| Cheie primară | Câmpul cheie unică \_id |
| Group by | Agregare |

## 3.5. Algoritmul de rutare Google Directions API

Înregistrarea utilizatorilor se face în mod obișnuit prin intermediul Google, ceea ce permite utilizatorilor să se conecteze în siguranță la servicii terțe cu contul Google prin intermediul sistemului de conectare Google. Aceasta este disponibilă în prezent de pe Android (sistem de operare) sau folosind JavaScript.

API este o interfață de calcul care definește interacțiunile între mai mulți intermediari software. Definește tipurile de apeluri sau solicitări care pot fi făcute, modul de realizare a acestora, formatele de date care ar trebui utilizate, convențiile care trebuie urmate, etc. Poate oferi, de asemenea, mecanisme de extensie, astfel încât utilizatorii să poată extinde funcționalitățile existente în diferite moduri și în grade diferite.Un API poate fi personalizată în întregime, specifică unei componente sau poate fi proiectată pe baza unui standard al industriei pentru a asigura interoperabilitatea. Unele API-uri sunt documentate, altele sunt concepute astfel încât să poată fi „interogate” pentru a determina funcționalitatea acceptată. Deoarece alte componente / sisteme se bazează doar pe API, sistemul care furnizează API-ul poate (în mod ideal) să își schimbe detaliile interne „în spatele” acelei API fără a-i afecta pe utilizatorii săi.API Google este un set de interfețe de programare a aplicațiilor (API) dezvoltate de Google, care permit comunicarea cu serviciile Google și integrarea lor în alte servicii.API-ul Direcții este un serviciu care calculează direcțiile între locații folosind o solicitare HTTP**.**

Căutarea de indicații pentru mai multe moduri de transport, inclusiv tranzit, conducere, mers sau mers cu bicicleta.Întoarcerea de direcții în mai multe părți folosind o serie de puncte de referință.

Specificatiile originile, destinațiile și punctele de cale ca șiruri de text (de exemplu, "Bucuresti, RO" sau "Londra,GB"), sau ca coordonate latitudine / longitudine sau ca coduri de loc.

API-ul returnează cele mai eficiente rute atunci când se calculează indicațiile. Timpul de călătorie este factorul principal optimizat, dar API poate lua în considerare și alți factori, cum ar fi distanța, numărul de viraje și multe altele atunci când se decide ce rută este cea mai eficientă.Acest serviciu nu este proiectat pentru a răspunde în timp real la intrarea utilizatorului. Pentru calcule de direcții dinamice (de exemplu, în cadrul unui element de interfață cu utilizatorul).

Înainte de a începe să utilizez API-ul am avut nevoie de un proiect cu un cont de facturare și API-ul activat. Pentru a utiliza API-ul Direcții, trebuie să obtin o cheie API. Cheia API este un identificator unic care este utilizat pentru autentificarea cererilor asociate proiectului de licenta în scopuri de utilizare și facturare.

O solicitare API de instrucțiuni are următoarea formă:

<https://maps.googleapis.com/maps/api/directions/outputFormat?parameters>

Unde outputFormat poate fi una dintre următoarele valori:

JSON (recomandat) indică ieșirea în notarea obiectului JavaScript (JSON)

XML indică ieșirea ca XML

Securitatea este importantă și HTTPS este recomandat ori de câte ori este posibil, în special pentru aplicațiile care includ date de utilizator sensibile, cum ar fi locația unui utilizator, în solicitări. Utilizarea criptării HTTPS face ca aplicația să fie mai sigură și mai rezistentă la inactivare sau alterare.

Parametrii obligatorii

**origin** - adresa, latitudinea textuală / valoarea longitudinală sau ID-ul locului din care se doresta să calculaze direcțiile.

**destination** - adresa, latitudinea textuală / valoarea longitudinală sau ID-ul locului către care doriți să calculați direcțiile. Opțiunile pentru parametrul de destinație sunt aceleași ca pentru parametrul de origine, descris mai sus.

**Key** - cheia API a aplicației Această cheie identifică cererea în scopul gestionării cotelor si fara cheie Google nu proceseaza solicitarea.

