**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

****

LUCRARE DE LICENȚĂ

**Student Assistant**

**propusă de**

***Marius-Cătălin Mihai***

**Sesiunea:** *februarie, 2020*

**Coordonator științific**

Colab. Florin Olariu

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

Student Assistant

*Marius-Cătălin Mihai*

**Sesiunea:** *februarie, 2020*

**Coordonator științific**

*Colab. Florin Olariu*

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ………………………. specializarea …………………………………………………………, promoția …………………………., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din 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, *data*

Absolvent *Prenume Nume*

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

(semnătura în original)

C**uprins**

[Introducere 8](#_Toc12449354)

[Motivație 8](#_Toc12449355)

[Gradul de noutate 8](#_Toc12449356)

[Metodologia folosită 8](#_Toc12449357)

[Structura lucrării 9](#_Toc12449358)

[Contribuții 10](#_Toc12449359)

[Capitolul 1: Descrierea problemei 11](#_Toc12449360)

[Capitolul 2: Descierea soluției 12](#_Toc12449364)

[Concluziile lucrării 32](#_Toc12449386)

[Bibliografie 33](#_Toc12449387)

[Anexe 34](#_Toc12449388)

# Introducere

In zilele noastre, datorita avansurilor tehnologice si a faptului ca traim in era informatiei, dorim sa fim conectati constant si sa avem acces la informatii intr-un mod cat mai rapid, mai facil si mai centralizat. Tinand cont de acest aspect al cotidianului, consider ca aceasta norma a interconectarii rapide si facile se aplica nu numai in viata noastra sociala, cat si in ceea ce priveste studiile noastre, mediul universitar fiind un candidat excelent pentru a aplica noi aceste standarde de comunicare si de transfer de informatii.

## Motivație

Fiind un absolvent al acestei facultati, si un pasionat al aplicatiilor mobile, mi-am dorit sa creez acest mediu interconectat, in care informatiile sunt transmise rapid si facil atat catre studenti cat si catre profesori si in acest sens, am realizat aceasta aplicatie, Student Assistant.

Scopul aplicatiei este acela de a transmite informatii in timp real de la profesori la studenti, prezentand celor din urma toate informatiile referitoare la facultate, intr-un mod centralizat si intuitiv.

## Gradul de noutate

Pana la momentul actual, nu au fost implementate aplicatii mobile ce doresc a oferi toate functionalitatile aplicatiei Student Assistant, aceasta fiind o inovatie adusa facultatii.

Este de mentionat faptul ca exista servicii ce expun aspecte similare celor oferite de catre Student Assistant (precum WebMail – transmiterea de anunturi si mesaje, eSims – gestionarea notelor si a situatiei scolare, orarul FII), insa Student Assistant ofera avantajul portabilitatii si a centralizarii datelor, atat studentul cat si profesorul avand acces in timp real la aceste informatii cu o simpla atingere de ecran.

## Metodologia folosită

Pe parcursul elaborării licenței am încercat să lucrez cât de cât structurat, pe etape, neavând în minte de la început o imagine clară a unui produs final. Am inceput cu luarea deciziilor referitoare la arhitectura aplicatiei, decizii precum o abordare server-less versus construirea si hostarea unui server, alegerea unei baze de date, reflectand in acelasi timp asupra posibilelor functionalitati ce ar putea fi impactate de catre luarea acestor decizii. In urma acestor decizii am putut stabili cum vor fi implementate diversele functionalitati ale aplicatiei si am putut elabora o diagrama arhitecturala a aplicatiei, pe baza careia am inceput dezvoltarea.

Atat pentru a avea un backup al aplicatiei, cat si pentru a putea lucra de pe mai multe calculatoare, am folosit Git-ul ca si sistem de versionare, asigurandu-ma astfel ca pot vedea istoricul schimbarilor, ca pot face rollback la o versiune precedenta, in cazul in care o abordare a unei functionalitati a fost gresita si ca la orice schimbare majora a aplicatiei dispun de o versiune stabila si functionala a acesteia.

