UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Aplicație hibridă pentru platforme mobile**

Propusă de

**Andrei Secăreanu**

**Sesiunea**: Iulie, 2018

Coordonator științific

**Lect. Dr. Anca Vitcu**

**UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI**

**FACULTATEA DE INFORMATICĂ**

**Aplicație hibridă pentru platforme mobile**

**Andrei Secăreanu**

**Sesiunea**: Iulie, 2018

Coordonator științific

**Lect. Dr. Anca Vitcu**

DECLARAȚIE PRIVIND ORIGINALITATEA ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul „Aplicație hibridă pentru platforme mobile” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
* codul sursă, imagini etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
* rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași, 2018

Absolvent Andrei Secăreanu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de accord ca Lucrarea de licență cu titlul „Aplicație hibridă pentru platforme mobile” codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

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

Iași, 2018

Absolvent Andrei Secăreanu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acord privind proprietatea dreptului de autor**

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, format executabil și sursă, să aparțină autorului prezentei lucrări, Andrei Secăreanu.

Încheierea acestui acord este necesară din următoarele motive:

Iași, 2018

Decan Adrian Iftene Absolvent Andrei Secăreanu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cuprins

**Introducere1**

CAPITOLUL I APLICAŢII MOBILE HIBRID VERSUS NATIV4

**1.1 Aplicația hybrid** 5

**1.2 Aplicația nativă Android** 7

**1.3 Aplicația nativă iOS** 8

**1.4 Concluzii** 9

CAPITOLUL II APLICAŢIA MOBILĂ11

**2.1 Prezentarea aplicație**.................................................................................................11

**2.2 Autentificarea**…………………………………………………………………………………………………………………12

**2.3 Meniul**15

**2.4 Tema vizuală – design, icons**16

**2.5 Modele de date și formulare**18

**2.6 Pagini**19

**2.7 Componentele**25

**2.8 Concluzii**...............................................................................................................................27

**CONCLUZII FINALE...........................................................................................................28**

Anexe TEHNOLOGIILE DE TIP EDGE30

**1.1 Tehnologii web – HTML, CSS, JavaScript**30

**1.2 Prepocesoare** 31

**1.3 Cordova**32

**1.4 Angular**.......................................................................................................................33

**1.5 Ionic**35

**1.6 RxJS**36

**1.7 Firestore**38

**1.8 Concluzii**42

**Bibliografie43**

Introducere

Odată cu dezvoltarea tehnologiei, în lumea mașinilor de calcul, tot mai multe persoane au găsit un interes pentru web și pentru aplicțiile mobile. Aceste două tehnologii au fost îmbrățișate la scară largă și au ajuns să fie industriile de top la nivel mondial.

Aplicatiile web si mobile sunt disponibile la nivel larg prind diverse dispositive hardware, cele mai folosite fiind calculatoarele personale, laptopurile si telefoanele inteligente. Conform *”DIGITAL AROUND THE WORLD”*în ianuarie 2018 pe glob sunt aproximativ 7.6 miliarde de oameni, dintre care aproximativ 4 miliarde folosesc internetul și 5 miliarde folosesc telefoanele mobile inteligente.(Fig. 1)

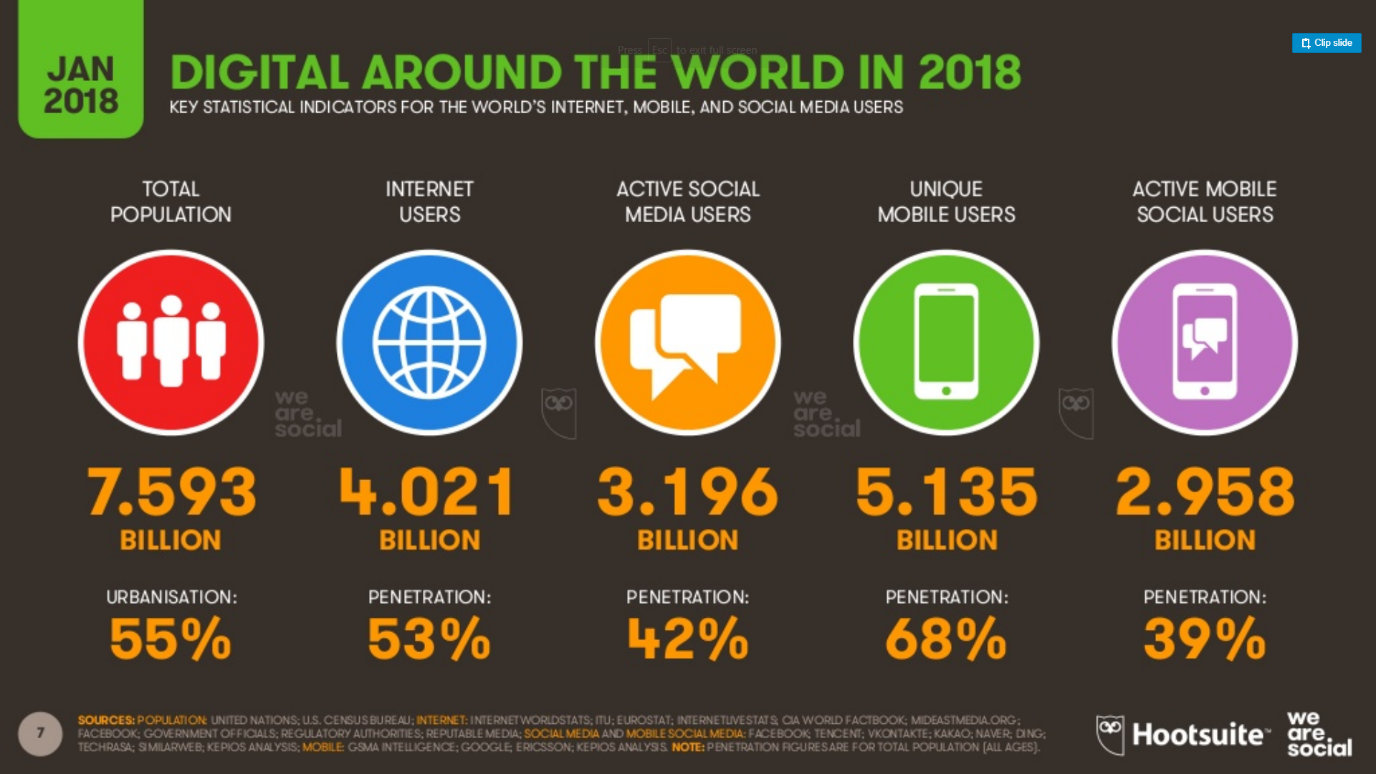


Fig. 1 – Numarul de utilizatori la nivel global al telefoanelor inteligente [3]

Numărul foarte mare de telefoanele mobile inteligente pe piața de consum se datorează faptului că au câteva mari avantaje față de alte dispositive atât pentru utiliztororul aplicațiilor, cât și pentru dezvoltatorul aplicațiilor.

Câteva avantaje pentru utilzatorul aplicațiilor sunt:[4]

* fiecare telefon inteligent are incapsulat o baterie, iar acesta poate fi folosit pentru o perioadă de timp fără a fi conectat la o sursă de energie;
* telefoanle inteligente au o dimensiune redusă, iar acestea poate fi folosite oriunde de către utilizatori;
* dispozitivele au un mod de funcționare foarte intuitiv, prin ecranele tactile, iar orice tip de utilizatorul poate să-l folosească foarte ușors;
* prețul accesibil pentru telefoanele mobile inteligente;
* prețul accesibil pentru aplicații;

Câteva avantaje pentru dezvoltatorul de aplicații:

* prețul accesibil pentru dispositive;
* documentație pentru limbajul de programare specific sistemului de operare;
* existența unei piețe virtuale unde fiecare dezvoltator de aplicații își poate vinde produsul, către utilizatorul final;
* existența unor unelte de dezvoltare și testare a plicațiilor;

Aceste telefoane mobile inteligente sunt diversificate astfel încât, cei mai mari producători de despozitive sunt Apple, Samsung și Windows. Acestea sunt într-o continua dezvoltare la nivel hardware si design, iar utilizatorii achiziționează un nou dispozitiv intre 1 și 3 ani.

De asemenea la nivel de soft telefoanele mobile inteligente se dezvoltă cu o rapiditate acerbă. Fiecare dintre aceste folosesc un sistem de operare diferit și sunt construite pe mai multe tehnologii, iar ele sunt folosite cum urmează: Apple folosește Swift, Samsung folosește Android și Windows folosește Windows Phone. Principalele limbaje de programare pentru fiecare tehnologie sunt prezentate în tabelul de mai jos.(Fig. 2)

|  |  |  |
| --- | --- | --- |
| Telefoane inteligente | Tehnologii | Principalele limbaje de programare |
| Apple | Swift | Objective-C |
| Samsung | Android | Java |
| Windows | Windows phone | C# |

*Fig. 2 – Tabelul tehnologiilor si limbajelor în funcție de dispozitiv*

În această piață a plicațiilor mobile, diversitatea este una foarte mare. Există foarte multe tipuri de aplicații precum: aplicații de comunicare, jocuri, cărți electronice, filme, aplicații de business, educaționale, muzică, cumpărături online și altele.3

Conform tuturor aspectelor prezentate anterior, doresc implementarea unei aplicații mobil hybrid, în care un dezvoltator să scrie o singură data sursa de cod a plicației, iar la compilare să ruleze pe mai multe dispositive diferite, pe mai multe sisteme diferite și pe mai multe limbaje de programare. Aplicația va intra în domeniul comunicării, astfel încât, fiecare profesor al facultății de informatică să comunice mai ușor cu studenții, iar fiecare student să comunice mai eficient si ușor cu profesorii.   
Aplicația este scrisă cu ajutorul tehnologii web, folosind cele mai noi tehnologii. Principalele limbaje web folosite sunt: HTML, CSS și JavaScript, iar principalele tehnologii web sunt: TypeScript, Ionic, Cordova, Angular, Firestore.

În urma studierii principalelor tehnologii web prezentate mai sus am ales să elaborez această aplicație pentru Facultatea de Informatică Iași. Modul de rulare al acesteia și cum funcționează principalele tehnologii folosite sunt prezentate pe parcursul întregii lucrări de licență.

Trăind într-o eră în continuă dezvoltare, în care telefonul mobil este nelipsit atât elevilor cât și studenților, aplicațiile au luat un avânt uriaș și au un impact imens în viața celor ce le folosesc. Atât cele de tip social-media, dar și aplicații informative sau de tip calendar sunt foarte căutate și solicitate de către utilizatori. Consider că studenții, colegii mei din genrațiile care vin acum în spate, vor găsi foarte utilă o aplicație care să le dea acces asupra notelor și informațiilor care se întamplă în cadrul facultății în timp real, fără a mai fi necesară deplasarea până la facultate pentru a verifica panoul care afișează notele sau mediile la sfârșit de sesiune.

Fiind o facultate foarte dinamică, diversă și în continuă dezvoltare, Facultatea de Informatică va putea oferi într-un timp mult mai real și mai eficient informații cu privire la ateliere, cursuri, situații de sfârșit de an și multe alte infotmașii extrem de utile și necesare, prin ajutorul unei aplicații.

Această aplicație a fost creată să ușureze viața studenților cât și a profesorilor, care vor putea mult mai ușor înstiințta pe studenți anumite schimbări care vor avea loc la cursuri.

Capitolul 1 - creioniază informații referitoare la aplicațiile hybrid și native (iOS și Android) cât și definițiile acestora. Tot aici vor fi prezentate avantajele și dezavantajele de a folosi aplicații hybrid în defavoarea aplicațiilor native.

Capitolul 2 prezintă în detaliu funcționalitățile aplicației și utilizarea frameworkului Ionic și cum acesta rezolvă probleme pentru aplicație ca să fie folosită pentru cele 2 mari sisteme de operare Android și iOS.

Anexa cuprinde buchetul de tehnologii folosite în aplicație cât și o scurtă descriere acestora. Principalul framework fiind Ionic, care este bazat pe Cordova și Angular.

