UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Madison - O aplicație mobilă pentru studenți

propusă de

Dacian-Ștefan Spînu

Sesiunea: *Iulie, 2017*

Coordonator ştiinţific

Lector Rădulescu Vlad

UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI

FACULTATEA DE INFORMATICĂ

Madison - O aplicație mobilă pentru studenți

Dacian-Ștefan Spînu

Sesiunea: *Iulie, 2017*

Coordonator ştiinţific

Lector Vlad Rădulescu

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Madison - O aplicație mobilă pentru studenți*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară 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 şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, *03.07.2017*

Absolvent *Dacian-Ștefan Spînu*

_____

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „*Madison - O aplicaţie mobilă pentru studenţi*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

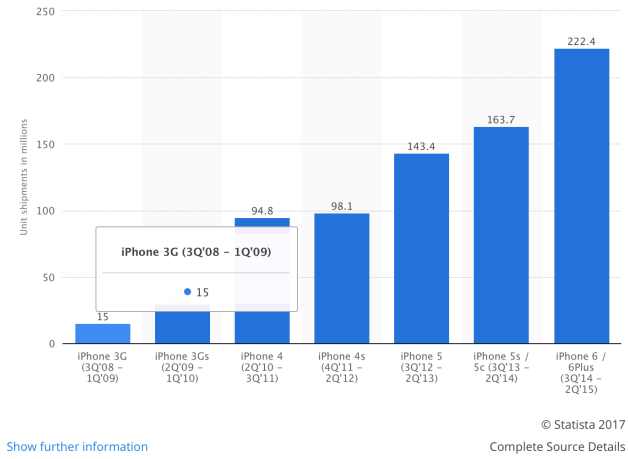
De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, *03.07.2017*

Absolvent *Dacian-Ştefan Spînu*

_____

Introducere



Dispozitivele inteligente fac parte din viața multora dintre noi, mai ales a tinerilor. De la brățări și ceasuri inteligente, până la laptop-uri performante si telefoane cu putere de procesare mult peste a calculatoarelor personale din urmă cu zece ani.

Însă de departe cele care au marcat existența cetățeanului modern sunt telefoanele inteligente (sau în engleză smartphones). Este imposibil ca mergând pe stradă sa nu vedem cel puțin o persoana cu un astfel de dispozitiv în mână.

Istoria acestor dispozitive a început să prindă contur în anii 2000, când Nokia lansa celebrele telefoane cu sisteme de operare Symbian. Microsoft a luat si el cu asalt această piață la mijlocul anilor 2000 cu ajutorul dispozitivelor cu sistem de operare “Windows Mobile”. Totodată, Blackberry căpăta recunoaștere si adopție în rândul americanilor.

Totuși, aceste telefoane inteligente aveau capabilități limitate comparativ cu cele din prezent, marele avantaje fiind accesul la internet și poșta electronică, însă cu viteze limitate (de cel mult 9.6 kbit/s).

Pe parcurs, sistemul de operare Android a început să arate primele semne de adopție, însă mai târziu a devenit o alegere ce merita cu adevărat luată în calcul pentru utilizatorul obișnuit.

Dar, în anul 2007 s-a produs o răsturnare de situație, paradigmă a ceea ce înseamnă un smartphone modern, dar și a modului de construcție a acestuia. Apple lansa în acest an prima versiune de iPhone. Ce aducea în plus era modalitatea de a comunica cu sistemul din partea utilizatorului, și anume un ecran tactil multicolor si foarte sensibil la atingerile utilizatorului. Până atunci acest lucru nu fusese obținut, de cele mai multe ori fiind necesar un stylus (un dispozitiv ce seamană cu un pix care o dată ce atinge ecranul declanșează o acțiune). De asemenea, aplicațiile proprietare aveau o interfață plăcută, navigarea pe internet se făcea mult mai rapid și fluent datorită tehnologiilor moderne de comunicare fără fir dar și a unui browser care era capabil să randeze pagini HTML așa cum am fi văzut pe desktop.

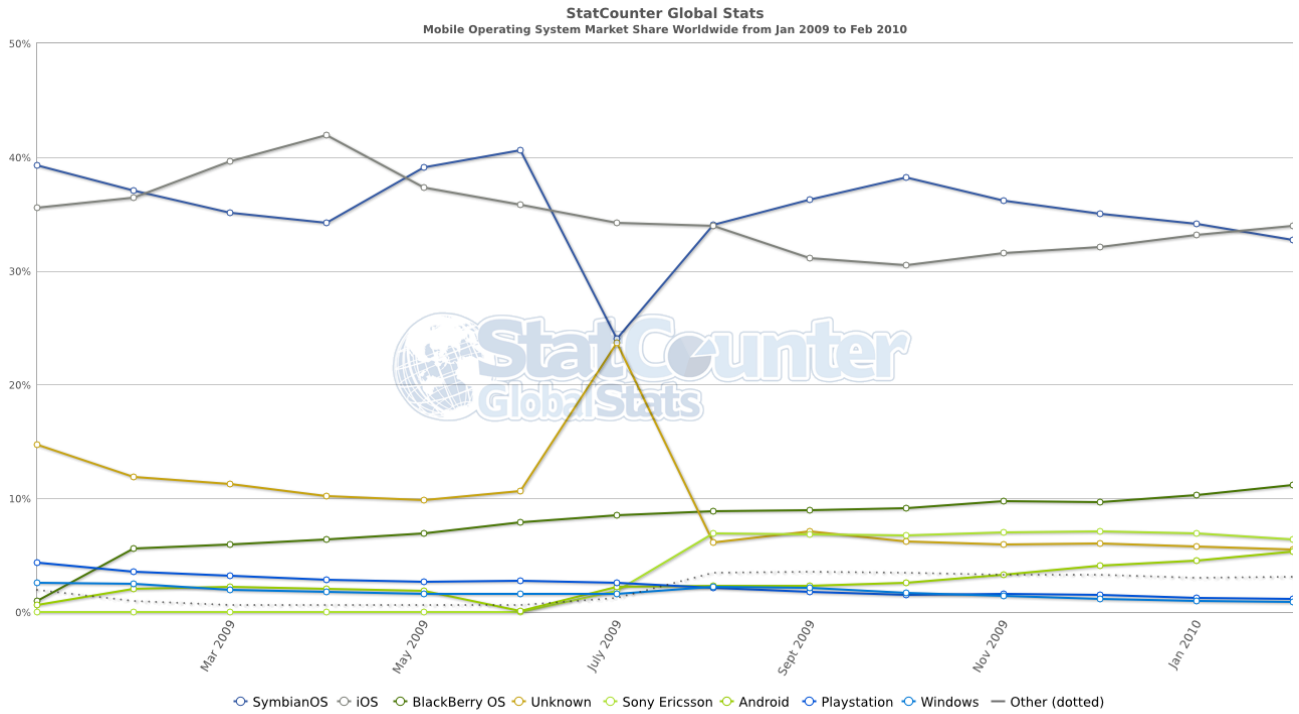
Totuși, sistemul era închis, dezvoltatorii de aplicatii pentru sistemele cu care concura Apple la acel moment, neavând nici o modalitate de a crea si folosi aplicații pentru noul sistem

de operare pus la dispoziție. Însă, în anul următor Apple decide deschiderea sistemului de operare catre dezvoltatori, punându-le la dispoziție un Software Development Kit (SDK) ce le permitea crearea de aplicații, dar si un magazin de aplicații, din care un utilizator poate cumpăra și folosi creațiile noilor entuziaști.