Exemple de solicitări de indicații

Următoarea solicitare returnează indicații de conducere din locatia curenta către o statie electrică.

https://maps.googleapis.com/maps/api/directions/**json**?**origin**=44.336,26.156&**desitantion**=44.973,26.726& **key**=AIzaSyA7N7PK0olNEkLval3s18qrKhONNS4tb68

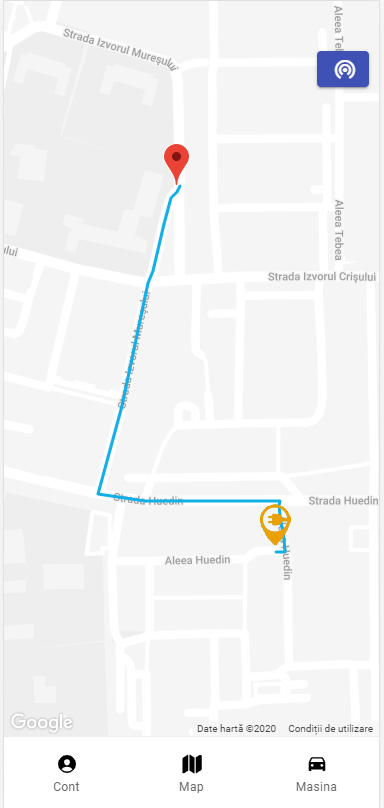


Fig.3.5. Reprezentarea grafica a rutei in aplicatie

Solcitarile catre Google API trebuiesc facute de o entitate tip Server-Side care sa construiasca mesajul exemplificat in figura 3.6 .Mesajul ințial provenind din interfața utilizatorului.

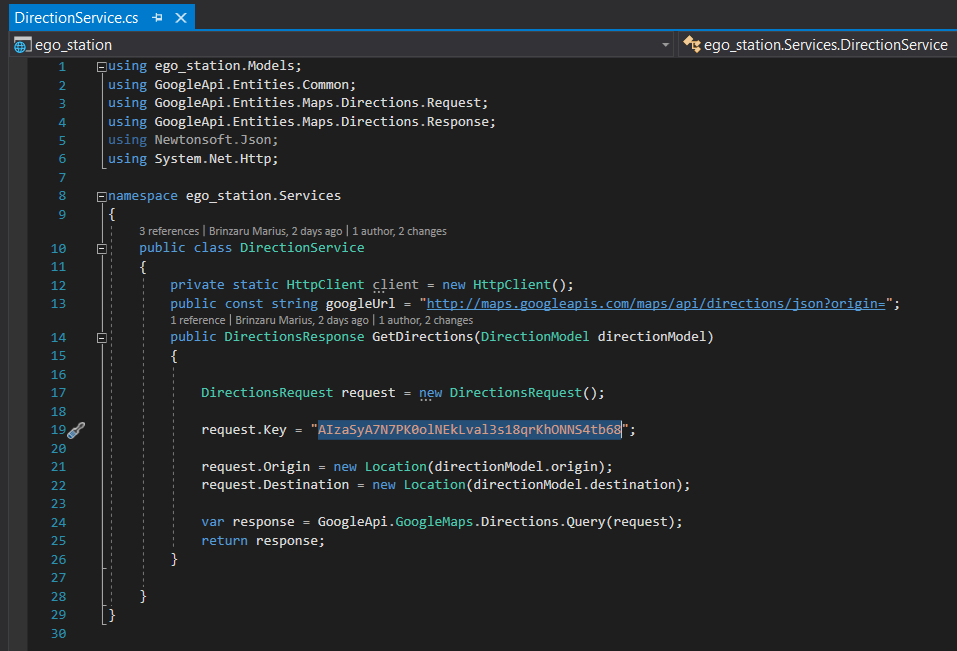


Fig.3.6. Interceptarea mesajului si construirea mesajului

**Parametri opționali**

**mode** (valorile implicite la conducere) - Specifică modul de transport pe care trebuie să îl utilizăm la calcularea indicațiilor. Valorile valide și alte detalii ale solicitării sunt specificate în modurile de călătorie.

**waypoints** - specifică o serie de locații intermediare pe care trebuie să le includă pe traseul dintre punctele de origine și destinație ca locații de trecere sau de oprire. Punctele de cale modifică o rută direcționând-o prin locația specificată. API acceptă puncte de acces pentru aceste moduri de călătorie: conducere, mers și bicicletă; nu tranzit. Puteți specifica waypoint-uri folosind următoarele valori:

**alternatives** - Dacă este setat pe adevărat, specifică faptul că serviciul Direcții poate oferi mai multe alternative de rute în răspuns. Aceasta furnizarea de rute alternative poate crește timpul de răspuns de la server. Aceasta este disponibilă numai pentru cereri fără puncte intermediare.