CAPITOLUL I APLICAŢII MOBILE HIBRID VERSUS NATIV

Genrațiile de azi au devenit foarte independente de aplicațiile mobile. Așa că dezvoltarea aplicațiilor mobile sunt esențtiale pentru orice fel de afacere sau organizație. Sunt diferite moduri de a dezvolta aplicațiile mobile, dar de asemenea sun și diferite frame-work-uri.

Corporațiile care au nevoie de o aplicație mobilă au la dispoziție trei opțiuni:

1. Aplicațiile native pot fi construite folosind limbajul de programare nativ al dispozitivului. De exemplu aplicațiile Android nu pot să funcționeze în iOS și aplicațiile Blackberry nu pot să ruleze pe Android, iar aplicațiile iOS nu pot rula pe platformele Windows și așa mai departe. Aplicațiile native sunt ditribuite prin platforma lor specifică cu magazinul de aplicații instalat direct pe dispozitiv.
2. Aplicațiile mobile rulează pe browser-ul dispozitivului și funcționează pe toate platformele. De exemplu o aplicație mobilă web funcționează la fel de bine și pe o platformă iOS, Blackberry dar și pe telefoanele cu platforme Windows. Spre deosebire de aplicașiile native, aplicațiile mobile sunt distribuite prin intermediul web-ului și nu sunt instalate propiu-zis pe dispozitiv. Este important de remarcat diferența dintre o aplicație mobilă web și o pagină web. Deși sunt adeseori încurcate aplicațiile mobile web si paginile web sunt lucruri foarte diferite. O aplicație mobilă web este o pagină statică HTML/CSS/JavaScript formatată pentru a fi folosită pe un dispozitiv de tip telefon.
3. Aplicațiile hybrid sunt trecerea dintre aplicații native și aplicațiile telefoanelor de tip web. O aplicație hybrid este o aplicație web acoperită intr-o platformă specifică. Această platformă specifică îi oferă aplicației native anumite calități cum ar fi portabilitatea, instalarea nativă și ditribuirea ei într-un magazin de tip ”app store” sau ”market ditribution.” (Ketan Anant More, 2006).

În generația de azi, dezoltarea plicațiilor mobile este un lucru extrem de esnțial. Dezvoltarea de aplicații mobile este crucială pentru multe dintre corporații. Există aplcații mobile Native, Hybrid și aplicații mobile de tip web, în capitolul ce urmează ne vom axa pe aplicațiile Native și Hybrid și cum sunt dezoltatea acestea.

Aplicațiile Native au documente în sistem binar executabil descăract direct pe dispozitiv și stocată local. Procesul de instalare poate fi inițiat de către utilizator, sau, în unele cazuri de către departamentul IT al unei organizații. Cel ami popular mod de a descărca aplicațiile native este prina a vizita magazinele on-line ale sistemelor de oprarare (Appstore, Andorid Marketplace sau Blackberry App World) dar mai pot exista și alte metode care pot fi uneori asigurate de furnizorul rețelei.

Aplicațiile hybride combină dezvoltarea aplicațiilor native cu tehnologia web. Folosind această abordare, dezvoltatorii scriu mare parte din cod în cross-platform web tehnologii, în timp ce mențin accesul direct la aplicația nativă API când este nevoie. (IBM, 2012).

Este foarte greu pentru un simplu utilizator să detecteze dacă o aplicație este construită hibrid sau nativ. Aplicațiile hibrid, dacă sunt scrise bine, nu ar trebui să facă nici o diferență față de una nativă. În general utilizatorii la scară largă nu au cunoștințe despre aplicații native sau hibride. Pe aceștia îi interesează doar ca aplicația să ruleze rapid, fără erori și să corespundă descrierii.

O aplicație mobilă poate fi construită nativ sau hybrid. O aplicație nativă este construită în limbajul de programare, pentru Apple trebuie scrisă în Swift, Objective-C iar pentru Samsung (sau orice alt telefon care folosește sistemul Android) în Java.

O aplicație nativă poate accesa fără limite orice parte soft și hard a platformei sale, fără să aibă limitări sau restricții. Aceste aplicații nu au nevoie de alte containere sau pachete ca să facă legătura dintre cod și platformă. Fiecare aplicație nativă rulează doar pe platforma proprie, fără să acceseze altă platformă. Aceste aplicații rulează mult mai rapid, iar uneori utilizatorii, în funcție de complexitatea aplicației pot simți o mică optimizare față de o aplicație hibrid.

Aplicațiile hibrid, sunt o combinație între siteuri și aplicațiile native. O aplicație hibridă este scrisă în limbaje web: HTML, CSS, JavaScript. După ce codul a fost scris, această aplicație trebuie compilată prin inermediul unui soft care face legătura dintre tehnologiile web și tehnologia folosită în sistemul de operare nativ.

Aplicațiile hibrid pot simula designul aplicațiilor native, astfel încât utilizatorul final să nu facă diferența dintre o aplicație nativă sau hibrid.

* 1. **Aplicația hybrid[7]**

Aplicațiile mobile se aseamană cu aplicațiile native. Din punct de vedere al utilizatorului acestea nu au nici o diferență față de cele native. Aceste aplicații se găsesc în magazinele online și se instalează ca orice altă aplicație. Aplicațiile hibrid sunt de diferite tipuri, ele pornind de la jocuri, la aplicații de comunicare între persoane, până la aplicații ce folosesc partea hardware a dispozitivelor: camera foto, gps și altele.[2]

La fel ca și siteurile, aplicațiile mobile hibrid sunt construite cu ajutorul tehnologiilor web: HTML, CSS și JavaScript. Pentru ca aceste aplicații să ruleze pe dispozitive este necesar de o legătură cu limbajele native. Aici inervine un WebView, care este un container ce face legătura dintre tehnologiile web și cele native, acesta mai este numit generic și browser.[1]

Cea mai cunoscută legătură pentru aplicațiile mobile hibrid și accesul la limbajul nativ este Cordova. Cordova este un container care la compilarea unui cod, scris în tehnologii web, îl împachetează intr-o aplicație ce îi oferă acces la platforma nativă.

Câteva avantaje atunci când se crează aplicații native:

* o aplicație hibrid are nevoie de dezvoltatori care să cunoscă doar tehnologii web, fără a învăța alte limbaje native specifice platformelor;
* aplicațiile create hibrid, după ce au fost compilate în limbajul nativ, pot fi distribuite în magazinel online;
* salariile dezvoltatorilor de aplicații hibride sunt mai mici față de dezvoltatorii nativi;
* scade timpul de lucru pentru crearea de aplicații similare care să acopere cât mai multe platforme;
  1. **Aplicația nativă Android**

****

Fig. 1.1 - Sistemul de operare Android[14]

Android este un sistem de operare bazat pe Linux pentru dispozitive mobile, tablete, televizoare și alte dispozitive. Android este deținut de compania Google, iar cu fiecare versiune aceasta aduce îmbunătățiri și mentenanță.

Android este cel mai folosit și cunoscut sistem de operare la nivel global. Acesta este susținut de către dezvoltatori seniori și specializați la nivel de hardware și software, iar sistemul de operare este intr-o continuă dezvoltare.

Android pune la dispoziția dezvoltatorilor un sitem unificat de instrumente: Android Software Development Kit cunoscut sub abrevierea SDK, toolori de test, editoare speciale pentru Android, emulatoare și alte unelte pentru creare de aplicații. Dezvoltatorii vor putea crea aplicații native puternice, stabile, care să profite din plin de accesul la software și hardware.[9]

Android oferă o gamă largă de funcționalități pentru dezvoltator, așa încât orice aplicație poate avea ne numărate activități fără a fi limitate de sistemul de operare. Designul oferit de android este unul foarte simplu de înțeles și orice utilizator nou, care nu a mai avut experiență cu sistemul sau aplicațiile, poate să se adapteze ușor.

Android pune la dispoziția dezvoltatorului cât și a utilizatorului o piață de desfacere pentru aplicațiile mobile. Aici orice dezvoltator poate să-și pună noua aplicație, iar orice utilizator poate să o achiziționeze pe baza unui cont, eventual a unei plăți, dacă aplicația este gratuită sau nu.

Câteva avantaje când se crează aplicații native Android:

1. dezvoltatorul are acces la tot hardwareul dispozitivelor și nu are nevoie de alte tehnologii auxiliare;
2. aplicația rulează cu o viteză mai mare decât aplicații le hibrid, iar utilizatorul are o expierență mai bună;
3. documentația este foarte diversă și completă, mulți dezvoltatori discutând pe forumuri speciale pentru Android;
4. problemele pot fi rezolvate mult mai ușor;
   1. **Aplicația nativă iOS**

****

Fig. 1.2 – Sistemul de operare iOS[13]

iOS, numit anterior iPhone OS, este un sistem de operare dezvoltat de compania Apple specific doar pentru telefoanele inteligente de la aceeași companie. Principalele dispozitive care utilizează sistemul de operare iOS sunt iPhone, iPad și iPod. Pentru a dezvolta o aplicație pe sistemul de operare iOS orice dezvoltator are nevoie de un dispozitiv pe care să ruleze aplicația, care trebuie să fie tot de la compania Apple. Dezvoltatorul are nevoie de o mașină MacBook, Mac Mini sau orice altă mașină care să poată rula codul.[13]

Dezvoltatorul mai are nevoie de Xcode și de un cont de dezvoltator la compania Apple – care este gratuit. De asemenea, va putea rula codul aplicației pe un simulator sau pe un device real.

Xcode este editorul de bază a companiei Apple, iar acesta include tot ce are nevoie un dezvoltator pentru a construi o aplicație în sistemul de operare iOS.

Apple oferă la dispoziție un magazin online unde aplicațiile pot fi stocate de către dezvoltatori, iar utilizatorii pe baza unui cont vor putea să descarce sau să cumepere aplicațiile.

* 1. **Concluzii**

Deoarece piața aplicațiilor se extinde tot mai mult, iar tot mai multe dispozitive sunt folosite cu siteme de operare fie Android sau iOS, am decis să mă axez pe o aplicație hibrid. Aceasta va putea fi rulată pe mai multe platforme și va fi putea folosită de orice utilizator care are un dispozitiv care funcționează cu un sitem de operare descris anterior.

Aplicația va fi construită în frameworkul Ionic, acesta fiind un framework care este construit pe Cordova și Angular. Cordova este containerul care cuprinde tehnologiile web și le conectează cu cele native, în funcție de sistemul de operare. Angular este un framework complex de tehnologii web bazat pe HTML, SASS și TypeScript.

Pentru stocarea datelor voi folosi Firebase, noua versiune de baze de date de la google pentru aplicatii mobile și web. Această bază de date este în timp real.

Aplicația va fi de tip comunicare, în care profesorii și stundeții vor putea comunica mai bine informațiile pentru cursuri, examene, conferințe, teme și multe altele.

CAPITOLUL II APLICAŢIA MOBILĂ

**2.1 Prezentarea aplicației**

Acestă lucrare are ca scop crearea unei aplicații mobile hibrid sub numele de FII pentru diferite platforme. Implementarea acestei aplicații are la bază implementarea prin combinare a tehnologiilor descrise în capitolul anterior. Aplicația FII are la bază frameworkul Ionic care combină tehnologiile Angular și Cordova.

Ionic folosește pachetele npm, care sunt create și menținute de comunități de dezvoltatori. Aceste pachete sunt open-source și sunt reactualizate și modificate de caătre comunități.

Prin intermediu frameworkului Angular este creată logica, sunt captate și manipulate datele aplicției și sunt expuse elementele vizuale către utilizator. Prin intermediul componentelor este creată logica unei pagini sau a unei entități vizuale folosită ulterior în mai multe pagini. Componenta manipulează datele și le trimite către fișierele HTML.

Prin intermediul frameworkului Cordova se accesează elementele native ale platformelor în funcție de tipul acestora. Astfel se folosesc pluginuri specializate care interpretează codul tehnologiilor web și folosește partea pluginului specifică platformei pentru a accesa elementul nativ dorit. Pentru a face posibil acest lucru se folosește un WebView, care este o cale de acces a tehnologiilor web la elementele native ale platformei.

Ideea aplicației este ca orice profesor universitar din cadrul Facultății de Informatică, să comunice printr-un mod mai accesibil și rapid cu studenții. Astfel informațiile oferite de profesori studenților și viceversa să fie instante și actualizate, astfel încât orice student, la orice periodă din timp să aibă acces la informații. De asemenea studenții vor putea să comunice cu prin intermediul unui chat pus la dispoziție.