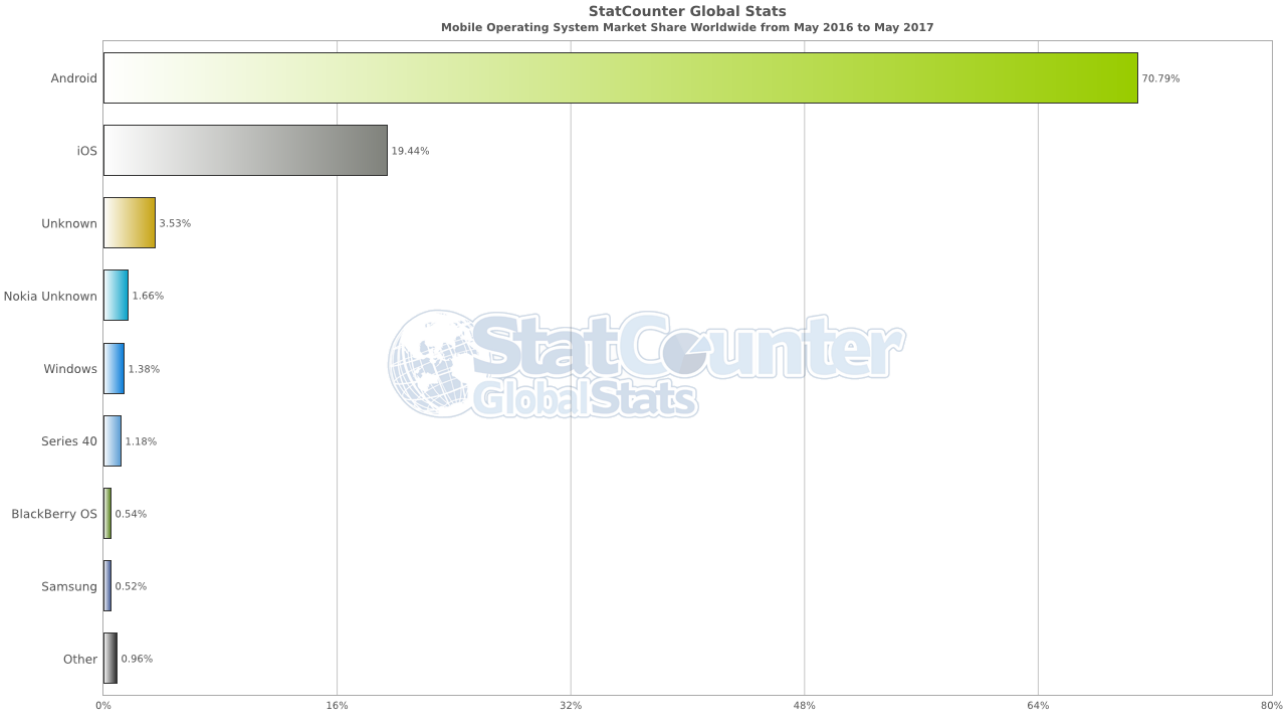
Aceste lucruri descrise mai sus, precum rapiditatea sistemului si un SDK puternic, au făcut din iPhone un adevărat succes atât pentru developeri, cât și pentru utilizatori, datorită varietății de aplicații ce se găseau in App Store, cât și a multitudinilor de probleme pe care acestea le rezolvau.
Figura alăturată descrie numărul de dispozitive vândute în anii 2008-2009, în jur de 15 milioane, atunci când s-a produs această schimbare radicală.

În tot acest timp, sistemul de operare Android devenea prima alternativă serioasă la soluția celor de la Apple, oferea aceleași lucruri, dar mai important, oferea potențialilor cumpărători diversitate în opțiuni, deoarece smartphone-urile Android vin intr-o mulțime de formate de la o mare varietate de producători.

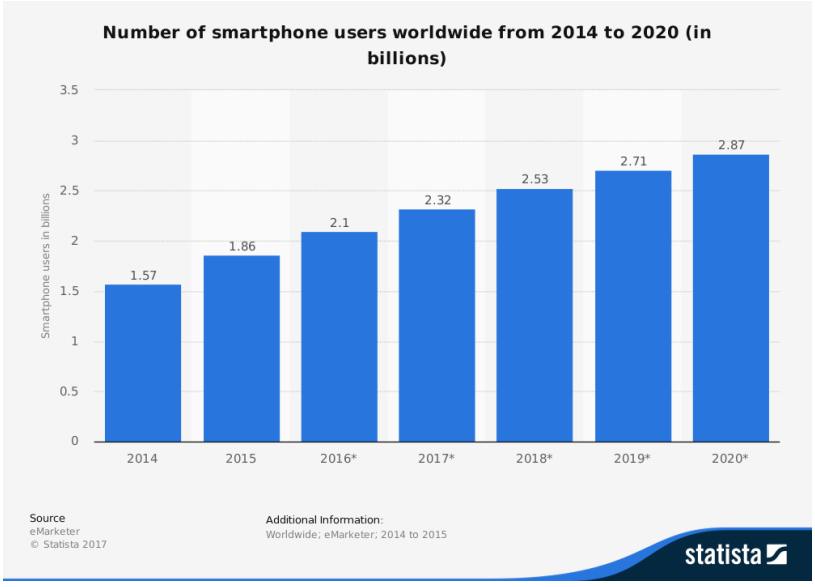
În graficul de mai jos observăm marii competitori din punctul de vedere al sistemului de operare pe piața de telefoane inteligente. Putem observa că din această piață o mare parte, în jur de 35% este a iPhone-ului, o parte aproape la fel de mare este a sistemului de operare Symbian, dar graficul devine interesant în partea de jos, unde putem vedea că pe parcursul anului 2009 Android crește de la un trimestru la altul.



În momentul de față, în anul 2017, cel mai mare sistem de operare, privit din punct de vedere al numărului de utilizatori, este Android, cu peste 70% din total, urmat de iOS (sistemul de operare ce pune în funcțiune iPhone-ul) cu aproximativ 20% din total.



Totuși, o statistică extrem de impresionantă este cea în legătură cu numărul de persoane de pe glob ce folosesc astfel de dispozitive. Mai mult de 30% din populația planetei are acces la un telefon inteligent, ceea ce inseamna peste 2 miliarde de persoane. O proiecție a acestui trend ne arată că există o potențială creștere de până la 2.9 miliarde de persoane ce dețin un smartphone.



Toate aceste informații ne oferă un grad ridicat de certitudine că piața oferită de telefoane inteligente prezintă un loc în care soluțiile la anumite probleme își pot găsi utilizatori cu un grad de ușurință ridicat, atât astăzi, cât și în viitor.

Astfel, luând în calcul faptul că aceste statistici ne oferă date despre utilizatorii de smartphone-uri global, putem să spunem că un grad destul de ridicat de încredere că o mare parte din studenții din facultățile din România, dar mai ales studenții din cadrul Facultății de Informatică a Universității Alexandru Ioan Cuza, sunt utilizatori de telefoane inteligente, în mare parte cu sisteme de operare Android si iOS.

Având în vedere toate aceste chestiuni, îmi propun cu ajutorul prezentei lucrări crearea unei soluții mobile pentru platformele de mai sus ce vine în completarea sistemelor actuale ce impactează, influențează și dictează în cele din urma viața academică a unui student din cadrul facultății noastre din punct de vedere administrativ dar și organizatoric. Accentul a fost pus pe două arii mari : interfața cu utilizatorul trebuie să fie atractivă și simplă, usor de utilizat și fluidă, și pe combinarea, prezentarea și adăugarea informațiilor ce țin de student într-un mod succint și unitar.

Soluția este compusă din două aplicații.

Prima, și cea mai complexă, ce va face și obiectul acestei lucrări, este cea pentru studenți, având rolul de a-l ajuta pe acesta să se mențină la curent cu informații precum orarul, teste, examene sau teme, dar oferă și soluții pentru comunicare.