**avoid** - Indică faptul că traseul (traseele) calculat (e) trebuie să evite caracteristicile indicate. Acest parametru acceptă următoarele argumente:

**tolls** - Indică faptul că ruta calculată ar trebui să evite drumurile / podurile cu taxă.

**highways** - Indică faptul că ruta calculată ar trebui să evite autostrăzile.

**ferries** - Indică faptul că ruta calculată ar trebui să evite feriboturile.

**language** – Indica limba în care să obțin rezultatele.Limba preferată are o influență mică asupra setului de rezultate pe care API alege să le returneze și asupra ordinii în care sunt returnate. Geocoderul interpretează prescurtările în mod diferit în funcție de limbă, cum ar fi prescurtările pentru tipurile de stradă sau sinonime care pot fi valabile într-o limbă, dar nu în alta.

**units** - Specifică sistemul de unități pe care trebuie să îl folosești la afișarea rezultatelor.

**region** - Specifică codul regiunii, specificat ca o valoare cu două caractere.

**arrival\_time** - Specifică ora de sosire dorită pentru direcțiile de tranzit, în câteva secunde de la miezul nopții, 1 ianuarie 1970 UTC. Puteți specifica fie plecare\_ ora sau sosire\_ ora, dar nu ambele. Rețineți că sosirea\_timp trebuie specificată ca un număr întreg.

**departure\_time** - Specifică ora de plecare dorită. Puteți specifica ora ca număr întreg în câteva secunde de la miezul nopții, 1 ianuarie 1970 UTC. Alternativ, puteți specifica o valoare de acum, care stabilește ora de plecare la ora curentă (corectă la cea mai apropiată secundă).

## 3.6. Tehnologii auxiliare

## 3.6.1Postman

Postman este o unealtă foarte importantă in dezvoltarea, consumarea si testarea resurselor folosind HTTP. In general este folosit pentru web services dar este destinat oricarui tip de API.

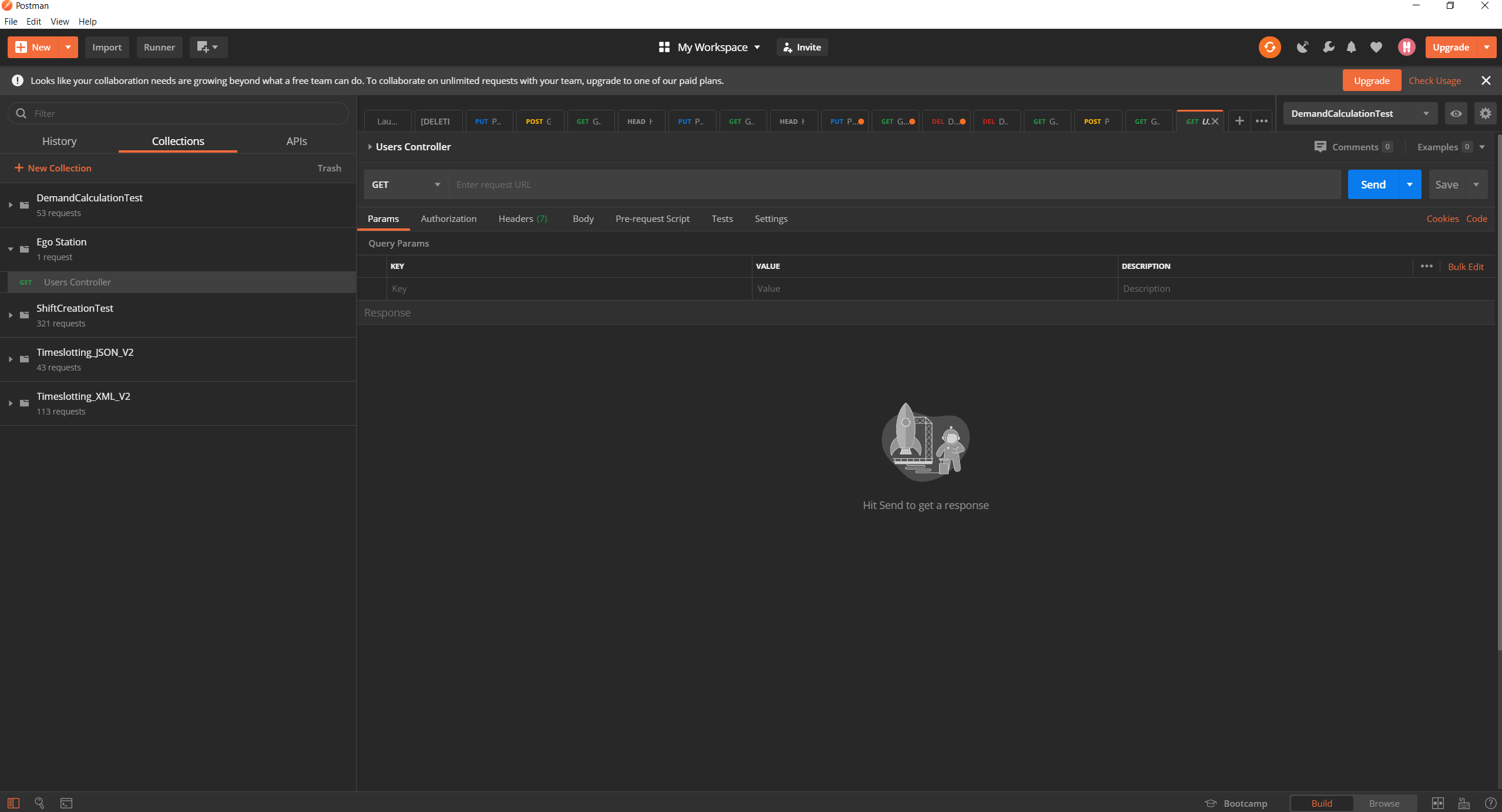
Fiind construit folosind framework-ul “Electron”, Postman are o interfață foarte prietenoasă și intuitivă, permitand utilizatorilor sa isi creeze colectii, viariabile, teste si scripturi , atat într-un mediu personal cât și într-un mediu imparțit cu toți membrii echipei. De aici putem spune ca are atât un caracter individual cât si unul dedicat companiilor. 