Aceste facilități puse la dispoziție sunt distribuite pe diverse pagini, fiecare cu topicul ei. Astfel avem pentru fiecare an câte o pagină: Anul I, Anul II și Anul III. Mai sunt alte două pagini, una este pentru informații generale, iar cea de a doua pagină este pentru informații primite de le secretariatul facultății.

**2.2 Autentificarea**

Accesul la aplicație se face pe baza unui cont. Contul de utilizator, pentru logare are un email și o parolă. Acestea sunt configurate din baza de date Cloud Firestore de către un administrator. Acesta va stabili user, parola și rolul utlizatorului prin intermediul unei interfețe oferite de consola bazei de date.

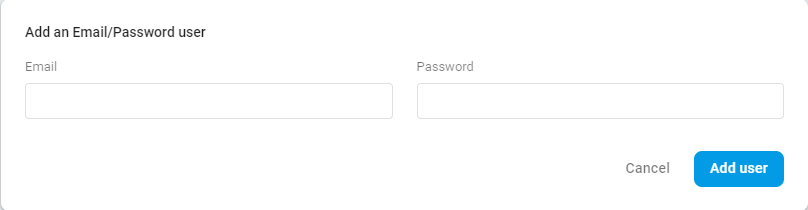


Fig 2.1 – Adăugare utilizator Firestore

Pentru stocarea utilizatorilor în baza de date Cloud Firestore are o bază de date specifică, iar principalele elemente stocate sunt email-ul, parola și un id generat dinamic. Aceste date vor fi folosite pentru ca Google să verifice atunci când un utilizator se logează.

Consola Cloud Firestore oferă o listă cu toți utilizatorii care pot avea acces la aplicație. Aici, adiministartorul poate să reseteza parola utilizatorilor, poate să activeze sau să dezactiveze un cont și poate să șteargă un utilizator din baza de date.

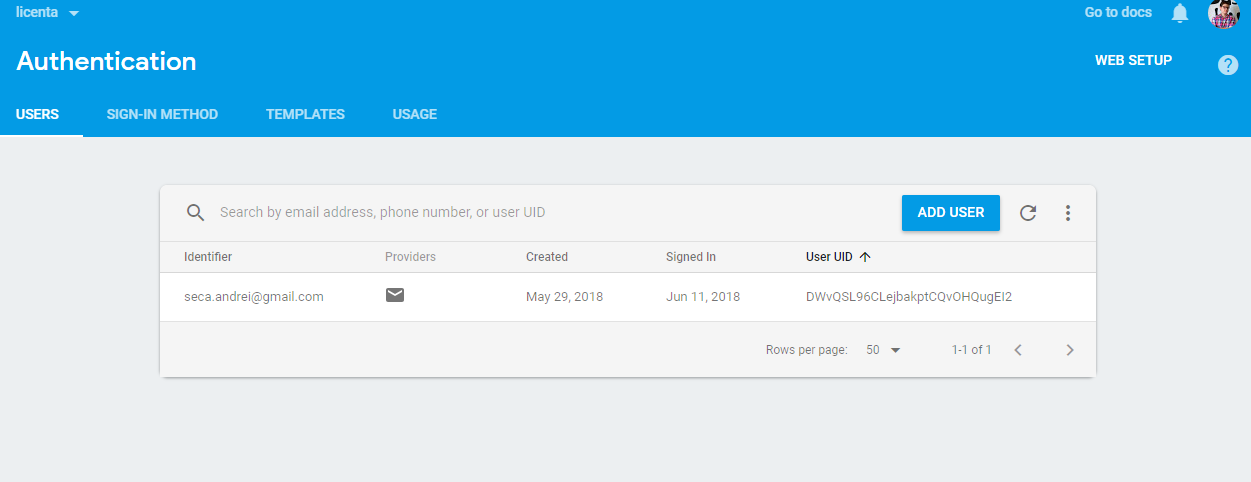


Fig 2.2 – Listă utilizatori Firestore

În baza de date pentru autentificare Firestore pune la dispoziția administartorului o serie de tipuri de autentificări. Acestea variază de la email, la autentificări cu conturi de social media, până la autentificaări cu numărul de telefon. Am ales să folosesc doar cea cu emailul, deoarece se pot folosi emailurile de la Faculteatea de informatică, iar utilizatorilor le va fi mai ușor să folosească platforma.

De asemenea am setat ca domeniul de acces al conturilor să fie pentru localhost, acesta fiind mediul folosit pentru dezvoltarea aplicației.

Pentru a stoca informații adiționale despre utilizatori, am creat o colecție în baza de date. Această colecție va conține documente, iar documentele vor stoca date adiționale despre utilizatori. Pentru a face legătura între baza de date de autentificare a Firestoreului și baza de date creată de administrator în colecție pentru utilizatori, fiecare document din colecția de utilizatori va avea id-ul generat de baza de date de autentificare.

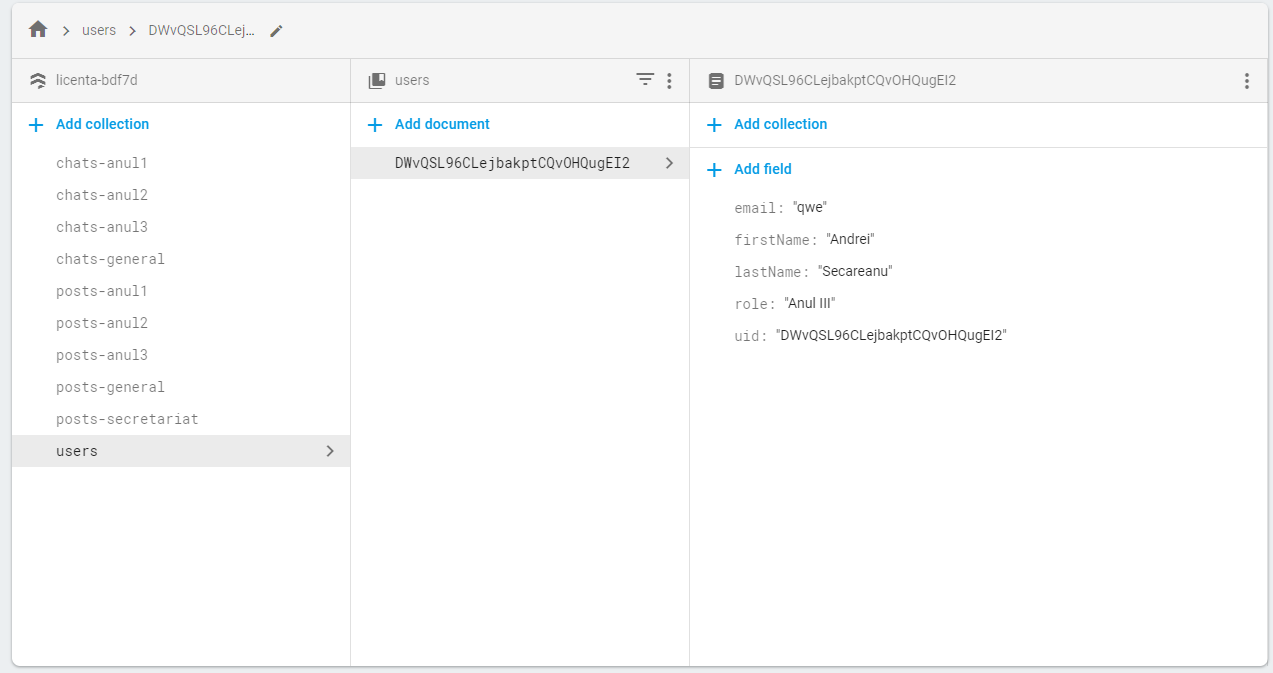
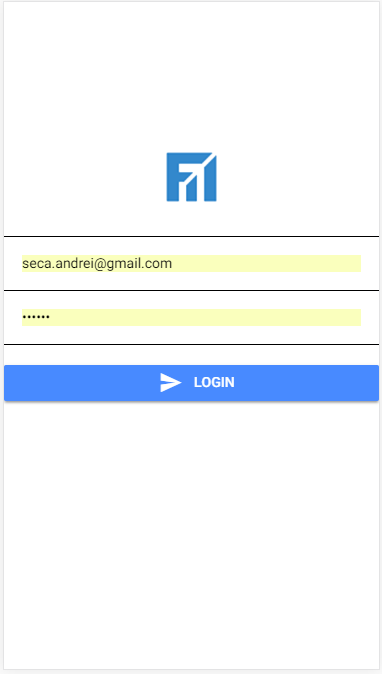


Fig 2.3 – Bază de date Firestore pentru utilizatori

Pe pagina de autentificare a plicației utilizatorul va vedea sigla facultății, va avea un formular cu două câmpuri unde va introduce emailul de utilizator și parola. De asemenea va avea un buton, iar prin evenimentul de apăsare va trimite datele formului spre a fi verificate pentru autentificare.

  
Fig 2.4 – Formular de logare

Formul de login este structurat pe o interfață și pe un formular reactiv din Angular. Interfața este construita din doua câmpuri: email și password separate printr-un chenar. După ce au fost introduce datele soecifice fiecărui utilizator, acesta va avea acces la aplicație prin apăsarea comenzii Login situate sub cele doua câmpuri menționate mai sus, conform Fig. 3.3.

Formul este un form group din Angular care are două form controls: email și password. Celor două controale le sunt adăugate validatori.

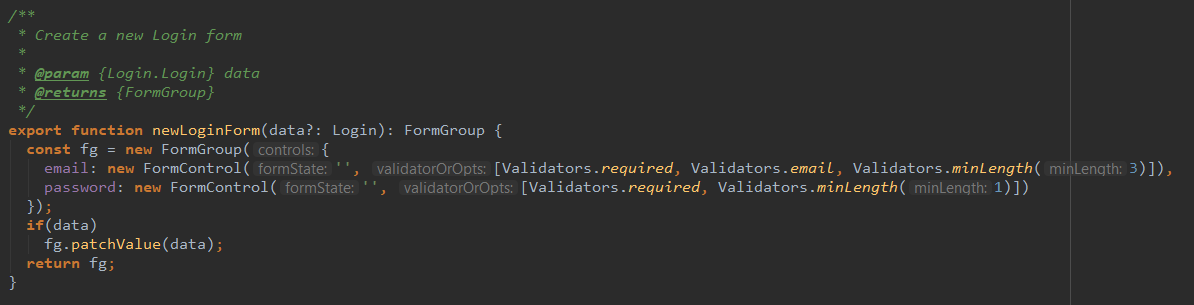


Fig 2.5 – Formular reactive din Angular pentru logare

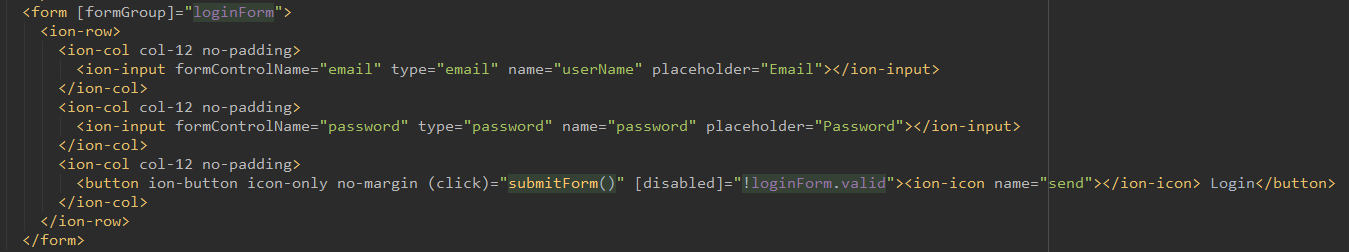


Fig 2.6 – Formularul în cod html cu componenete în Ionic

Formularul este conectat în fișierul HTML, printr tagul form. În interiorul formului se află în taguri de input form controls unde sunt făcute legăturile cu formularul.

**2.3 Meniul**

Sistemul de navigare al frameworkului ionic este bazat pe module. Aceste module încapsulează o pagină compusă din componentă și apgina HTML. Pentru creearea meniului s-a folosit un obiect ce conține titlul paginii, componenta care va fi activă atunci când este selectată pagina și iconița afișată utilizatorului.