Cea de-a doua este pentru profesori și secretariat și oferă profesorilor abilitatea de a răspunde mesajelor primite de la studenți și de a adăuga note, teste sau teme în sistem, iar în cazul în care tipul contului nu este de profesor, ci de secretariat va oferi doar funcționalitatea de adăugare anunțuri.

Capitolul 1 analizează sistemul actual (orarul, paginile profesorilor, examenele, temele, notarea, etc) și își propune să ofere câteva concluzii referitoare la acesta ce vor deveni baza sistemului adiacent propus prin intermediul acestei lucrări.

Capitolul 2 propune o instrospecție a tehnologiilor și serviciilor utilizate, argumentarea folosirii lor și prezintă aplicații existente implementate cu aceste tehnologii ce vin în

completarea argumentelor aduse integrării lor în soluțiile descrise mai sus.

Capitolul 3 descrie functionalitățile aplicațiilor, cum funcționează ele împreună pentru a forma un sistem, și cum au fost utilizate tehnologiile prezentate în primul capitol pentru a dezvolta aceste soluții.

CAPITOLUL 1. Sistemul actual

În acest capitol vom analiza pe scurt aplicațiile ce compun actualul sistem administrativ pe care un student din Facultatea de Informatică trebuie să îl utilizeze pentru a fi la curent cu desfășurarea vieții academice dintr-un an de studiu.

1.1 Contextul

De la bun început, un student trebuie sa urmeze o serie de pași pentru a putea intra in mecanismul administrativ ce se desfășoară într-un an universitar. Printre acestea se numără

- crearea unui cont in sistemul eSIMS al facultății pentru a avea acces la notele sale de la final de semestru
- accesarea periodică a contului de poștă electronică pentru a putea comunica cu profesorii săi dar și pentru a primi anunțuri de la secretariat sau cadre didactice
- identificarea grupei din care face parte pentru a putea accesa orarul
- accesarea orarului
- verificarea periodică a paginilor profesorilor pentru a vedea note, anunțuri sau materiale pentru cursuri.

Toate aceste lucruri se desfășoară pe platforme diferite, deoarece orarul are un anumită adresă, fiecare profesor are o pagină separată ce trebuie identificată si accesată, eSIMS-ul are un alt link iar webmail-ul facultății trebuie accesat dintr-o aplicație diferită.

Deoarece toate aceste chestiuni necesită pași separați și informații adiționale de la un context la altul fac ca experiența studentului să fie greoaie. De asemenea, toate aceste sisteme pot fi accesate doar online.

1.2 Situația școlară pe parcursul unui semestru sau an de studiu

Verificarea situației școlare necesită de foarte multe ori consultarea repetată a notelor atât pe eSIMS la final de semestru cât și pe parcursul semestrului pe paginile profesorilor.

Deci, dacă într-un semestru un student are minim 5 obiecte asta va însemna că el trebuie sa verifice cel puțin 5 pagini diferite și să descarce cel puțin 5 documente pentru a putea vedea notele.

De asemenea, aceste pagini sunt formatate în diverse moduri de către deținătorii acestora, însă de cele mai multe ori notele se împart in 4 categorii pentru toți profesorii și studenții, și anume teste, teme, proiecte și examene și conțin ca informații studentul, grupa din care face parte și nota acordată și data acordării notei.

Având in vedere aceste lucruri, apare necesitatea unui sistem care sa înglobeze cele două părți, înregistrarea notei de către profesor și vizualizarea ei de către student.

1.3 Orar

Orarul este instrumentul oferit studentului pentru a ști atât cu ce profesori va lucra în acest semestru cât dar mai important este orarul în sine.

Aceste lucruri aduc cu sine aceeași problema a accesării în mod repetat pe parcursul semestrului o resursă care este cu totul alta decât restul pe care trebuie să le utilizeze, și anume pagina orarului ce conține la rândul ei alte pagini pentru fiecare grupă în parte.

De asemenea, datele de contact ale profesorilor cu care va avea cursuri sau seminarii vor trebui preluate pe baza informațiilor din orar referitoare la profesorii ce țin orele respectivelor grupe din paginile fiecărui cadru didactic în parte.

Din nou, se face apel la laitmotivul acestei lucrări, și anume, aducerea către student a acestor informații într-un mod unitar si într-o singură locație.

1.4 Comunicarea

Comunicarea între studenți și profesori este ceva ce se petrece nu doar la curs sau seminar, ci pe parcursul întregului semestru cu ajutorul email-ului și a paginilor profesorilor unde aceștia pot posta anunțuri.

Din nou, avem o nouă platformă ce trebuie accesată, și anume webmail-ul, pe langă paginile profesorilor.

De asemenea, prin intermediul email-ului se trimit anunțuri din partea secretariatului către toți studenții sau doar pentru cei dintr-un anumit an, dar și profesorii pot trimite astfel de email-uri.

Din nou, faptul că există mai mult de un loc unde se petrece această comunicare virtuală implică eforturile de a fi la curent cu toate aceste platforme.

1.5 Concluzii și obiective

După cum putem observa, un student pe parcursul unui semestru trebuie să acceseze o multitudine de platforme, aplicații web și site-uri pentru a fi la curent cu toată activitatea.

De asemenea, toate aceste sisteme si aplicații nu sunt concepute pentru a fi consumate pe telefoane inteligente, care au ecrane, ca dimensiuni, mult mai mici decat cele ale laptop-uri, ceea ce duce la o experiență neplăcută în utilizarea lor. De asemenea ele nu ofera suport pentru navigarea lor in mod offline și uneori ne putem găsi in situația în care nu putem avea acces la internet.

Acest lucru a oferit oportunitatea conceperii și gândirii unui obiectiv, și anume un sistem mult mai simplu și coerent care să ofere o bază pentru cercetarea unor soluții moderne ce pot veni în completarea celor existente deja în facultatea noastră și care pot fi accesate atât de pe telefon (online și offline) cât și de pe un calculator personal conectat la internet.

CAPITOLUL 2. Tehnologiile utilizate

Capitolul de față va realiza o trecere prin tehnologiile folosite în crearea suitei de aplicații necesară pentru a atinge îndeplinirea obiectivului de mai sus, și anume un sistem mai simplu si mai eficient care să le ofere studenților acces la informațiile relevante dintr-un an universitar, dar si o modalitate de comunicare mai comodă și directă între profesori si studenți, dar și secretariat și studenți.

2.1 Contextul

După cum am menționat în introducerea tezei, dispozitivele mobile prezintă o mare parte din viața studenților si pentru mulți dintre noi reprezintă unealta pe care o folosim cel mai des pentru a consuma conținut de orice fel.

De aceea, aplicația cu ajutorul căreia studentul va avea acces la datele dintr-un semestru universitar va fi una mobilă, dezvoltată atât pentru sistemul de operare iOS cât și pentru Android.

Pe de altă parte, aplicația destinată profesorilor este una web, pentru browser. Deoarece obiectivul acestei aplicații de cele mai multe ori va fi introducerea de note pentru studenții unei anumite grupe, experiența cadrului didactic va fi una mult mai eficientă astfel.

2.2 Tehnologiile folosite la dezvoltarea aplicației mobile

Deoarece obiectivul a fost crearea unei aplicații atât pentru Android cât și pentru iPhone soluția găsită a fost folosirea unor tehnologii cross-platform. Aceasta înseamnă că același cod odată scris poate fi executat pe mai multe sisteme de operare, cu mai multe sau mai puține modificări.

De asemenea, având în vedere că datele și modul în care acestea sunt servite către clienți trebuie să fie unul cât mai agnostic de tehnologiile clienților cu putință, deoarece vrem ca acestea să fie ușor folosite și integrate de catre orice tehnologie.