Fig.3.7. Interfața Postman

## 

## 3.6.2 Visual Studio

Visual Studio este un IDE (integrated development environment) proprietar Microsoft , folosit pentru a dezvolta programe, site-uri web, servicii web, sau aplicații mobile. Este foarte bine integrat cu ecosistemul .NET dar suportă foarte multe alte limbaje si tehnologii, producând atât cod nativ cat si cod interpretat.

Deasemenea, alt aspect important este editorul de text care beneficiază de InteliSense (componenta de auto completare) si de refactorizare automata. Printre celelalte functionalitati putem enumera personalizarea codului, meniul pentru creearea interfetelor, meniul pentru designul paginilor web, creere si vizualizare pentru clase si baze de date.

Există foarte multe plug-in-uri care pot fi aplicate ce pot schimba si îmbunatații diferite aspecte si este foarte puternic înclinat catre partea de source control având implementate functionalitati ce includ Team Foundation Server, Git, Subeversion..etc.

In total, in Visual Studio putem scrie in 36 limbaje de programare diferite atat pentru compilare cat si pentru editare si InteliSense.

La baza, arhitectura acestui IDE este una modulara , fiind defapt o platforma la care se pot adăuga functionalități ( ca utilizarea unui limbaj de programare) prin adăugarea lor ca serviciu, sub forma de plug-in-uri numite VSPackage.

Istoric versiuni:

* Visual Studio 4.0 - a fost primul produs de acest gen , în principal scopul lui a fost integrarea cu Windows 95
* Visual Studio 97 - numit și Cod Boston a fost oferit ca Professional sau Enterprise si aducea inovații ca Visual Basic 5.0, Visual C++ 5.0, Visual J++ sau FoxPro 5.0
* Visal studio 6.0 - a fost lansat in 1998 și era numit Cod Aspen. Includea atunci suport pentru Internet Exporer 4.0, control HTML și au fost introduse pentru prima data tehnologii ca Active Directory sau Queue Server.
* Visual studio 2002 - a fost lansat in 2002 și era numit cod Rainier fiind lansat în mai multe variante și intrand în istorie pentru aparitia pentru prima data a limbajului C#.
* Visual studio 2005 - lansat in 2005 si numit cod Whidbey, a fost un update major de la predecesorul sau introducând compilarea pentru x86-64
* Visul Studio 2008 - introduce .NET Framework ca tehnologie de bază
* Visual Studio 2010/2012/2015/2017 sunt ultimele versiuni, care au adus foarte multe functionalitați ca: Integrarea cu platforme de Cloud ca Azure, introducerea de noi versiuni de C# , programare de jocuri, dezvoltare pentru dispozitive mobile, unelte pentru inteligența artificială dar si multe alte transformari utile.[14]