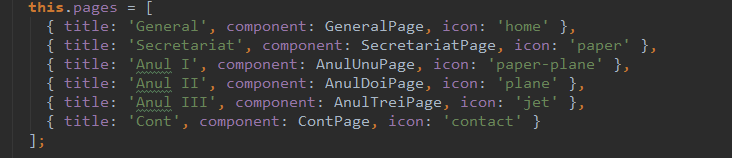


Fig 2.7 – Obiectul unde sunte stocate informațiile despre pagine

Meniul este format din șase pagini după cum urmează: General, Secretariat, Anul I, Anul II, Anul III și Cont.

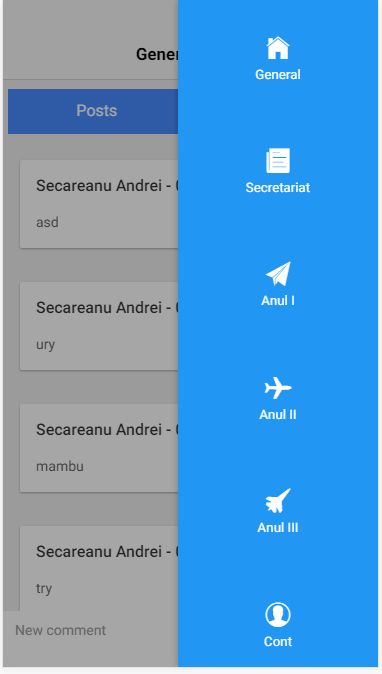


Fig 2.8 – Meniul în aplicație

Bara de meniu va fi oricând afișată în colțul din dreapta sus și poate fi accesată din orice pagină a aplicației.

Sistemul este bazat pe o pagină rădăcină. Aceasta va fi afișată prima în aplicație. În cazul aplicației „Fii” va fi afișată pagina de Log in. Dacă utilizatorul introduce corect credențialele pagina root se va seta pe General.

De aici frameworkul Ionic modifică paginile pe baza unei stive. Atunci când o nouă pagină este afișată se va introduce în stivă Componenta activă. Acest sistem de a naviga către următoarea pagină se numește Push. Iar atunci când utilizatorul închide o pagină sau dorește să se întoarcă la pagina anterioară pagina activă se închide și va apărea pagina anterioară, iar nou pagină va fi cea activă. Acest proces se numește Pull. Pagina rădăcină nu poate să fie închisă, dar poate să fie modificată

Pentru a naviga către o pagină, nu este nevoie ca aceasta să fie în meniu, iar ionic pune la dispoziție serviciul de navigare pentru a fi folosit oriunde în aplicație.

**2.4 Tema vizuală – designe, icons**

Frameworkul Ionic vine incorporat cu o temă structurată pentru telefoanel mobile. Tema conține elementele generale specifice unui telefon mobil, cum ar fi butonul de a închide o pagină pentru a vizualiza pagina anterioară, sau meniul care se deschide din partea dreaptă sau stăngă. De asemenea acesta conține o tematică anume pentru telefoanele mobile, astfel că simulează elementele native ale fiecărei platforme.

Ionic pune la dispoziție un simulator la nivelul browserului și oferă cele trei mari sisteme de operare pentru telefoane mobile, fiecare cu stilul său aparte. Astfel avem sistemul de operare Androi, iOS și Windows.

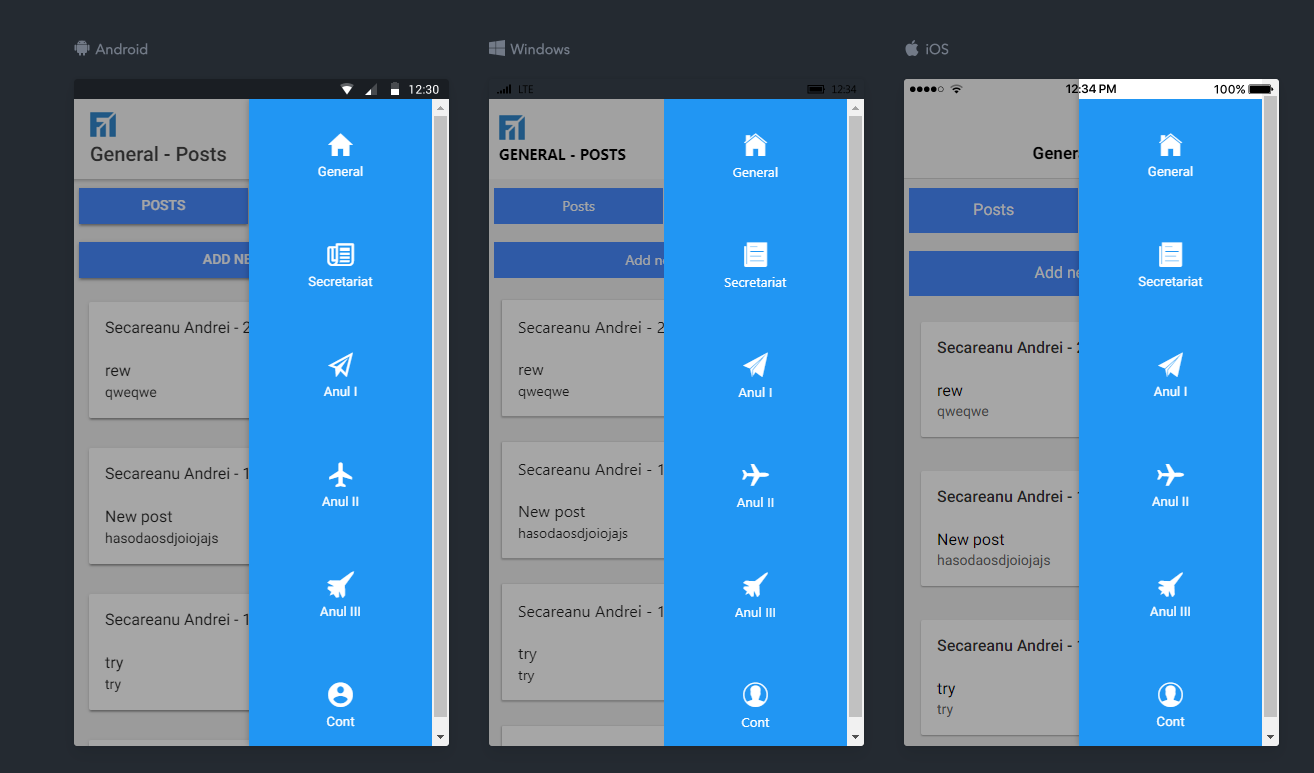


Fig 2.9 – Vizualizare multiplă pentru platforme

Aplicația a fost simulată pentru toate cele trei sisteme de operare, principalul fiind Android.

După cum s epoate observa în Fig 2.9, pentru dezvoltarea aplicației au fost utilizate numeroase iconițe reprezentative fiecărui titlu de meniu, puse la dispoziție de către Ionic. Iconițele pot fi obervate și în restul aplicației pentru conferirea unui aspect dinamic si eficient.

Tema vine cu un scss specific, iar acesta conține și un grid specific telefoanelor mobile. Tema este scrisă în scss, iar frameworkul ionic oferă posibilitatea dezvoltatorului de a suprascrie variabilele. Aplicația a folosit tema pusă la dispoziție de ionic, precum o parte din stil a fost creat specific pentru aplicație.

De exemplu, în aplicație pentru a stiliza cutiile unde sunt afișate posturile scrise de către utilizatori, s-a folosit elementul card din framework. Aici se regăsesc și stilizări proprii pentru a crea elementele într-un stil necesar.

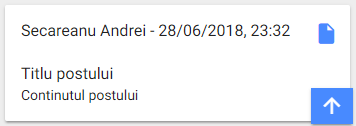


Fig 2.10 – Componentă ionic card

Pentru crearea cardului s-a folosit componenta ion-card din frameworkul Ionic.

Codurile ionic specifice aplicației conțin informații despre numele și prenumele utilizatorului, data și ora la care a fost postat un anumit conținut. Și informații succinte ale acestuia, specifice primelor două rânduri, preluate automat, cum ar fi titlul și conținutul post-uilui restrâns într-un număr specific de caractere.

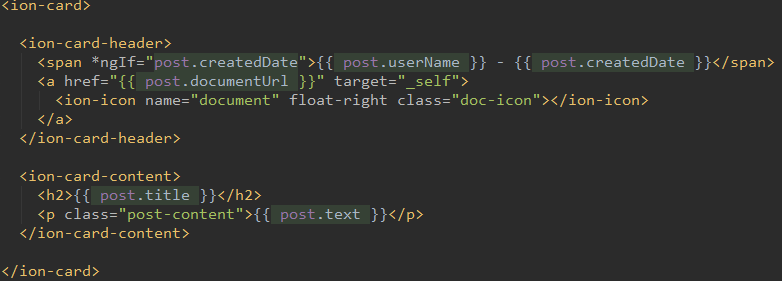


Fig 2.11 – Componentă ionic card cod

Ionic oferă mult mai multe elemente cu stil nativ, cum ar fi check boxurile, listele, butoanele radio, formulare și multe altele.

* 1. **Modelele de date și formulare**

În aplicație am creat modelele de date în namespaceuri. În fiecare namespace este compus din elementele specifice acelei colecții de date. De exemplu pentru datele de tip post care vin de la baza de date, am creat un namespace unde sunt înglobate elementele ce țin de post. Astfel avem modelul pentru post care este o interfață cu elemente de tip cheie valoare. Unde cheia este numele specific pentru a ține informația, iar valoarea este tipul de informație deținută în cheie.

De asemenea tot aici se regăsește formularul postului, care este folosit atunci când un utilizator dorește să creeze un post. Formularul este bazat pe formularele reactive din Angular. Formularul este creat din FormGroup, iar în interiorul lui se află chei și valori. Cheile sunt reprezentate de cheile intefeței și valorile sunt FormControls, unde se setează valoarea lor și Validatorii.

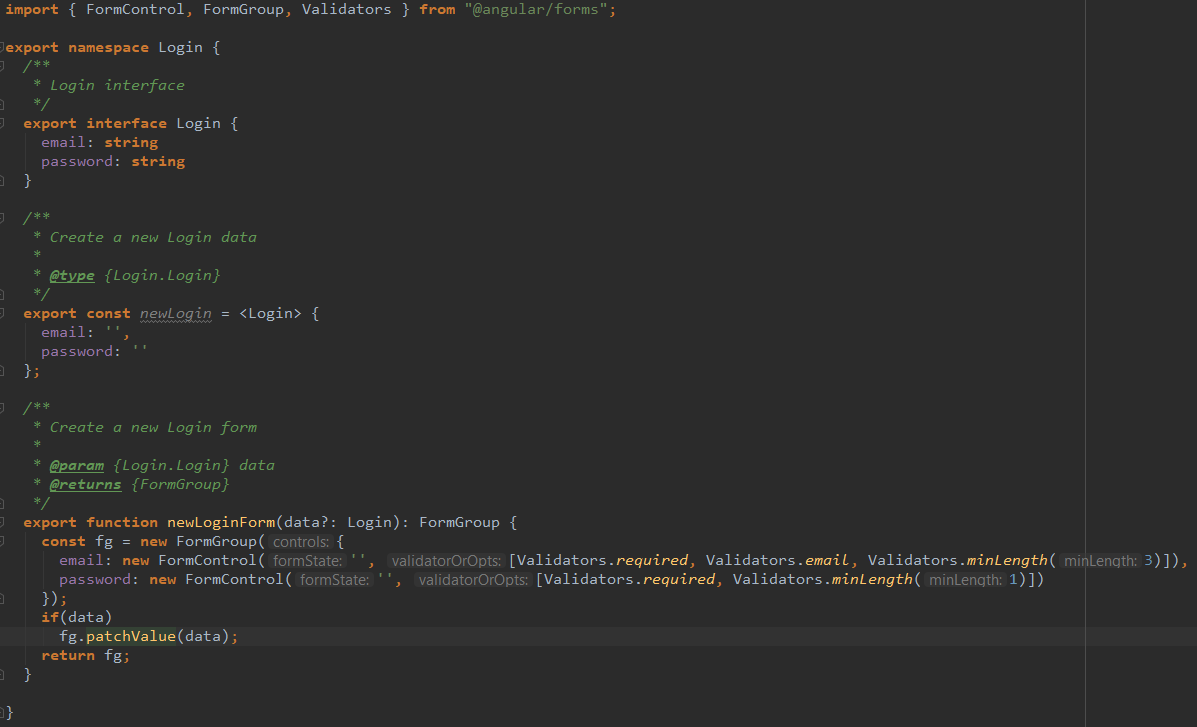


Fig 2.12 – Modelul și formularul pentru logare

* 1. **Pagini**

Paginile proiectului sunt reprezentate de paretea vizuală și funcționalitate. O pagină este creată din trei fișiere: HTML – unde se afla codul pentru viualizare, SCSS – unde se află stilizarea particulară a paginii și TS – unde este logica și funcționalitatea paginii.