Astfel, printre tehnologii, cele mai importante sunt React și React-Native, Angular 1 și Firebase.

2.2.1 React și React Native

Problema aplicațiilor ce au același cod la bază și rulează pe mai multe sisteme de operare nu este una nouă și a avut parte de nenumărate soluții mai mult sau mai puțin de succes.

Până de curând, o aplicație pe mobil hibridă însemna o aplicație ce rula într-un container denumit generic WebView, cu implementări pe majoritatea sistemelor de operare mobile, inclusiv iOS si Android, ce este de fapt o aplicație web, ce randează HTML si CSS pentru interfață si execută cod Javascript pentru logică.

Cu ce este acest lucru diferit față de un site web rulat în browser? Faptul că aplicația poate accesa prin intermediul acestui container WebView diverse funcții ale sistemului de operare, precum camera foto sau funcții ale tastaturii și chiar baze de date locale de tip SQLite (folosite acum si pe web, dar mai ales pe aplicațiile native Android). În schimb depinde de implementarea motoarelor de randare HTML/CSS de pe dispozitivul selectat, ceea ce poate duce la efecte vizuale diferite pe platforma Android față de platforma iOS si viceversa.

Însă acest tip de aplicație nu oferea o viteză a interfeței foarte bună și nici a execuției codului Javascript.

Astfel, paradigma aplicațiilor hibride (sau cross-platform) pe telefoane inteligente, a fost schimbată cu ajutorului unui nou framework dezvoltat de Facebook, denumit React Native, ce își are rădăcinile în librăria pentru aplicații web, React, dezvoltată tot de ei.

React este cunoscut drept acea librărie care a încercat și chiar reușit să vină cu o nouă abordare a dezvoltării pentru web, și anume All-in-JS (totul în Javascript). All in Javascript înseamnă că un element al unei pagini web, de exemplu un buton ce este folosit în intreaga aplicație, este scris sub forma unei clase(în noul ES6) sau sub forma unei functii React.createClass(…) ce înglobează atât partea vizuală cât și partea logică și care poate fi exportat către aplicație si folosit oriunde este nevoie.

Acest lucru permite crearea de aplicații web scrise cu ajutorul React să implementeze mult mai bine conceptul de programare Separation Of Concerns.

Argumentul adus este că Separation of Concerns nu înseamnă separarea după tipul de fișier sau tehnologie, ci după logică, deoarece o pagina web modernă este compusă din toate tehnologiile Web și anume HTML, CSS, JS si de cele mai multe ori probleme pe care programatorii web le rezolvă se întind pe toate macar doua din aceste trei tehnologii la un moment dat, nu doar pe una dintre ele.

React-Native este un framework ce preia paradigma din spatele React și oferă o soluție pentru dezvoltarea aplicațiilor hibride de o calitate mult mai inaltă celor dezvoltate într-un WebView.

Cum funcționează?

React-Native pune la dispoziție o serie de elemente ce se comportă și arată precum tag-urile HTML (această suită de elemente se numește limbajul JSX), însă ele sunt inspirate din elementele ce se găsesc în aplicațiile mobile. Adică, vom găsi ListView pentru afișarea de liste, ScrollView pentru randarea unei componente ce are o înălțime mai mare decât cea a ecranului, etc.

O dată scris aceste elemente, compilatorul framework-ului va prelua limbajul JSX și va crea elemente native din sistemul de operare ce vor fi mai departe afișate utilizatorului. Acest lucru este posibil deoarece sistemele de operare Android și iOS permit injectarea și executarea de cod Javascript și framework-ul conține si elemente native ce comunică cu elementele Javascript.

În acest fel, experiența utilizatorului este exact ca cea oferită de o aplicație nativă scrisă in Java(Android) sau în Objective-C (iOS).

Acest framework reprezintă baza aplicației, și fără aceste noțiuni minime nu vom putea

înțelege de ce s-a ajuns la un anumit rezultat și nu altul, de ce aplicația acționează într-un anumit fel și nu altul.

2.2.2 Firebase

Firebase este alegerea de-facto pentru soluții de back-end pentru aplicațiile mobile scrise cu ajutorul React-Native.

Oferă o serie de soluții ce permite autentificarea utilizatorilor, mesagerie în timp real sau baze de date în timp real fără ca dezvoltatorii ce utilizează Firebase să trebuiască să mențină o infrastructură în acest sens.

Credibilitatea îi este oferită atât de faptul că este un produs dezvoltat de Google și este folosit de companii precum Shazam (cea mai mare aplicație de recunoaștere de muzică), The Economist (una dintre cele mai apreciate publicații jurnalistice din lume) sau Trivago (unul dintre cele mai folosite servicii pentru găsirea de hoteluri).

Serviciile Firebase folosite pentru a dezvolta aplicația de mobil, dar și clientul web sunt cele de autentificare și baze de date.

Baza de date este una NoSQL, însă tehnologia folosită pentru a pune la dispoziție un astfel de serviciu nu este public afișată de Google. Acest serviciu permite atât interogari foarte rapide cât și scrieri ce declanșează ca clienții conectați la baza de date să primeasca noile modificări imediat.

2.2.3 EcmaScript 6

Ultima versiune a limbajului de programare EcmaScript (sau Javascript) aduce cu sine o mulțime de îmbunătățiri limbajului. Două dintre ele, care stau la baza aplicației de mobil, sunt clasele și noua modalitate de a scrie cod asincron în Javascript cu ajutorul funcționalității async/await.

Implementarea claselor în EcmaScript 6 (ES6) este de fapt doar o notație mai frumoasă a modelului de moștenire “prototype”, însă oferă avantajul unui cod concis, mult mai ușor de citit.

Pe de cealaltă parte funcționalitatea async/await este o noua modalitate de a rula cod asincron, față de promisiunile sau callback-urile clasice. Printre avantaje se numără evitarea apelului “then(…, …, …)” și crearea funcțiilor anonime date ca paremetru acestei funcții.

Promisiunile sunt rezultatele completării cu succes sau eroare ale unor operații ce se desfășoară asincron. Acestea permit dezvoltatorilor să facă apeluri la resurse de pe alte servere în timp ce alte părți de cod se execută.

2.3 Tehnologiile folosite la crearea aplicației web

Deoarece obiectivul aplicației web este acela de a oferi cadrelor didactice o interfață rapidă si simplă către adăugarea de note, teme sau teste și de a răspunde mesajelor de la studenți am apelat la cel mai folosit framework de dezvoltare de aplicații web, și anume Angular 1.

Acest framework este unul de tip MVW (Model View Whatever), deoarece pentru a crea liantul intre model si interfață, acesta ne propune, pe langa Controllere, și Directive sau Servicii.

Directivele sunt bucăți de cod ce conțin o funcție ce crează logica și un template HTML și sunt folosite în general pentru a crea componente reutilizabile în interiorul aplicației.

Angular are multe avantaje, printre care viteza cu care se pot dezvolta aplicații web cu ajutorul lui, dar și “2-way-data-binding”. Cel din urmă ne permite să adăugăm valori aflate pe this-ul controller-ului atașat unui anumit template și atunci când acestea se modifică in interior-ul controller-ului automat template-ul va reflecta această schimbare.

De asemenea, utilizăm aceeasi bază de date NoSQL de pe serverele Firebase cu avantajele acestui serviciu evidențiate mai sus.

2.4 Concluzii

În concluzie, aplicația mobilă face uz de ultimele tehnologii în cea ce privește dezvoltarea de aplicații mobile ce oferă o experiență de utilizare foarte apropiată de cea a aplicațiilor native, cu avantajul că nu trebuie să dezvoltam de două ori pentru doua platforme.