## 3.6.3MongoDB Compass

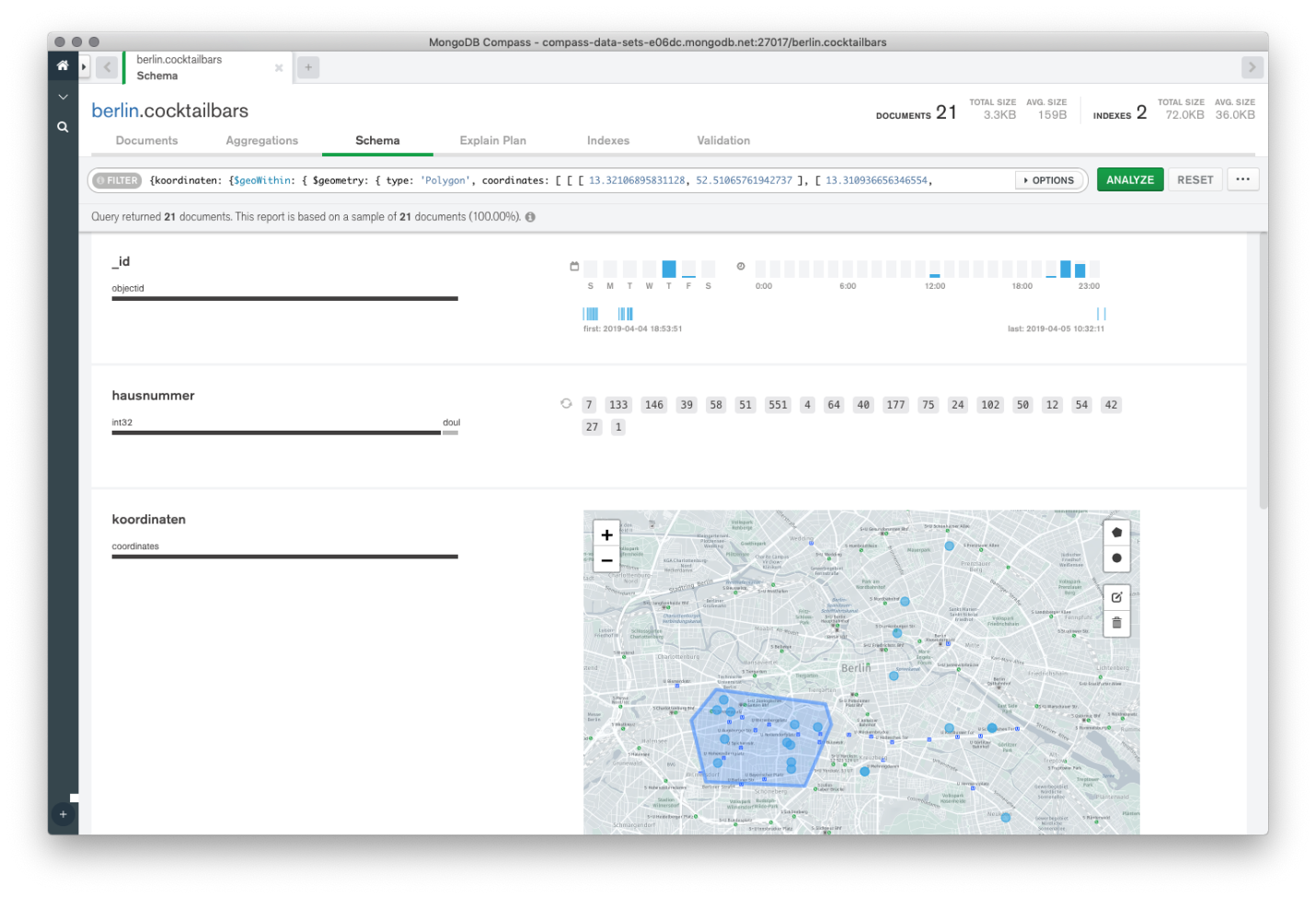


Fig 3.8. Interfața MongoDB Compass

Compass este un GUI (Graphical user interface) adica o interfată care transpune o cale de a comunica cu entitatile din această bază de date mult mai usor. Executa interogări ad hoc în câteva secunde. Interacționează cu datele tale cu funcționalitatea CRUD completă.Folosirea lui oferă posibilitatea de a vizualiza și optimizaț performanțele interogării. Este disponibil pe Linux, Mac sau Windows. Compass permite decizii mai inteligente despre indexare, validarea documentelor și multe altele.Permite vizualizare rapidă și explorarea schemei pentru a înțelege frecvența, tipurile și intervalele de câmpuri din setul de date.

Caracteristici:

* Statisticile serverului în timp real permit să vizualizați valorile serverului cheie și operațiunile bazei de date. Afișarea statisticilor cu ușurință pentru operațiunile bazei de date.
* Modificari pentru documentele existente cu mai multă încredere utilizând editorul vizual intuitiv, sau introducerea dedocumente noi și clonari sau ștergerea cele existente în doar

câteva clicuri.