Aplicația are opt pagini și sunt acestea:

* Anul-unu: în care sunt informațiile despre primul an de facultate. Aici se regăsesc postările și chatul pentru primul an;
* anul-doi: în care sunt informațiile despre al doilea an de facultate. Aici se regăsesc postările și chatul pentru al doilea an;
* anul-trei: în care sunt informațiile despre al treilea an de facultate. Aici se regăsesc postările și chatul pentru al treilea an;
* general: în care sunt informațiile generale. Aici se regăsesc postările și chatul pentru informațiile generale;
* secreatariat: în care sunt informațiile care țin de secretariatul facultății. Aici se regăsesc postările care țin de secretariatul facultății;
* cont: în care sunt informațiile care țin de utilizatorul logat care folosește aplicația;
* login: este pagina de logare;
* post-add: este pagina ce conține formularul pentru crearea unui post;

Pagina de login este prezentată în detaliu în secțiunea de autentificare.

În pagina de contact se afișează informațiile despre utlizator. Acestea sunt preluate de la baza de date Cloud Firestore și sunt afișate pe pagina Cont. Aici se vor găsi, numele și prenumele utilizatorului, precum și emailul și rolul.

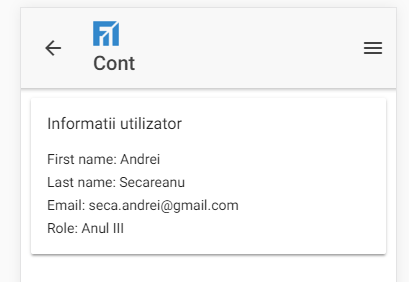


Fig 2.13 – Informații utilizator

Paginile ”Anul unu”, ”Anul doi” , ”Anul trei”, ”General” și ”Secretariat” sunt foarte asemănătoare între ele, referitor la conținutul acestora. Fiecare secțiune are la rândul ei o pagină de postări, care va afișa propriile postări. Pentru viitor, aplicația își propune să permită ca utilizatorii să acceseze doar postările specifice anului din care fac parte, respectiv paginile cu același nume. Utilizatorii Anului 1, vor avea acces doar la „Anul unu”, ”Secretariat” și ”General”. Doar trecerea dintr-un an în altul, va permite utilizatorilor să acceseze și celelalte platforme. În anul doi, doar studenții acestui an vor avea acces la această parte a aplicației.

Toate aceste pagini au în comun următoarele elemente:

* O parte pentru postări, unde se vor putea afișa postări specifice fiecărui an, în funcție de tipul de pagină accesată de utilizator.
* De asemenea un buton pentru creare de noi posturi, care va duce la pagina de creere a posturilor.
* Chat-ul sau Vizualizarea de chat – care permite studenților să vadă discuțiile și ultimele comentarii afișate, specifice grupului din care fac parte.
* Vizualizarea de postări - care v-a permite studenților să vadă informațiile postate.

Fiecare post va afișa numele autorului care a distribuit sau compus anumite documente. Informațiile vor fi adăugate în contul fiecărui utilizator de către administratorul contului. La începutul fiecărui an, administratorul paginii va crea conturi pentru fiecare dintre studenți. Aceștia vor primi ca ”User name” propria adresă de e-mail din baza de date a facultății, iar parola va fi un cod unic generat și ales de către administrator. Toate informațiile personale, privitoare la autorul unei pagini, conform Fig. 2.13 vor fi completate în contul unei pagini doar de administratorul acesteia, tot el având acces la schimbarea sau modificarea de ”User name” și ”Parolă”. Fiecare postare afișează automat numele și prenumele utilizatorului care a compus conținutul.

Fiecare student va putea sa vadă informațiile însușite contului său, elaborate in figura de mai sus, accesând din bara de meniu afișată în header-ul fiecărei pagini, pagina ”Cont”. Pagina Cont conține următoarele informații:

* „First name” - care va afișa informații despre numele utilziatorului,
* „Last name” - Prenumele sau numele de familie din baza de date a facultății,
* “E-mail” - afișează adresa de e-mail a studentului, care reprezintă de asemenea și “User name-ul” abonatului.
* ”Role” – căsuță care face referire la anul de studiu generată în baza de date a facultății și specifică fiecărui utilizator, dar această secțiune reprezintă și pagina specifică la care studentul are acces.

Pe această pagină se află butoanele care vor duce la chatul tipului de pagină. De asemenea este un buton pentru creare de posturi, care va duce la pagina de creere a posturilor. chat-ul, admin post, vizualizarea de chat, vizualizarea de postări.

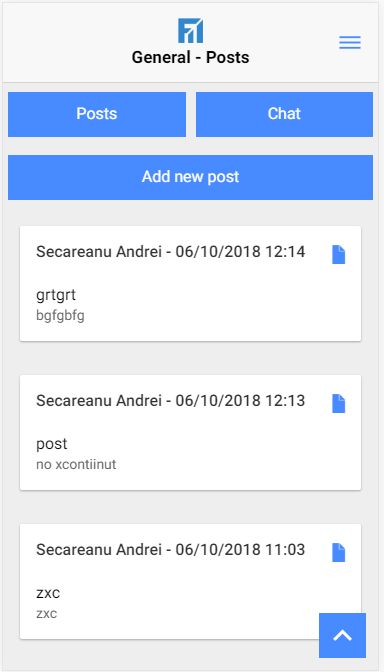


Fig 2.14 – Pagina cu posturile

Farmecul acestei aplicații îl reprezintă posturile care sunt preluate direct din baza de date Cloud Firestore. Baza de date Cloud Firestore este o bază care oferă clienților conectați datele în timp real. Atunci când un utilizator va crea o nouă postare, apăsând butonul ”Add new post”, aceasta ce va fi scrisă în baza de date. Firestore va trimite tuturor utilizatorilor conectați noile date introduse în timp real iar postarea va fi afișată pe peretele aplicaței, reprezentat în Figura 2.14. Fiecare utilizator va fi capabil să vadă postările celorlalți utilizatori fără vreo restricție. Pentru a evita supra-solicitarea aplicației, pe pretele care conține toate postările vor fi afișate mereu ultimele 10 adăugiri, celelalte documente fiind stocate în baza de date, pentru a rămâne în isoric, spre accesul tuturor utilizatorilor, în caz că cele mai recente postări vor fi mai mult de 10. Utilizatorul este liber când, accesează peretele cu postările să le vizulaizeze pe toate afișate, și să le vadă prin ”scroll down”. Fiecare postare poate fi accesată printr-un simplu click. De asemenea postarea va permite utilizatorului să descarce documente adiționale.

Butonul albastru cu o săgeată albă în interior, va fi fixat mereu pe ecranul cu post-urile, permițând utilizatorului să se întoarcă în top-ul paginii, unde v-a putea revizualiza ultimele postări adăugate. Butonul va fi afișat doar pe pagina cu post-urile, nu și în interiorul acestora.

Pe peretele principal, la nivel de client se folosesc obsarvabile, care așteaptă o notificare din partea bazei de date pentru a actualiza templaet-ul utilizatorului. Astfel se crează un canal de comunicare între baza de date și fiecare utilizator, iar datele adăugate sunt comunicate instant, fără ca utilizatorul să facă o nouă cerere la baza de date.

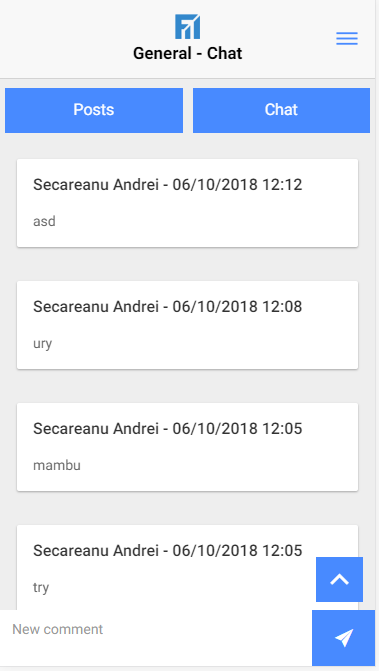


Fig 2.15 – Pagina general chat

Pe pagina de chat vor fi afișate comentariile, iar acestea conțin numele celui care a creat comentariul, data, ora la care a fost creat și conținutul postului, cu mesajul pe care utilizatorul a ales să îl redactezte distribuindu-l spre vizulaizare, celor ce au descărcat și instalat aplicația mobilă. De asemenea, pe aceeși pagină se mai găsește și câmpul care v-a permite utilizatorului să creeze noi comentarii pentru pagina respectivă. Același mecanism de comunicare cu baza de date ca la crearea de post este folosit și pentru crearea de comentarii. Iar pentru a crea un comentariu, utilizatorul trebuie să introducă conținutul acestuia în inputul de la baza ecarnului în secțiunea ”New Comment”, conform figurii 2.15 afișată mai sus; și apoi să trimită la server noul comentariu prin apăsarea butonului ”SEND” prezentat în Fig. 2.15 sub forma unei iconițe care reprezintă un avion de hârtie.

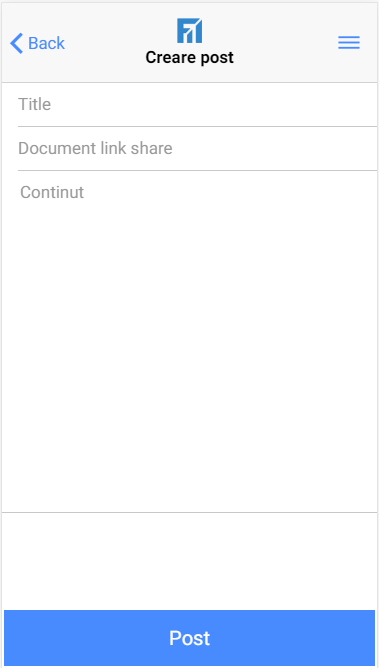


Fig 2.16 – Pagină adăugare postare

Pentru creerea de noi postări în aplicație există o pagină dedicată acestui lucru, unde utilizatorul poate completa formularul care va conține datele postului. Conform Fig 2.16 utilizatorul va introduce titlul postului după propria preferință in secțiunea ”Title” aflată in partea de sus a ecranului. Sub secțiunea ”Titlul”, utilizatorul va avea posibilitatea să posteze Conținutul paginii, în secțiunea sub numele de ”Conținut”. Adițional Conținutului, utilizatorul va avea posibilitatea să posteze și un link, în secțiunea ”Document link share”. Link care v-a permite utilizatorului să posteze un document adițional textului scris în conținut. La alegerea utilizatorului pot fi popstate diferite link-uri care vor conține informații suplimentare, chestionare, formulare, documente.