De asemenea folosim unul din cele mai utilizate servicii de aplicații de tip “serverless”, unde nu trebuie să creăm infrastructură si ce ține toți clienții la curent cu ultimele schimbări din baza de date.

Aplicația web este implementată cu ajutorul celui mai cunoscut framework pentru front-end ce oferă o flexibilitate sporită si rapiditate în dezvoltare, cu numeroare resurse pe internet.

CAPITOLUL 3.Prezentarea soluției

Acest capitol are rolul de a prezenta cum funcționează aplicația de mobil. Vom vedea cum este structurată baza de date dar si cum comunică aplicația cu ea, cum am folosit tehnologiile prezentate în capitolul doi si vom motiva pe exemple de ce tehnologiile alese oferă o bază solidă pe care se poate dezvolta mai departe.

3.1 Autentificarea și baza de date

Orice aplicație care are nevoie de date are nevoie si de un back-end care să asigure o modalitate de autentificare dar și stocarea și preluarea datelor.

După cum am evidențiat mai sus, pentru acest lucru vom utiliza serviciile Firebase pentru autentificare și bază de date în timp real.

Singura modalitate prin care un utilizator se poate loga în aplicația noastră e cu un cont de email și o parolă. Din clientul mobil sau cel de web nu este posibilitatea înregistrării în sistem.

Acest lucru urmează calea firească a lucrurilor, și anume, mai întâi utilizatorul trebuie să devină un student al Facultății de Informatică, si abia după aceasta i se crează un cont cu o adresă de email cu domeniul info.uaic.ro și o parola, iar abia după ce se întâmplă toate aceste lucruri, studentul primește credențialele și se loghează în aplicație.

Din acest punct de vedere, Firebase ne oferă control total, și putem crea userii ce au drept să se logheze în aplicație, să îi ștergem sau să le edităm parola, după cum putem observa în figura de mai jos (fig. 3.1).

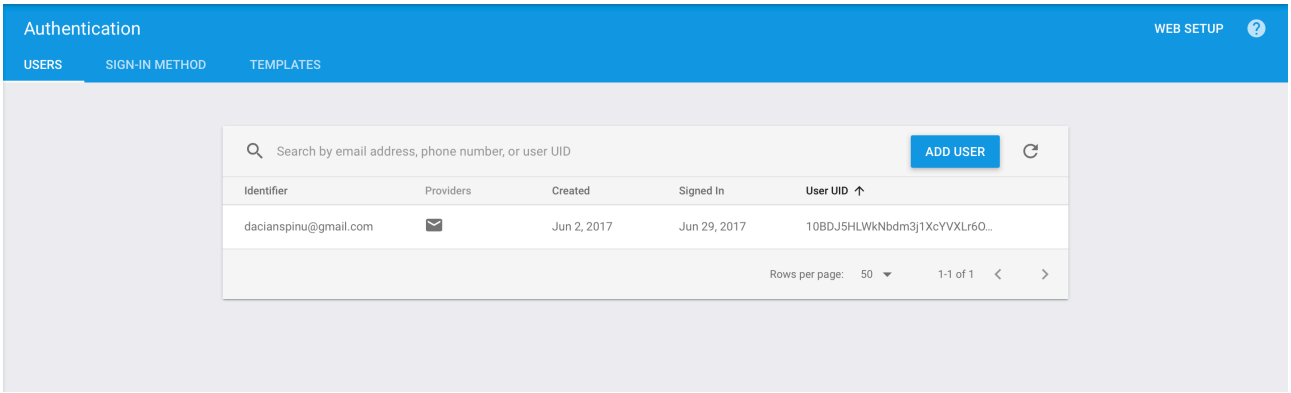
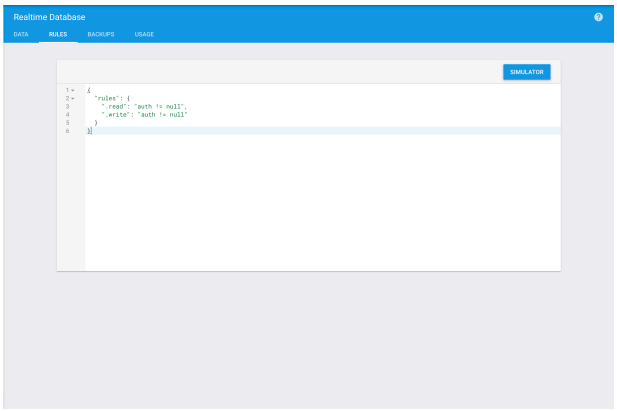


Figura 3.1 *Panoul de administrare al utilizatorilor*

De asemenea, Firebase oferă numeroase modalități de autentificare, precum autentificare cu numărul de telefon, Facebook, Twitter sau chiar utilizare anonima, însă singura de care avem nevoie este cea cu adresă de e-mail și parolă.

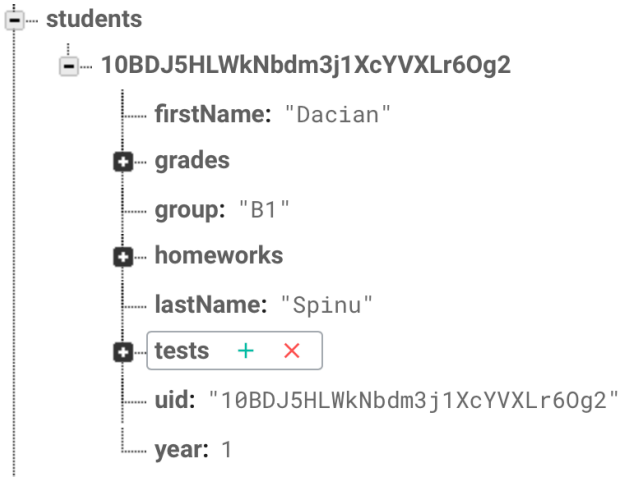
3.2.1 Structura bazei de date

În primul rând nu putem face operațiuni pe baza de date decât dacă suntem autentificați. Acest lucru este obținut cu ajutorul a două reguli simple, ce arată în felul urmator:

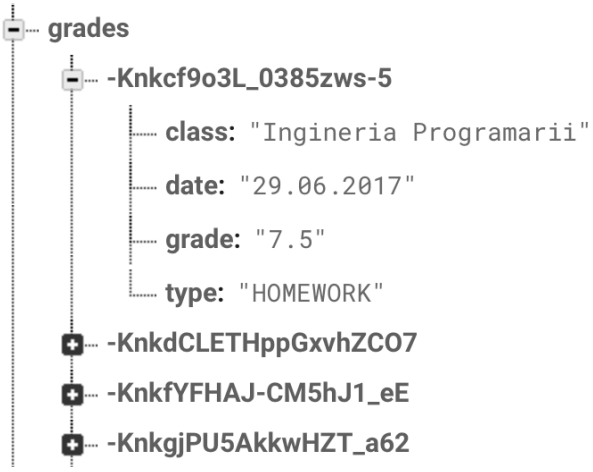



```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

În baza de date există o serie de entități. Prima entitate ce se va crea în baza de date este cea a studentului, după ce acestuia i se crează contul cu adresa de email si parola. Studentul va avea un nume (`lastName`), prenume (`firstName`), va fi atașat la o grupă (`group`) și un an de studiu (`year`), va avea teme (`homeworks`), teste (`tests`) si note (`grades`).



Entitatea de note (`grades`) conține informații precum nota, tipul notei (examen, temă, test, proiect), data la care a fost pusă această notă și obiectul la care s-a primit nota. În momentul adăugării unei entități de tip `grade` acesteia i se generează un id unic (unique identifier), astfel încât dacă doi profesori adaugă simultan note în baza de date pentru studenții de la aceeași grupă acestea nu vor intra în conflict.

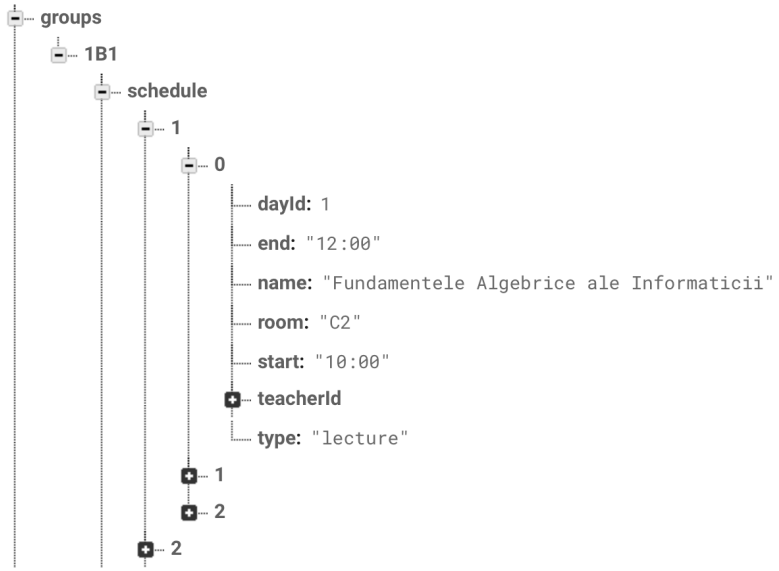


Entitatea de teme (`homeworks`) conține informații precum obiectul la care s-a dat acea temă, un titlu și o descriere si data la care trebuie predată. Aceleași reguli precum la entitatea de tip `grade` în ceea ce privește adăugarea în baza de date se aplică și in cazul intrării de tip `homework`.

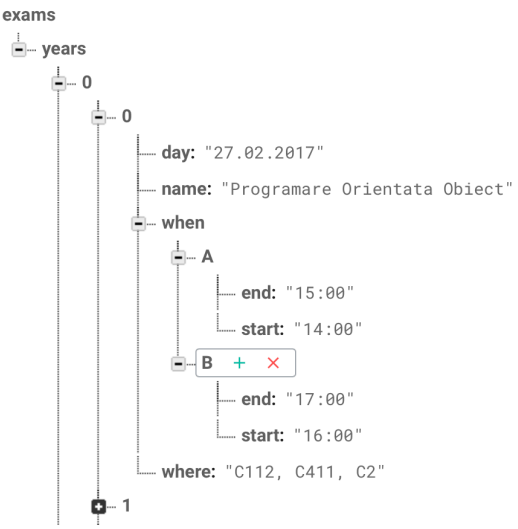
Următoarea entitate în ordinea firească a lucrurilor este cea a grupei, care în baza noastră de date are forma “AnDeStudiuSemianNumărulGrupei”. Adică studentul Onofrei din anul 1, grupa B1 va face parte din grupa 1B1.

Entitatea grupă (`group`) are atașat ei un orar (`schedule`) ce conține cheile 1, 2, 3, 4, 5, 6 ce numerotează zilele în care au loc ore în Facultatea de Informatică.

Fiecare cheie ce reprezintă una din zilele saptămânii este un `array` de obiecte, de tip `schedule entry`. Informațiile de care dispunem sunt ora de început și ora de final a acestui `schedule entry`, în ce sală se desfășoară si tipul intrării (laborator sau curs, valorile acceptate în baza de date fiind `lab` și `lecture`).

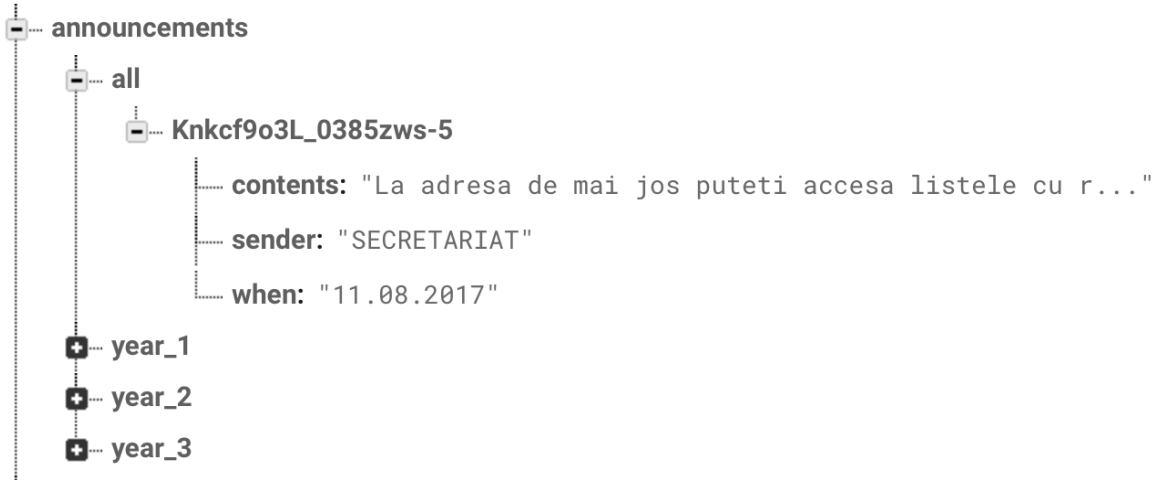


Următoarea entitate din listă este cea de examen (`exam`). Examenele sunt împarțite pe ani de studiu (`years`) numerotați 0, 1, 2, și fiecare astfel de cheie conține o listă de examene, un element din listă având următoarele atribute: ziua în care se desfășoară examenul (`day`), numele obiectului la care se dă examen (`name`), în ce săli se va desfășura examenul (`where`), si la ce oră va incepe și se va termina examenul (`when`) pentru fiecare semi-an în parte (cheile `A` și `B`).



Poate cea mai simplă structură din baza de date este cea de anunțuri (`announcements`) și reprezintă anunțurile trimise de către secretariat sau cadre didactice către studenții facultății. Această structură conține patru chei, `all` (anunțuri trimise către toți studenții), `year_1` (anunțuri trimise către studenții din anul întâi), `year_2` (anunțuri trimise către studenții din anul doi), `year_3` (anunțuri trimise către studenții din anul 3).

Un obiect de tip `announcement` prezintă următoarea structură: conținut (`contents`), și descrie textul ce va fi trimis și afișat în aplicația de mobil a studenților, emițătorul anunțului (`sender`), care poate fi secretariatul facultății sau un cadru didactic, și data la care acesta a fost trimis (`when`).



Structura la care s-a ajuns cel mai greu este cea de conversație (`conversation`). O conversație este o listă de mesaje între un student și un profesor. Ne vom îndepărta de ideea de conversație pe care o abordează aplicații precum WhatsApp sau Facebook Messenger, și ne vom apropia mai mult de cea de email, în care o conversație este un șir de mesaje care la un moment dat se termină deoarece subiectul nu mai are relevanță de la un anumit moment dat încolo. Astfel, între un profesor și student pot exista numeroase conversații de acest fel.

La momentul inițierii conversației de către un student se vor întampla 3 lucruri:

- crearea conversației în baza de date
- preluarea identificatorului conversației și
    - introducerea sa în lista de conversații a studentului
    - introducerea sa în lista de conversații a profesorului

O conversație (`conversation`) are o listă de mesaje (`messages`), iar fiecare obiect de tip mesaj are următoarele proprietăți: emițătorul mesajului (`sender`), textul mesajului (`text`), data la care a fost trimis mesajul (`sentDate`).

3.2 Aplicația mobilă

Aplicația mobilă este primul lucru care comunică cu serviciile Firebase descrise mai sus. Ea va prelua în mod asincron date de la server și le va afișa pe parcursul navigării studentului, atât online cât și offline.

3.2.1 Stocare pe dispozitiv și preluare de date în modul offline

Aplicația funcționează in modul offline făcând verificări la fiecare scenă asupra conectivității la internet. Dacă nu există una, atunci vom prelua datele din fișierele salvate local. De asemenea ascultăm la evenimentul de schimbare

Pentru acest lucru framework-ul ne pune la dispoziție o clasă denumită `AsyncStorage` pe care o vom folosi pentru a salva și prelua datele de pe dispozitiv, și o clasă denumită `NetInfo` ce ne poate oferi detalii despre starea dispozitivului (conectat la wireless, date mobile sau neconectat).