Timpul alocat dezvoltarii aplicatiei nu a fost unul constant, obiectivele de timp alocat lucrarii de licenta fiind uneori ratate, dar depasite in alte zile. In mare parte, am alocat in medie doua ore pe zi, iar daca nu reuseam sa indeplinesc acest numar de ore, suplimentam in timpul week-end-ului si a vacantelor.

## Structura lucrării

Lucrarea pe care o propun este structurată in doua capitole în care sunt prezentate în detaliu elementele ce stau la baza creării acestei aplicații și interacțiunea dintre ele:

* Capitolul 1 – prezintă pe scurt problemele existente ce au dus la crearea acestei aplicații. Practic aici vom vedea care sunt problemele ce pot fi rezolvate cu ajutorul lucrării propuse de mine.
* Capitolul 2 – acesta va fi cel mai stufos capitol în care se vor prezenta pe rând arhitectura aplicației, tehnologiile utilizate pentru crearea aplicației și scenarii de utilizare a acesteia

# Contribuții

Contribuțiile mele în realizarea acestei lucrări de licență sunt următoarele:

* Crearea unei aplicații mobile pentru platforma Android, aplicatie ce dispune de o interfață grafică prietenoasă si intuitiva, utilizatorul avand, astfel, o experiență cât mai plăcută de-a lungul utilizării ei.
* Oferirea posibilității unui utilizator al aplicației de a-și crea un cont și de a se autentifica cu acesta pentru accesa funcționalitățile aplicației.
* Oferirea posibilitatii unui profesor de a transmite informatii referitoare la cursul, la laboratorarele sau seminarele acestuia, la orarul pe care aceasta disciplina il urmeaza, cat si la punctajele de laborator sau din examen studentilor.
* Oferirea posibilitatii unui profesor de a partaja fisiere relevante disciplinelor sau laboratorelor sale, fisiere vizibile doar studentilor inscrisi la disciplina respectiva.
* Suport pentru notificări. Utilizatorul este notificat intermediul aplicației in momentul modificarii unei discipline la care acesta este inscris, spre exemplu modificarea orarului, a notelor, a fisierelor partajate.
* Ofera posibilitatea profesorilor sau administratorilor aplicatiei de a trimite atat mesaje multicast, precum cele adresate unei grupe sau unui an, cat si mesaje individuale, adresate unui singur utilizator.
* Ofera studentilor posibilitatea de a-si verifica notele, orarul, de a vedea informatii si fisiere referitoare la disciplinele studiate, informatii actualizate in timp real.
* Ofera studentilor un ajutor in alegerea cursurilor optionale pentru semestrul urmator de studiu, pe baza notelor obtinute de acestia in semestrele precedente.

# Capitolul 1: Descrierea problemei

Precum am mentionat in introducere, nu exista niciun motiv pentru a priva mediul universitar de un transfer facil, centralizat si in timp real de informatii, un grad de conectare a participantilor, atat a studentilor cat si a profesorilor, aducand beneficii majore, simplificandu-ne viata de zi cu zi.

Atat informatiile descentralizate ce sunt disponibile individual intr-un numar mare de locatii, lucru care face urmarirea acestora dificila, cat si anunturile ce nu au ajuns in timp util la destinatari reprezinta impedimente atat in viata unui student cat si a unui profesor, generand de multe ori probleme ambelor parti.

O simpla solutie la aceste probleme ar fi centralizarea acestor informatii, astfel acestea devin mai usor de accesat si de urmarit, si transmiterea lor in timp real impreuna cu notificarea persoanelor pentru care acestea prezinta interes, pentru a elimina posibilitatea nereceptionarii acestora in timp util.

# Capitolul 2: Descierea soluției

În acest capitol voi prezenta pe ansamblu arhitectura aplicației, tehnologiile folosite pentru creearea aplicației și scenariile de utilizare ale aplicației.

## 2.1 Arhitectura aplicației

Aplicatie este o aplicatie Android, compusa din doua module, modulul de Front End si modulul de Back End, la acestea adaugandu-se functiile din cadrul Firebase Cloud Functions.