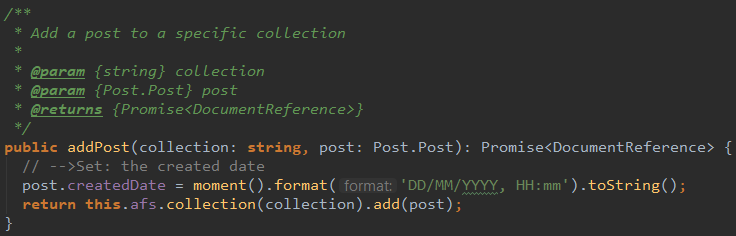


Fig. 2.17 – Adăugare post în Firestore

**2.7 Componentele**

Componenetele reprezintă cod reutilizatbil în aplicație. Pentru a reutiliza cod în aplicație s-au folosit următoarele componente:

* chat: unde este scris codul pentru a vizualiza un comentariu scris de un utilizator. Aici sunt vizualizate informațiile despre un comentariu, acestea fiind data creerii comentariului, numele utilizatorului care a scris comentariul, titlul comentariului și textul comentariului;
* post: unde este scris codul pentru a vizualiza o postare scrisă de un utilizator. Aici sunt vizualizate informațiile despre o postare, acestea fiind data creerii postului, numele utilizatorului care a creat postul, titlul postului, textul postului și documentul distribuit de utilizator;
* header: unde este scris codul pentru a vizualiza butoanele care intershimbă paginile dintre post și chat.
* general-chat: unde este scris codul pentru a vizualiza toate comentariile scrise de utilizatori, iar aceste comentarii sunt structurate intr-o listă;
* general-post: unde este scris codul pentru a vizualiza toate comentariile scrise de utilizatori, iar aceste comentarii sunt structurate intr-o listă;
* Aceste componente sunt structurate intr-un modul care le exportă pentru a putea fi reutilizate în pagini sau în subcomponente.

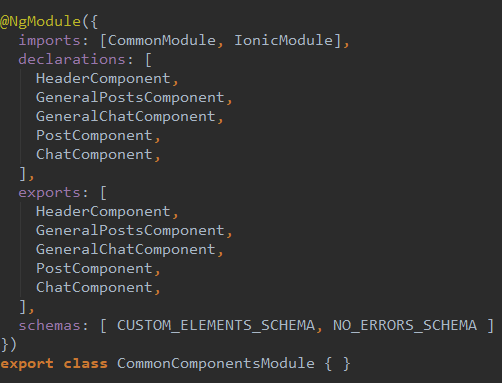


Fig. 2.18 – Structura modulelor commune

În figura 2.18 există declarațiile modulelor și exporturile lor, acestea sunt configurările modulelor pentru a fi folosite în mod comun oriunde în aplicație.

Comunicarea dintre componente și pagini sau între componente se face prin legături de Input și Output.

Variabilele de tip Input sunt folosite pentru a utiliza o variabilă din componenta părintelui în componenta reutilizatbilă. În componenta părinte se folosește o componentă în HTML care face legătura dintre componenta părinte și componenta copil.

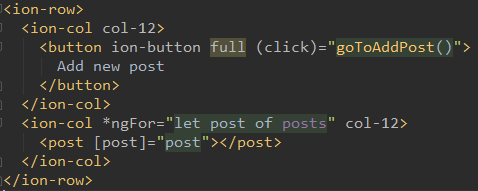


Fig 2.19 – Folosirea componentelor

În figura 2.19 se observă tagul post, care interpretat de angular acesta va face legătura cu componenta post. În interior este legătura dintre variabila părinte post și Inputul variabilă cu același nume din copil. Parantezele pătrate reprezintă legătura cu variabila input sub numele de post din componenta copilului, iar atribuirea valorii este dată de variabila post din ngFor.

Variabilele de tip Output sunt în componenta copil, iar acestea, la un anumit moment, vor genera un emițător de evenimente către componenta părinte. În emițător se stochează orice informație și este trimisă către componenta părinte.

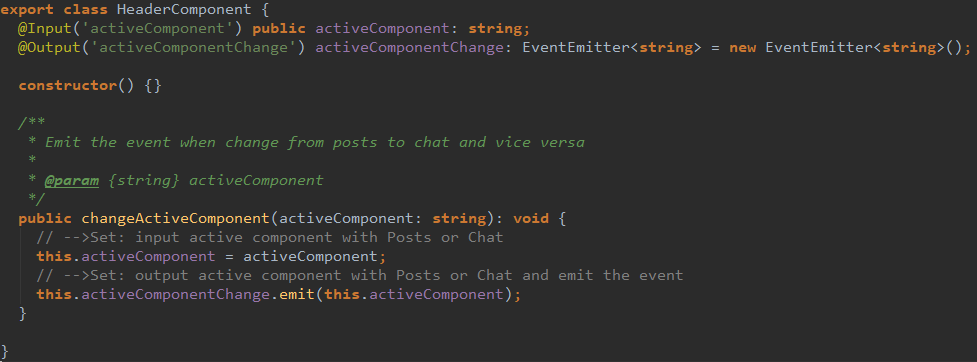


Fig 2.20 – Componenta cu input și output

În figura 2.20 se observă variabila output, iar aceasta este un emițător pentru componenta părinte. Atunci când metoda changeActiveComponent este chemată, variabila activeComponent își va modifica valoarea, iar apoi variabila output va emite un eveniment către părinte cu noua valoare. Părintele va prinde evenimentul emis și va consuma evenimentul având acces la activeComponent.

**2.8 Concluzii**

Aplicațiile mobile hibrid joacă un rol foarte important în creerea de aplicații mobile la nivel global. Aplicația este construită pentru a oferi utilizatorului o funcționalitate cât mai apropiată de aplicațiile native. Cea mai mare provocare este ca plicația hibrid să fie funcțională pentru toate sistemele native la nivel global, deoarece ionic suportă doar versiunile sistemelor de operare mobile cele mai utilizate.

Aplicația mobile hibrid construită oferă utilizatorilor o funcționalitate fluidă, astfle că cei care o utilizează nu pot face diferența dacă aplicația este nativă sau hibridă. Acestă aplicație este optimizată pentru, iar utiliztorul se bucură de aplicație folosind-o ca pe una nativă.

**Concluzii finale**

**Concluzii**

Lucrarea de licență este o aplicație hibridă care oferă atât profesorilor cât și studenților posibilitatea de a interacționa mai rapid, indiferent de platforma folosită. Fiecare utilizator poate să-și aleagă modelul de telefon mobil care îl dorește, indiferent de sistemul de operare pe care acesta rulează, și să folosească aplicația mobilă hibrid, fără a întâmpina dificultăți.

**Contribuții personale**

Am reușit prin îmbinarea ultimelor tehnologii, unele chiar în variantă beta, să implementez o aplicație hibrid care să poată fi folosită de orice utilizator la standardele impuse de aceștia. Aplicația rulează în timp real, iar fiecare utilizator poate să comunice activ cu profesorii sau cu ceilalți studenți.

Experiența utilizatorului este completată de poziționarea intuitivă a elementelor native și de fluiditatea care este folosită în dinamica aplicașiei prin interschimbarea paginilor și a componentelor. Prin crearea de componente și servicii specializate prin carea acestea pot fi reutilizate în pagini, codul este stabil, iar utilizatorul se poate bucura de toate funcționalitățile aplicației.

**Adăugări de viitor**

Pentru versiunea următoare de aplicație există mai multe elemente ce pot fi integrate:

* Adăugarea ca fiecare student și profesor să comunice individual, în funcție de proiect și de teme.
* Intruducerea opțiunii utilizatorilor de a trimite fișiere în secțiunea de chat, astfel că fiecare va putea comunica mai ușor.
* Adăugarea opțiunii ca postul să fie șters în următoarele 10 minute, în caz ca acesta a fost trimis din greșeală.
* Crearea de grupuri pentru diferite proiecte care trebuie dezvoltate în comun de studenți sub îndrumarea profesorilor.
* Crearea unei secțiuni specializate pentru discuția dintre profesorii coordonatori pentru examenul de licență și studentul care va susține lucrarea.
* Adăugarea de emoticonuri în chat.
* Implementarea apleurilor video și audio pentru comunicarea mai fluientă între studenți și rpofesori.

Anexe TEHNOLOGIILE DE TIP EDGE

**1.1 Tehnologii web – HTML, CSS, JavaScript[1]**

Tehnologiile web sunt folosite tot mai mult folosite de dezvoltatorii, pentru că cereare de platforme web este în creștere.

****

Fig. 1.1 – HTML, CSS, JS[15]

**HTML** este limbajul care este folosit pentru orice website. Acesta este un limbaj bazat pe taguri. Tagurile conțin informație, iar în funcție de tipul tagului, informația va fi afișată utilizatorului sau va fi folosită pentru o mai bună funcționare a siteului.

**CSS:** Numele CSS stă pentru Cascading Style Sheet. Css este un stylesheet limbaj folosit să descrie prezentarea unui document scris în HTML sau XML (incluzând dialectele XML cum ar fi SVG sau XHTML). CSS descrie cum elementele ar trebui să fie redate pe ecran, pe hârtie, în vorbire sau în alte elemente media. Este limbajul interpretat de browser pentru a oferi siteului o stilizare specifică. Acesta este bazat pe clase și id-uri, care sunt atașate unui tag în HTML. Oricărei clase și id i se poate atașa diverse proprietăți, cum ar fi: culoare, mărime, distanță. Fiecare propietate este bazată pe o cheie și valoare.

**JavaScript** este un limbaj care manipulează tagurile HTML și astfel crează pagii dinamice. Orice element al tagului manipulat poate fi modificat de către dezvoltator. Astfel orice utilizator poate să se bucure de pagini interactive, iar siteul devine mult mai atrăgător.

**1.2 Prepocesoare – SCSS, TypeScript**

Prepocesoarele sunt limbajele de programare care ajută dezvoltatorii să scrie cod mai stabil și mai scalabil. Dezvoltatorii vor scrie cod mai puțin și mai puternic, iar codul compilat este mult mai optimizat datorită standardelor impuse de preprocesoare.

****

Fig. 1.2 – SASS->SCSS, TS[13, 16]

**SCSS**, cunoscut sub numele de SASS,este preprocesor pentru CSS și este bazat pe limbajul Ruby. Fișierele SCSS sunt asemănătoare cu cele de CSS, dar beneficează de extensii pentru ca dezvoltatorii să scrie codul sursă mai eficient, mai stabil și mai structurat. Fișierele SCSS nu pot fi interpretate de browser, iar prin limbajul Ruby, acestea vor fi transformate în fișiere CSS. Prepocesorul nu extinde funcționalitățile și caracteristicile CSS-ului, ci oferă flexibilitate și simplitate în scrierea codului.

Principalele avantaje ale prepocesorului sunt:

* variabilele: acestea sunt folosite în SCSS pentru a stoca informații care vor fi refolosite. Avantajul este că dezvoltatorul, în cazul unei schimbări, trebuie să modifice doar valoarea variabilei;
* gruparea: este un mare avantaj pentru stilizarea elementelor în cascadă. Utilizând gruparea dezvoltatorul poate să urmărească și să modifice codul cu o acuratețe mai mare;
* parțialele: sunt fișiere care cuprind o parte din stilizarea websiteului. Aceste fișiere pot fi reutilizate fără a mai duplica cod. De exemplu dacă meniul are un anumit stil, parțialul va fi inclus în fiecare din paginile websitului.
* operatorii: sunt cei cunoscuți în matematica de bază.Operatorii de adunare, de scădere, de înmulțire, de împărțire și procentele sunt folosiți pentru a fluidiza calculele pentru dimensiuni în stilizarea paginii;

**Typescrip** este un superset al JavaScriptului. Fișierele sursă scrise în typescript trebuie să fie compilate în fișiere javascript pentru ca browserul să le poată rula. Typescript suportă ES6, ES2016, ES2017. Câteva dintre funcționalitățile suportate sunt clases, modulele și funcțiile cu săgeți.

Cel mai mare avantaj al typescriptului este că suportă tipuri pentru variabile. Aceste tipuri ajută la scrierea unui cod stabil, ușor de citit și testat. De asemenea oricărui dezvoltator îi vine mult mai ușor să refactorizeze un cod cu tipuri. Unul dintre lucrurile benefice ale tipurilor este că oferă explicație asupra codului, fiind considerată ca documentație, iar alți dezvoltatori pot să înțeleagă mai eficient un cod scris de alt dezvoltator.

Tipurile se folosesc la variabile și la ceea ce funcțiile returnează. Typescript suportă unirea de tipurilor, astfel că o variabilă poate să fie de două sau mai multe tipuri. Principalele tipuri sunt: string, number, boolean, array, any, enum, tuple și void

**1.3 Cordova**

Cordova este cel mai folosit framework care oferă utilizatorilor o legătură între tehnologiile web și limbajele de programare native în funcție de platforma mobilă folosită.