* Informarea in perspectivă pentru cum se execută interogările printr-o GUI ușor de înțeles, care vă ajută să indentificarea și să rezolvarea problemele de performanță.
* Înțelegerea tipul și dimensiunea indexurilor, utilizarea lor și proprietățile speciale. Adăugare și eliminare indexuri cu un singur clic.
* Posibilatea de a scrie reguli de validare a schemei JSON într-un editor inteligent care sugerează automat nume de câmp, tipuri de date BSON și cuvinte cheie de validare. Cu o previzualizare în direct a documentelor care trec și nu reușesc regula de validare, este ușor de văzut dacă regula are comportamentul dorit.
* Compass Plugin Framework este expus ca API, ceea ce îl face extensibil de către utilizatori.
* Construirea conductelor de agregare într-o interfață de utilizator intuitivă. Scheletele de cod și completarea automată simplifică construirea unei etape, în timp ce o previzualizare a documentelor va arata dacă stadiul face ceea ce este nevoie. Adăugare etape, eliminare sau tragerea și plasarea pentru a re-comanda în conductă.

## 3.6.4 Visual Studio Code

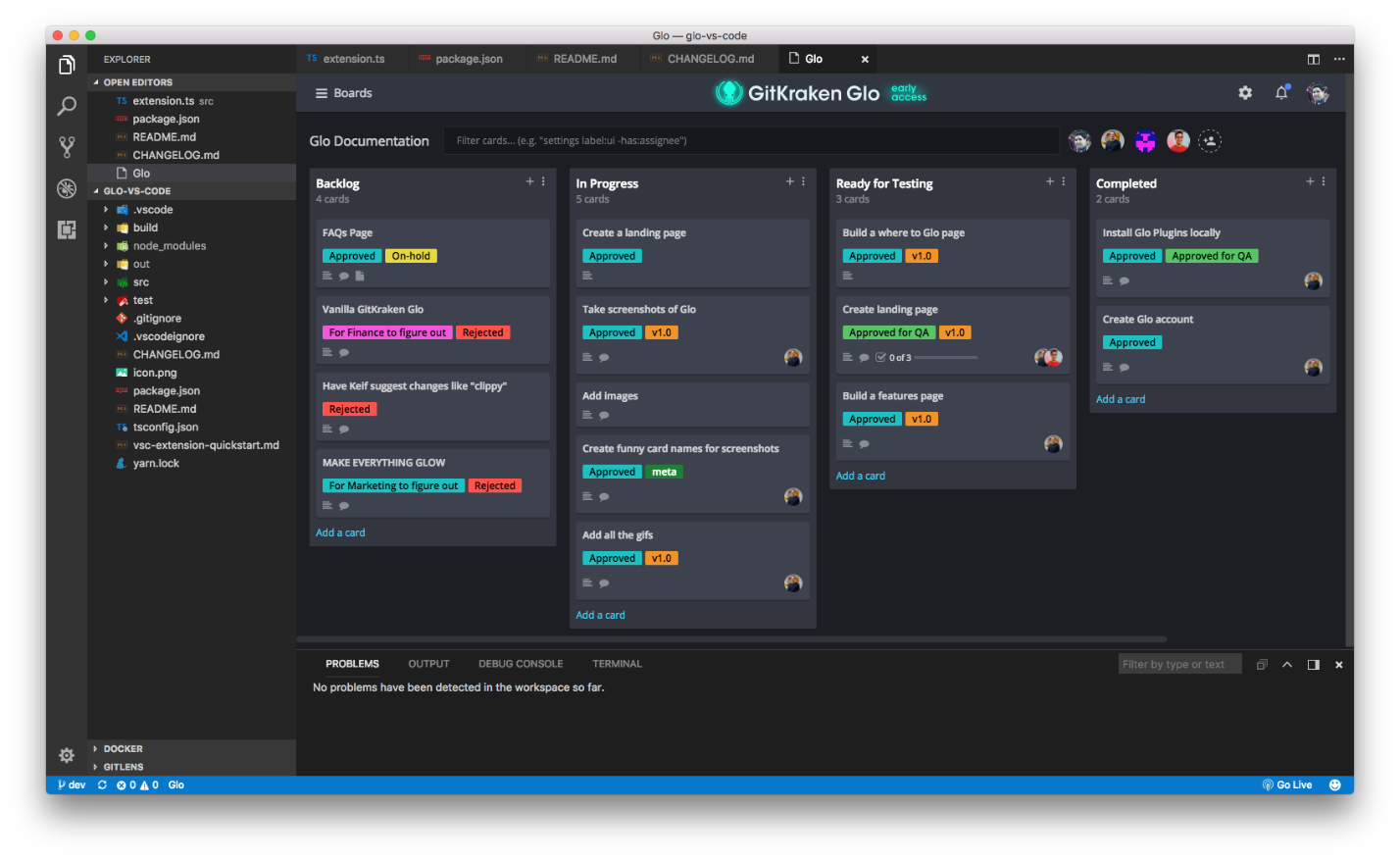


Fig. 3.9 IDE-ul Visual Studio Code

Visual Studio Code este un editor de cod sursă gratuit realizat de Microsoft pentru Windows, Linux și macOS.Caracteristicile includ suportul pentru depanare, evidențierea sintaxei, completarea inteligentă a codului, fragmente, refactorizarea codului și Git încorporat. Utilizatorii pot schimba tema, comenzile rapide de la tastatură, preferințele și pot instala extensii care adaugă funcționalitate suplimentară. Codul sursă este gratuit și open-source, eliberat sub Licența MIT permisivă.Binarele compilate sunt gratuite pentru orice utilizare.

Visual Studio Code a fost anunțat pe 29 aprilie 2015 de către Microsoft în cadrul conferinței Build 2015. O compilare Preview a fost lansată la scurt timp după aceea. Pe 18 noiembrie 2015, Visual Studio Code a fost lansat sub licența MIT și codul sursă a acestuia a fost postat pe GitHub. De asemenea, a fost anunțat sprijin pentru extindere. Pe 14 aprilie 2016, Visual Studio Code a absolvit etapa de previzualizare publică și a fost lansat pe web.

Visual Studio Code este un editor de cod sursă care poate fi utilizat cu o varietate de limbaje de programare, inclusiv Java, JavaScript, Go, Node.js și C ++. Se bazează pe cadrul Electron, care este utilizat pentru a dezvolta aplicații web Node.js care rulează pe motorul de layout Blink. Visual Studio Code folosește aceeași componentă de editor (denumită în nume „Monaco”) folosită în Azure DevOps (numită anterior Visual Studio Online și Visual Studio Team Services).