```
            NetInfo.fetch().done(
                    async (networkType)=> {
                      if (networkType === 'none') {
                        schedule = await this.getLocalSchedule();
                        this.computeTabNames(schedule);
                      } else {
                        schedule = await this.getSchedule();
                        this.computeTabNames(schedule);
                      }
                    }
                )
```

Acest lucru permite consultarea orarului, a notelor, temelor, testelor, vizualizarea conversațiilor și a anunțurilor fără a avea acces la internet. Evident, această funcționalitate depinde de o sincronizare anterioară a datelor pe dispozitivul mobil pentru a putea fi ulterior preluate.

Sincronizarea în memoria smartphone-ului se realizează la fiecare apel la server pentru a putea fi siguri că studentul beneficiază de ultimele date.

`AsyncStorage` permite acest lucru cu ajutorul unui apel de tipul:

```
AsyncStorage.setItem('currentStudentSchedule', JSON.stringify(schedule));
```

Astfel, pe sistemul de operare iOS se va crea un dicționar serializat în memoria destinată aplicației noastre pentru fișiere destul de mici sau va crea un fișier cu totul separat pentru fișiere mari, iar sistemul de operare Android va folosi o bază de date locală, RocksDB sau SQLite.

Preluarea datelor din memoria locală se va efectua cu ajutorul unui apel de tipul:

```
let student = await AsyncStorage.getItem(‘currentStudent');
```

ce va returna un string ce trebuie parsat cu ajutorul JSON.parse(string).

3.2.2 Preluarea de date în mod online

Cu ajutorul funcțiilor `async` și a apelurilor de metode precedate de cuvântul rezervat `await` vom interoga baza de date prin intermediul clasei `Database` ce pune la dispoziție metode pentru a prelua date de la server în funcție de necesități. Spre exemplu, pentru cazul de mai sus, avem nevoie de informații legate de orarul grupei din care face parte studentul si de profesorii ce predau materiile din orarul respectiv.

Astfel, înainte ca scena ce afișează orarul să fie desenată, facem un apel la aceste două resurse în același timp folosind metoda `all()` implementată în clasa `Promise`, precedată de un

apel await care astfel returnează valoarea pe care o trimite serverul, in caz de succes.

```
      let data = await
Promise.all([Database.getLoggedInStudentGroupSchedule(JSON.parse(student).gro
up, JSON.parse(student).year), Database.getTeachers()]);
```

Apelurile Database.getLoggedInStudent si Database.getTeachers sunt apeluri la baza de date, ce arată în felul următor

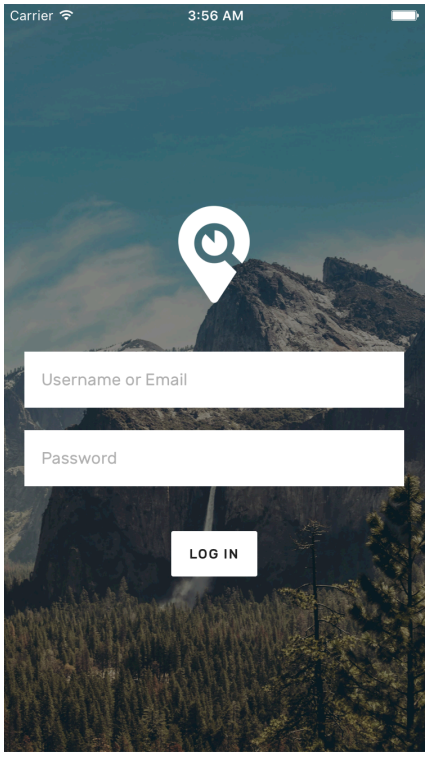
```
      static getLoggedInStudentGroupSchedule(groupId, year) {
      let path = "/groups/" + year + groupId;
      console.log(path)