****

Fig. 1.3 – Cordova[11]

Cordova, inițial cunoscut sub numele de PhoneGap, este framework pentru crearea de aplicații mobile. Cordova oferă dezvoltatorilor opțiunea de a scrie codul sursă în limbaje web, HTML, CSS și JavaScript. Cordova folosește un webview nativ pentru a acesa elementele native în funcție de platforma mobilă prin intermediul tehnologiilor web. Acesta extinde funcționalitatea tehnologiilor web pentru ca ele ruleze pe o platformă mobilă, Android, iOS și Windows Phone. [11]

Aplicațiile create cu ajutorul frameworkului cordova sunt cunoscute ca aplicații hibrid. Acestea nu sunt scrise numai cu ajutorul tehnologiilor web, și nu pot fi numite aplicații web, dar nici numai cu ajutorul tehnologiilor native. Partea vizuală a plicației este scrisă în HTML și CSS, iar partea logică este scrisă în JavaScript. Deoarece browserele nu pot accesa elementele native ale unei platforme, se folosește webviewul care este o cale de comunicare între tehnologiile web și cele native prin intermediul unor pluginuri.

Un plugin poate fi scris pentru una sau mai multe platforme și poate fi accesa de către tehnologiile web pentru a activa și dezactiva elementele native, astfle o aplicație hibrid poate să folosească elemente native precum GPS, informații despre device, camera pentru poze și filme, microfonul, sistemul de sunet, memoria telefonului și orice este pus la disăpziție. Cordova oferă dezvoltatorilor oportunitatea de a crea singuri diverse pluginuri ce pot fi folosite în aplicațiile hibrid.

**1.4 Angular**

Angular este unul dintre cele mai puternice frameworkuri pentru dezvoltare web și oferă o stabilitate și optimizare forate bună.

****

Fig. 1.4 – Angular[8]

Angular este un framework pentru a crea aplicații și platforme web. Acesta folosește la bază TypeScript, iar codul sursă este scris în TypeScript, iar apoi este compilat în JavaScript.

Angular este bazat pe module, acestea fiind blocurile care oferă context componentelor. O componentă este compuns dintr-un set de elemente logice care modifică datele și templaturile după cum dorește dezvoltatorul. Componentele folosesc și servicii care oferă logică pentru a fi reutilizată. Fiecare serviciu trebuie injectat în fiecare componentă pentru a putea fi utilizat în contextul respectiv. Componentele și serviciile sunt clase care folosesc metadate pentru a fi interpretate de Angular. În aceste metadate sunt asociate si templaturile. Templaturile sunt partea vizuală scrisă în html cu ajutorul directivelor și elemntelor de legătură.

**Modulele** sunt elementele de legătură pentru o aplicație scrisă în Angular. Orice aplicație începe cu un modul de bază unde sunt adunate toate celelalte elemente: servicii, componenete, alte module. Parctic modulul de bază cuprinde toate legăturile pentru funcționarea aplicației.

Modulele ajută la organizarea aplicațiilor, iar acestea pot fi reutilizate . Acest tip de structurare oferă beneficiul că fiecare modul poate fi încărcat la cerere, fără ca aplicația să încarce module nenecesare.

**Componentele** sunt clase specifice care au ca decorator @Component, iar Angular știe cum să-l manipuleze pe baza metadatelor. Orice componentă conține o parte logică a aplicației și o parte de manipulare a datelor și este asociată cu un template HTML.

**Templaturile** sunt fișiere HTML care conțin directive și elemente de legătură. Directivele sunt elementele logice ale templatului, iar elementele de legătură conectează datele aplicație cu elementele din DOM. Elementele de legătură se bazează pe proprietăți ale componentelor sau pe emiterea de evenimente. Angular suportă elemente de legăture duble, ceea ce înseamnă că orice modificare adusă la nivelul DOMului va modifica și valoarea datelor din aplicație. Templaturile de asemenea suportă și canalele de transformare a datelor, numite generic pipes. Acestea transformă datele afișate utilizatorilor în format agreabil.

**Serviciile** sunt clase care pot fi injectate în componente pentru a fi reutilizate. Acestea oferă o parte logică a aplicației și sunt decorate cu metadata @Injectable. Odată injectate într-o componentă, serviciile devin parte dependentă a acelui component.

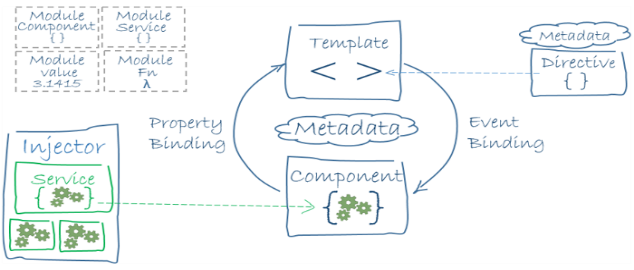


Fig. 1.5 – Structura Angular[8]

**1.5 Ionic**

Ionic este frameworkul care, prin tehnologiile web, crează aplicații mobile hibrid pentru cele mai cunsocute platforme mobile.

****

Fig. 1.6 – Ionic[12]

Ionic este o platformă folosită pentru a forma și construi aplicații mobile hibrid, pe multiple platforme și au anumite tool-uri de dezvoltare specifice. Ionic pe de altă parte este o platformă independentă care folosește HTML , CSS și JavaScript. Porcesul de dezvoltare este detul de rapid și developării care îl folosesc au acces direct la API cu Cordova. ( Aarush Gupta, Abdul Gaffar H. 2016 )

Ionic este un framework care are la baza un kit de dezvoltare software. Acesta oferă dezoltatorilor să creeze aplicații mobile hibrid utilizând tehnologiile web. Ionic este bazat pe Cordova și pe Angular. Ionic este un framework care se bazează pe imaginea aplicației mobile, iar logica și funcționalitatea este bazată pe Angular și Cordova.[12]

Ionic folosește Cordova pentru a acesa elementele native ale platformei. Dezvoltatorii pot crea aplicații hibride pentru Android, iOS și Windows. Ionic are inclus teme specifice pentru fiecare platformă, iar elementele vizuale sunt specifice cu cele native, astfel utilizatoii finali nu vor putea să diferențieze o aplicație nativă de una hibrid.

Ionic și-a dezvoltat propriile plugine bazate pe cordova, iar dezvoltatorii le pot instala cu ajutorul pachetului npm. Acesta este cunoscut sub numele de Ionic Native, și poate accesa elementele native în funcție de platformă.

Ionic și-a dezvoltat și propriul API, acesta este bazat pe componente Angular. Aestea pot fi ușor modificate sau adaptate de dezvoltator.

Navigarea înntre pagini ete bazată pe o stivă. Fiecare pagină poate face push pentru a intre în stivă, iar pentru a goli stiva se folosește pop.

**1.6 RxJS**

RxJS este o componentă care oferă dezvoltatorilor puterea de a crea cod în timp real și este destul de puternic să suporte diferite date care să fie actualizate în timp real.

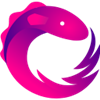
****

Fig. 1.7 – RxJS[17]

ReactivX combină Modelul Observator cu modelul Iterator si programarea funcționale cu colectile, pentru a satisfice nevoia unui mod ideal de gestionare a secvențelor de evenimente.

Conceptele essențiale din RxJS care rezolvă managmenentul evenimentului async sunt:[18]

* Observabil: reprezintă ideea unei colecții invocate de valori sau evenimente viitoare.
* Observator: este o colecție de callbacks care știu cum să asculte valorile trimise de Observabil
* Abonarea (Subscriprion): reprezintă execuția unei Observabile, este utilă în primul rând pentru anularea execuției
* Operatorii: sunt funcții clare ce permit unui program funcțional să se ocupe cu colecții și operații ca:  [map](https://rxjs-dev.firebaseapp.com/api/operators/map), [filter](https://rxjs-dev.firebaseapp.com/api/operators/filter), [concat](https://rxjs-dev.firebaseapp.com/api/index/concat), [flatMap](https://rxjs-dev.firebaseapp.com/api/operators/flatMap), etc.
* Subiectul: este echivalentul unui EventEmitter, și singurul mod de a multiplica o valoare sau un eveniment la mai multi observatory.
* Planificatori: sunt dispecerei centralizați pentru a controla concurența, permițând coordonarea atunci când apare calculul în ex: setTimeout sau request Animation Frames și multe altele.

Abrevierea Rx vine de la ReactiveX, care este o librărie și este utilizată în crearea de programe folosid evenimente asincrone. Rx folosește șiruri de date, cunoscute sub nume de obsarvable. Abrevierea JS vine de la JavaScript, iar RxJS este programarea reactivă pentru JavaScript.

Comunicarea de date se poate face sincron sau asincron, prin cerere sau trimitere. Astfel RxJS folosește comunicare asincron, prin trimitere reactivă.

|  |  |  |
| --- | --- | --- |
|  | Returnează o singură valoare | Returnează mai multe valori |
| Cerere/Sincron | Obiectele | Array |
| Trimitere/Asincron | Promise | Observable |

*Fig. 1.8 – Compararea elementelor Sincron și asincron*

Comunicarea la cerere este făcută de receptorul de date la un anumit timp. Această cerere se face sincron, adică se așteaptă alte execuții să fie finalizate și mai apoi se face cerearea și se așteaptă rezultatul pentru a continua executarea liniilor de cod următoare. La comunicarea prin cerere se emite o singură valoare, iar după consumarea acesteia nu mai există comunicare între emițător și receptor.

De asemenea se poate face cerere de un șir de date, iar după consumarea lui nu mai există conexiune între receptor și emițător. De aceea dacă receptorul dorește să primească noi date este nevoie de o nouă cerere.

Comunicarea prin trimitere de date, pentru programarea reactivă, este făcută de emițător către un receptor care a subscris către acel șir de date. Codul poate rula simultan cu alte linii de cod, fără a bloca continuarea execuției programului. Pentru a consuma o singură dată datele se folosește Promise. O promisiunea preia datele de la emițător, iar receptorul consumă datele prin funcția then, iar dacă există o eroare se face prin funcția catch.

Pentru a prelua mai multe date se folosește Obsarvable. Pentru ca receptorul să primească șirul de date în timp real trebuie să subscrie la observable. Șirul de caractere este numit generi Observer. Acesta este compus din trei funcții, next, error și complete. Funcția next este obligatorie, pe când celelalte două sunt opționale. În funcția next se preia șirul de date, iar dezvoltatorul poate să le folosească. În funcția error este prinsă eroarea, dacă Observerul emite o eroare, obsarvableul nu mai emite date, iar secvența s-a terminat. Odată ce șirul de date s-a finalizat cu succes, fără să apară vreo eroare, este chemată funcția complete.

De asemenea, receptorul poate folosi diferiți operatori pentru manipularea șirului de date. Câțiva dintre operatorii mai cunscuți sunt:

* throttle: care oferă puncte de rupere a timpului în transferul de date, astfel că datele sunt grupate în funcție de intervalul de timp și receptarea lor;
* map: care oferă opțiunea de a modifica datele după bunul plac al dezvoltatorului;
* filter: care oferă opțiunea de a flitra șirul de date;

Toți acești operatori returnează un obsarvable, iar la sfârșitul utilizării lor observableul se consumă cu subs cribe.

Pentru că un receptor a subscris la un observable, orice stare care a modificat datele, va trimite la receptor un nou șir de date cu noile valori.

Dacă receptorul dorește să închidă comunicarea, va trebui să nu mai fie subscris la acel obsarvable. Acest lucru se face prin metoda unsubscribe. După ce această metodă a fost chemată, șirul de date nu va mai fi reactualizat.

**1.7 Firestore[19, 20]**

Firestore este baza de date oferită de Google care se bazează pe principiile a altor două baze de date, Firebase și Spanner, create tot de Google. Firestore oferă utilizatorilor opțiunea de a utiliza baza de date în aplicații web sau platform mobile pentru Android și iOS.

****

Fig. 1.9 – Firestore[19]

Cloud Firestore este o bază de date în versiunea beta care a apărut în octombrie 2017. Deși gigantul Google mai avea o bază de date similară, cunoscută sub numele de Firebase, compania a dorit să unifice principiul de TrueTime al produsului Spanner cu funcționalitatea de a oferi date în timp real, ca baza de date Firebase.