În loc de un sistem de proiect, permite utilizatorilor să deschidă unul sau mai multe directoare, care pot fi apoi salvate în spațiile de lucru pentru reutilizarea viitoare. Acest lucru îi permite să funcționeze ca editor de coduri agnostice pentru orice limbă. Acceptă o serie de limbaje de programare și un set de caracteristici care diferă pentru fiecare limbă. Fișierele și folderele nedorite pot fi excluse din arborele de proiect prin setări. Multe funcții ale Codului Visual Studio nu sunt expuse prin meniuri sau interfața de utilizator, dar pot fi accesate prin intermediul paletei de comenzi.

Visual Studio Code poate fi extins prin extensii, disponibil printr-un depozit central. Aceasta include completări la editor și asistență lingvistică. O caracteristică notabilă este capacitatea de a crea extensii care să adauge suport pentru limbi, teme și depanatoare noi, să efectueze o analiză de cod static și să adauge linteruri de cod utilizând Protocolul Server Server.

Visual Studio Code include mai multe extensii pentru FTP, permițând utilizarea software-ului ca alternativă gratuită pentru dezvoltarea web. Codul poate fi sincronizat între editor și server, fără a descărca niciun software suplimentar.

Visual Studio Code permite utilizatorilor să stabilească pagina de cod în care este salvat documentul activ, caracterul newline și limbajul de programare al documentului activ. Acest lucru îi permite să fie utilizat pe orice platformă, în orice locale, și pentru orice limbaj de programare dat.

## 3.6.5 Swagger Swashbuckle

Swagger este un software open-source susținut de un ecosistem mare de instrumente care ajută dezvoltatorii să proiecteze, să construiască, să documenteze și să consume servicii web RESTful. În timp ce majoritatea utilizatorilor identifică Swagger cu instrumentul Swagger UI, setul de instrumente Swagger include suport pentru documentație automată, generare de coduri și generare de cazuri de testare.

La crearea API-urilor, instrumentul Swagger poate fi utilizat pentru a genera automat un document Open API bazat pe codul însuși. Aceasta încorporează descrierea API în codul sursă al unui proiect și este denumită în mod informal cod de dezvoltare sau de jos în jos dezvoltare API.