      return firebase.database().ref(path).once('value');
    };
```

Acest tip de apel returnează o promisiune ce va fi rezolvată in cadrul primului apel de tip `await`.

Aceste lucruri se întamplă pentru fiecare componentă la începutul ciclului de viață (component lifecycle), și anume la apelul automat al funcției `componentWillMount()`. Această funcție este pusă în funcțiune doar atunci când este declarată în interiorul clasei ce crează respectiva componentă.

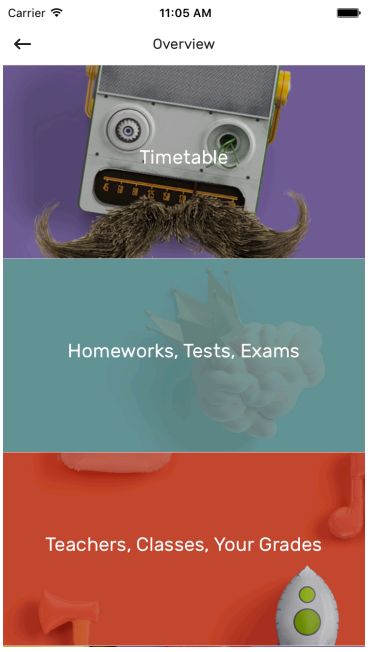
3.2.3 Autentificarea



După cum am spus, autentficarea studentului se va face cu ajutorul Firebase și este limitată doar la cea pe bază de adresă de e-mail și parolă. Acesta este primul ecran afișat utilizatorului în cazul în care nu este autentificat. Dacă este autentificat vom ajunge pe ecranul de categorii.

O dată cu introducerea credențialelor și apăsarea butonului LOG IN vom face un apel la serviciul de autentificare al Firebase iar în caz de succes vom naviga automat la următoarea scenă.

3.2.3 Meniul aplicației



Scena care crează legăturile cu toate celelalte ecrane este cea de Overview (privire de ansamblu) și afișează ce categorii poate accesa studentul.

Acestea sunt

- Timetable (Orar)
- Homeworks, Tests, Exams (Teme, Teste, Examene)
- Teachers, Classes, Your Grades (Profesori, Materii, Notele Tale)
- Announcements (Anunțuri)
- Communication (Comunicare)

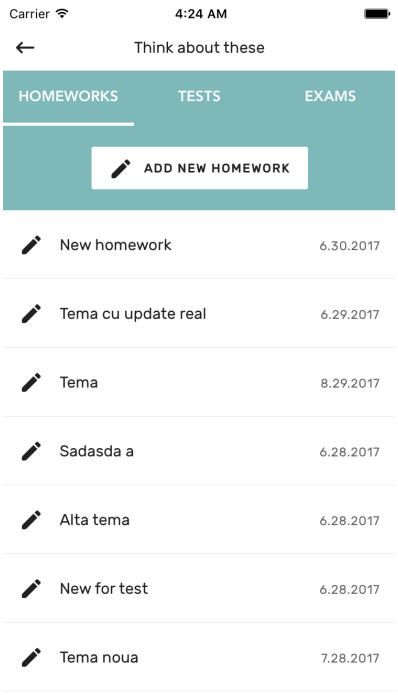
3.2.4 Ecranul pentru orar

Ecranul pentru orar este unul compus din 3 secțiuni care se schimbă pe baza zilei în care este accesat după următoarele reguli:

- de luni până joi vom afișa orarul pentru ziua curentă și următoarele două zile (de ex. Marti, Miercuri, Joi)
- dacă ziua curentă este joi vom afișa orarul pentru ziua curentă, urmatoarea zi și prima zi din saptămâna urmatoare
- dacă ziua curentă este vineri, vom afișa orarul pentru vineri și următoarea zi și prima zi din săptămâna următoare
- dacă ziua curentă este duminică pentru prima secțiune nu vom afișa nimic, iar următoarele două secțiuni vor conține orarul pentru zilele de luni și marți.

În figura alăturată putem observa cea de-a treia situație.

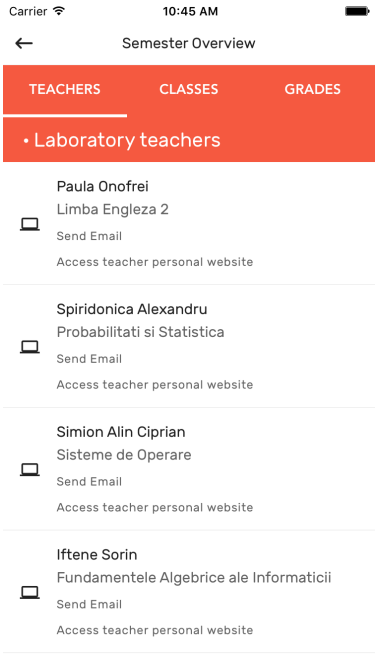
3.2.5 Ecranul pentru Teme, Teste și Examene

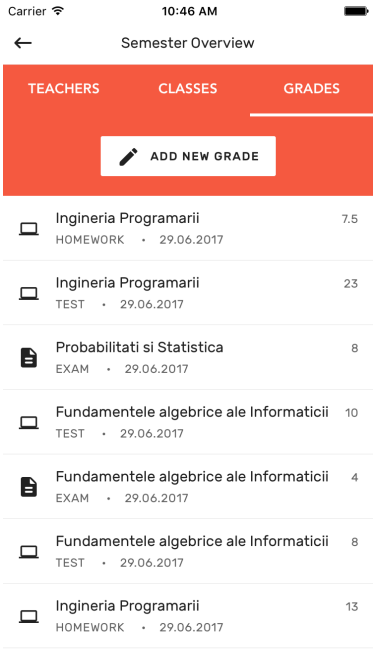


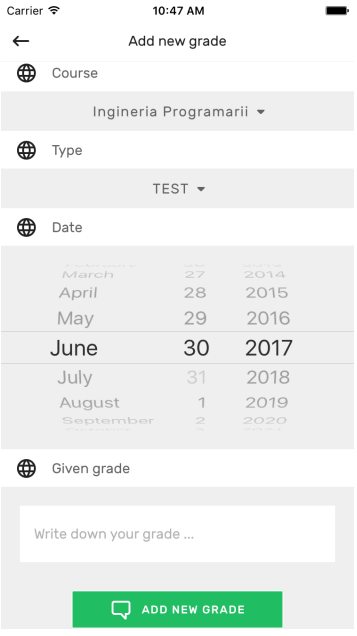
Din nou, acest ecran este compus din 3 secțiuni, fiecare corespunzând uneia dintre categorii. Datele pentru teme și teste provin din profilul studentului. Ele pot fi adăugate atât de student cu ajutorul unui ecran dedicat de adăugare cât si de către profesor dacă face acest lucru cu ajutorul aplicației web, iar cele pentru examene din structura `exams` corespunzătoare anului de studiu în care se află studentul.

În figura alăturată observăm ecranul cu temele studentului. Ecranul pentru teste arată asemanator.

3.2.6 Ecranele Profesori, Materii și Note





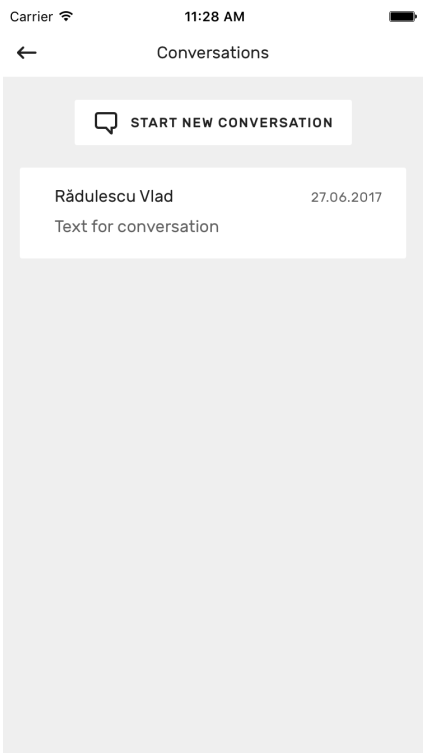


Datele legate de profesori și materii sunt strâns legate de orarul studentului. Astfel, pentru a putea afișa aceste informații este nevoie de accesarea orarului studentului din baza de date, la fel ca în primul ecran și maparea `id`-urilor profesorilor din câmpul `teacherId` la

informațiile primite prin apelul la server ce ne returnează profesorii. Aceste două apeluri se fac asincron precum s-a prezentat în sub-capitolul *3.2.2 Preluarea datelor în mod online.*

Astfel, am putut extrage profesorii ce predau laboratoare sau seminarii într-un obiect separat de cel ce înglobează profesorii de curs. De asemenea am oferit studentului posibilitatea de a-și adăuga singur notele în listă în cazul în care profesorul preferă sa nu facă acest lucru în anumite situații.

3.2.7 Ecranele pentru comunicare



După cum am spus si in Capitolul 1, o mare parte din comunicarea dintre studenți si profesori se petrece în mediul virtual, cu ajutorul mail-urilor sau anunțurilor pe paginile personale ale cadrelor didactice.

Astfel, modulul de comunicare din aplicație poate oferi o alternativă la comunicarea clasică unu la unu de pe mail, având avantaje precum salvarea pe dispozitiv a mesajelor și faptul că poate fi accesată foarte simplu prin intermediul prezentei aplicații.

Concluzii

După cum am menționat în concluziile primului capitol al prezentei lucrări, acest sistem are rolul de a veni în completarea celor deja existente si de a oferi o experiență mai plăcută utilizatorilor. De asemenea prezenta lucrare poate servi ca o bază pentru dezvoltări ulterioare ale produsului și eventual chiar adoptarea sa. Având în vedere tehnologiile utilizate si modalitatea în care acestea sunt folosite, consider că această lucrare aduce un grad de noutate atât prin prisma acestora, cât și prin prezentarea ei către utilizator si încercarea găsirii unei soluții la problema prezentată de existența a nenumărate aplicații pe care un student trebuie să le folosească zilnic pe parcursul unui an universtar.

Direcții în care aplicația poate fi extinsă se găsesc în modalitatea de a introduce datele în baza de date, deoarece o mare parte din ele trebuie introduse manual. Așadar o aplicație care să permită management-ul tuturor datelor, de la studenți si profesori și până la orar.

De asemenea, extinderea funcționalității către studenții cu restanțe este un alt lucru care ar fi binevenit în aplicație, notificări push pe device atunci când se adaugă o notă sau o temă.

Bibliografie

1. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
2. http://gs.statcounter.com/os-market-share/mobile/worldwide#monthly-200901-201002
3. http://gs.statcounter.com/os-market-share/mobile/worldwide
4. https://facebook.github.io/react-native/
5. https://firebase.google.com/docs/
6. https://docs.angularjs.org/api