Cloud Firestore este o bază de date care stochează datele într-un format specific create de Google. Firebase stochează datele într-un fișier gigant JSON unde nu există reguli care să creeze o structură consistentă în fișier, însă în Cloud Firestore există o structură organizată a datelor.

Cloud Firestore este o bază de date structurată pe documente, ceea ce înseamnă că toate datele sunt stocate în obiecte numite documente care au la bază perechi de chei și valori. Valorile stocate în documente pot conține diferite informații, de la elemente simple cum ar fi numere, șiruri de caractere, până la date binare și obiecte JSON. La rândul lor documentele sunt structurate în colecții. Aceste colecții vor conține doar documente, iar dezvoltatorul nu va putea pune date decât în documente.

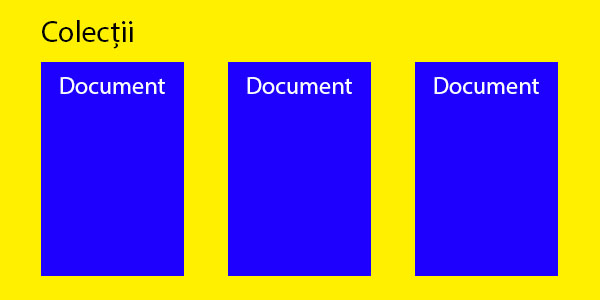


Fig. 1.10 – Organizarea documentelor în colecții

De obicei, în proiectele mari, documentele vor conține ca date și subcolecții. Iar la rândul lor subcolecțiile vor avea documente care vor avea ca date alte subolecții, astfle că structura de baze de date va arăta ca un arbore, conform figurii de mai jos:

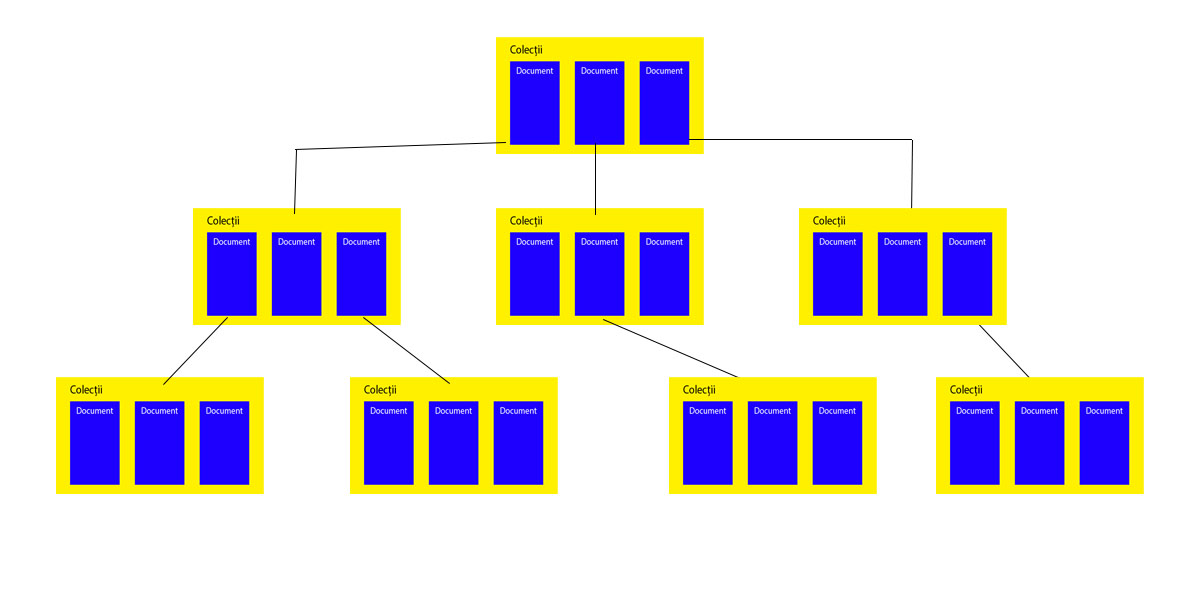


Fig 1.11 – Arbore de colecții, dubcolecții și documente

Acastă structură oferă flexibilitate în creare de interogări în baza de date, astfle că dezvoltatorul va putea colecta date din documentul ales fără să preia toate datele din toate documentele. Astfel dezvoltatorul poate crea o structură logică și specifică pentru fiecare aplicație

Cloud Firestore pune la dispoziție crearea de indecși la nivelul bazei de date, iar aceștia vor organiza documentele în funcție de setări. Setările pot fi de două tipuri, ordonare ascendentă sau descendentă pe multiple chei ale documentelor.

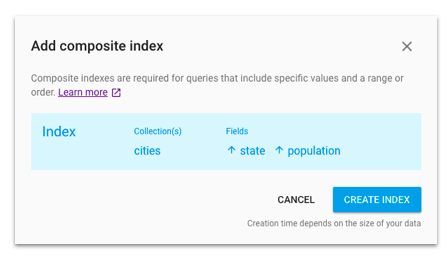


Fig 1.12 – Creare de indecși[19]

După crearea de indecși dezvoltatorul va crea interogări simple, astfel că va putea să caute pe mai multe chei ceea ce dorește.

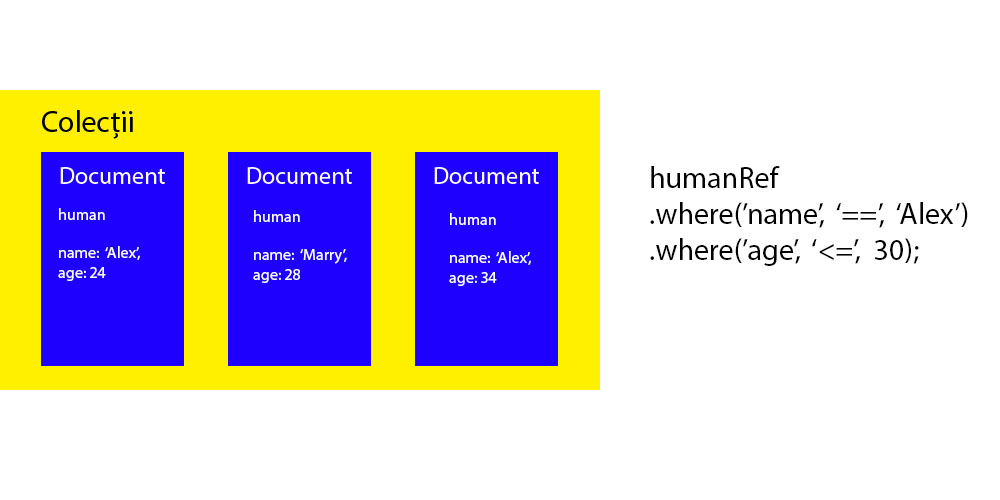


Fig 1.13 – Creare de interogări simple

Baza de date Cloud Firestore este creată pentru a pentru a fi scalabilă la orice număr de date, astfel că interogările se scalează la numărul de rezultate nu la dimensiunea de date stocate. Dacă o bază de date are stocate 100 de orșe, 1 milion de orașe sau 10 de miliane de orașe timpul de răspuns pentru a prelua primele 5 cele mai bogat orașe este același.

Google a construit Spanner care este o bază de date, considerată cea mai inovativă structură de baze de date. Google a dorit să aibă o scalabilitate la nivel global. Baza de date Spanner are distribuite servere la nivel regional si global.

Toate aceste baze de date comunică între ele, iar datele sunt menținute și actualizate în toate serverele centralizate. În baza de date Spanner datele sunt consistente astfel că la citirea lor utilizatorul va primi datele cele mai actualizate în timp. De asemenea Spanner folosește replicarea datelor în bazele distribuite, acest aspect fiind automatizat pentru fiecare server.

Atunci când un utilizator dorește să scrie intr-o bază de date, iar dacă baza de date nu este disponibilă serverul caută cea mai apropiată bază de date ce conține copii ale datelor ce trebuie actualizate. Cel mai apropiat server de baze de date din regiune va fi utilizat pentru a scrie noile valori în baza de date. Dacă nici un server la nivel regional nu este disponibil, se caută un server disponibil la nivel global.

După ce datele au fost actualizate, bazele de date comunică între ele și actualizează toate bazele cu noile valori.

Bazele de date Spanner sunt bazate pe teoria generală a relativității. Aceasta spune că spațiul influențează timpul. Cea mai mare problemă a bazelor de date spanner este scrierea acelorlași date de către doi utilizatori.

Pentru a rezolva această problemă Googleul a folosit două majore soluții. Prima este că a minimizat transferul de date la nivel global cu ajutorul GPSului. Fiecare server de baze de date are un sitem GPS. Acesta comunică cu stateliți și actualizează datele exacte la celelalte baze de date într-un timp cât mai scurt. Cea de-a doua rezolvare este utilizarea ceasurilor atomice.

Ceasurile sunt bazate pe atomi de cesiu. În funcție de frecvența oferită, sub influența radiațiilor, vibrațiile cesiului aliniază ceasul atomic la frecvența stabilită. Fiecare server are un ceas atomic pentru a stabili în ce moment un utilizator a făcut cererea de a scrie noi valori în baza de date. Astfel, dacă doi utilizatori doresc să modifice valoarea acelorlași date, se calculează la nivel global un timp mediu, iar be baza acestuia se determină care dintre utilizatori a scris primul și care a scris al doilea.

Cloud Firestore merge pe același principii lae bazelor de date Firebase și Spanner.

**1.8 Concluzii**

Deoarece utlizatorii consumă tot mai mult conținut web, iar aplicațiile sunt cele mai căutate, am decis să utilizez tehnologii care să îmi ofere toate componentele care să mă ajute să scriu o aplicație mobil hibrid.

Astfel am ales Cordova pentru că este cea mai stabilă legătură dintre tehnologiile web și cele native și de asemenea suportă multiple platforme. Angular este cel mai stabil framework pentru fron end și este folosit și de ionic. Ionic oferă o structură foarte dezvoltată și stabilă. Am ales să dezvolt aplicația pe Ionic și datorită uneltelor puse la dispoziție, extensiilor oferite și modului de organizare.

Cloud Firebase oferă performanța necesară pentru o aplicație în timp real și cu date consistente. Această bază de date este foarte complexă, dar în acelați timp oferă o structură foarte eficientă pentru dezvoltatori pentru a o utiliza.

**Bibliografie**

1.Sabin Buraga, Lenuța Alboaie, (2006) – Servicii web, Polirom

2. Joshua Morony, (2017) – Building mobile apps with Ionic 3, e-book <https://www.joshmorony.com/building-mobile-apps-with-ionic-2/?utm_source=link&utm_campaign=homepage&utm_term=dropdown#buy>

3. Digital in 2018, We are social, Global Reports. <https://wearesocial.com/blog/2018/01/global-digital-report-2018>

4. Native mobile apps The wrong choice for business: <https://www.mrc-productivity.com/research/whitepapers/NativeAppsWrongChoice.pdf>

5. Native, Web or Hybrid mobile-app development United States of America, April, IBM

6. International Journal of Current Trends in Engineering & Research p. 563 – 572, Native Vs Hybrid Apps Ketan Anant More , Ms.Priya Chandran: June 2016

7. International Journal of Innovative Research in Computer Science & Technology, Hybrid Application Development using Ionic Framework & AngularJS, Aarush Gupta, Abdul Gaffar H., March – 2016

8. Angular, <https://angular.io/>

9. Sistemul de operare Android <https://www.android.com/>

10. Sistemul de operare iOS, <https://en.wikipedia.org/wiki/IOS>

11. Apache Cordova, <https://cordova.apache.org/>

12. Ionic Framework <https://ionicframework.com/>

13. SASS, SCSS, <https://sass-lang.com/>

14. Sistemul de operare Android, <https://en.wikipedia.org/wiki/Android_(operating_system)>

15. Global <https://www.planet-source-code.com/vb/default.asp?lngWId=14>

16. Typescript, <https://www.typescriptlang.org/>

17. Reactive programming <http://reactivex.io/>

18. Reactive RxJs <https://rxjs-dev.firebaseapp.com/>

19. Firebase, Firestore <https://firebase.google.com/docs/firestore/>

20. Spanner Google <https://cloud.google.com/spanner/>

21. Angular fire npm package <https://github.com/angular/angularfire2>