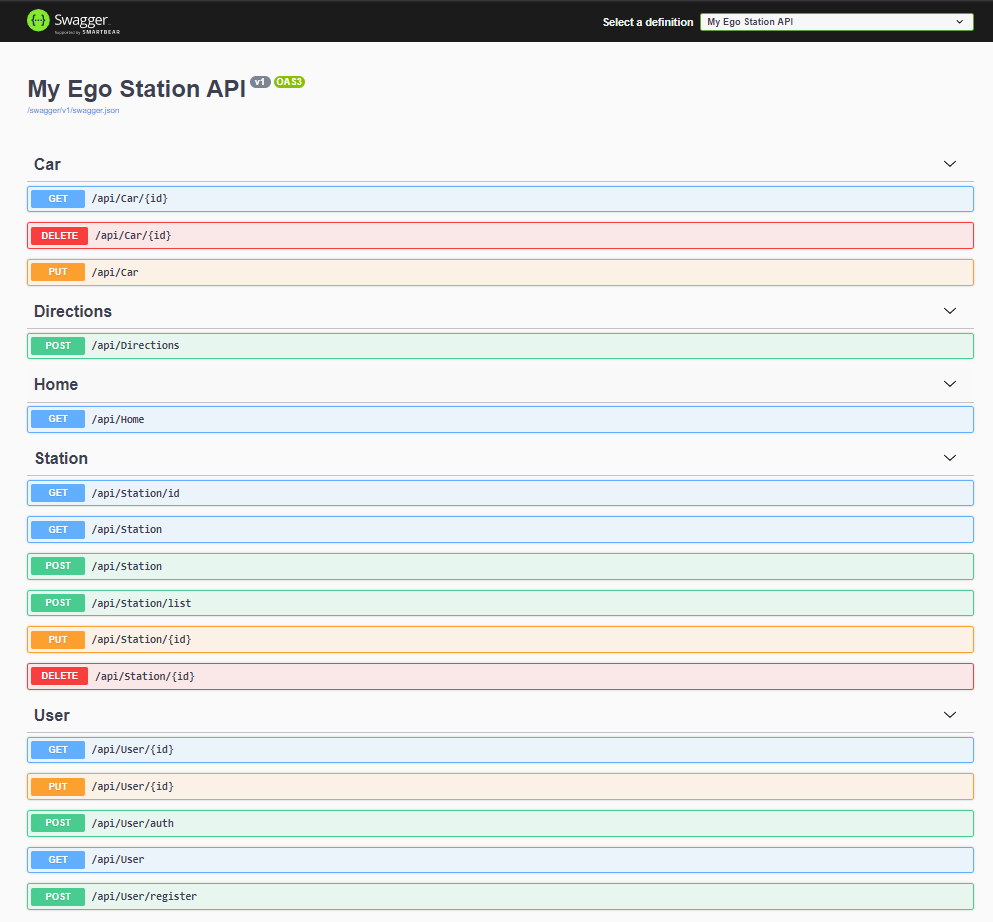


Fig 3.10 Swaggerul generat pe baza proiectului

## 3.7. Rezultate obtinute

# CAPITOLUL 4. CONCLUZII SI DEZVOLTARI ULTERIOARE

# BIBLIOGRAFIE

1. *THE GLOBAL SMARTPHONE REPORT: The forces behind the global deceleration in smartphone sales ,* <https://www.idc.com/promo/smartphone-market-share/os>
2. *DEJA UN TREND: propulsia hibridă-electrică,* <https://www.promotor.ro/masini-noi/dosar-analize/deja-un-trend-propulsia-hibrida-electrica-11727417>
3. *History of the electric vehicle,* https://en.wikipedia.org/wiki/Hystory\_of\_the\_vehicl
4. *Unde incarci o masina electrica in 2019,* <https://0-100.ro/2019/01/22/unde-incarci-o-masina-electrica-in-2019-si-cat-costa-e-romania-pregatita-pentru-electromobilitate/>
5. *Android (operating system),* <https://en.wikipedia.org/wiki/Android>
6. *Global Smartphone Market Share,* <https://www.counterpointresearch.com/global-smartphone-share/>
7. *Global EV Outlook,* <https://www.iea.org/reports/global-ev-outlook-2020>
8. *Sales rates of electric vehicles GLOBAL,* <https://www.iene.eu/heres-how-electric-cars-will-cause-the-next-oil-crisis-p3240.html>
9. *Hoog, A., 2011, Android Forensics Investigation, Analysis and Mobile Security for Google Android, Elsevier, Inc.,* Capitolele 2-3
10. *Reto Meier 2010, Andorid 2 Application Development,* Capitolele 5-10
11. *Azure Documentation,* <https://docs.microsoft.com/en-us/azure/?product=featured>
12. *An overview of HTTP,* <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
13. *MongoDB Docs,* <https://docs.mongodb.com/cloud/>
14. *History Of C# Programming Language,* <https://www.c-sharpcorner.com/blogs/history-of-c-sharp-programming-language>

# ANEXE