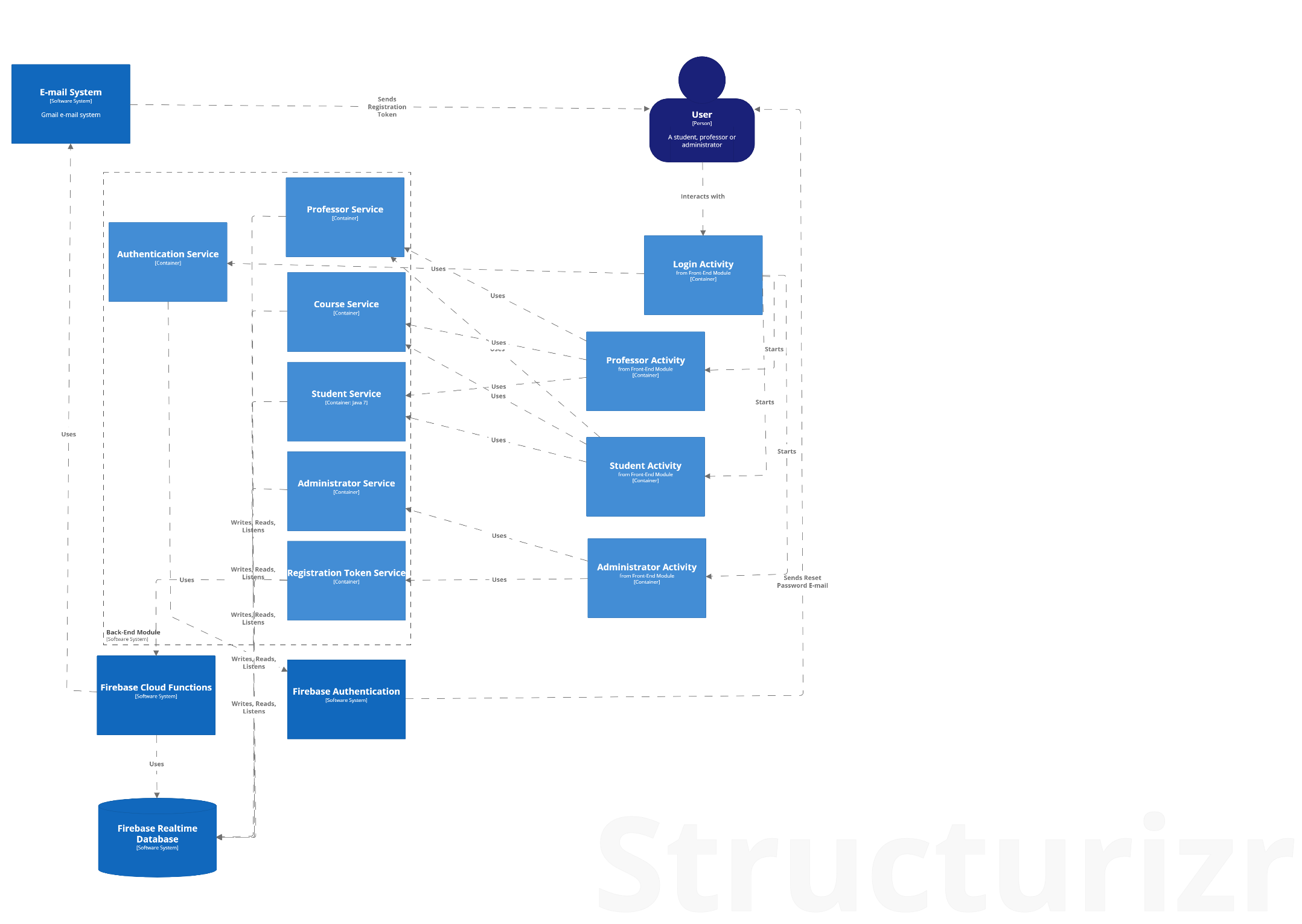
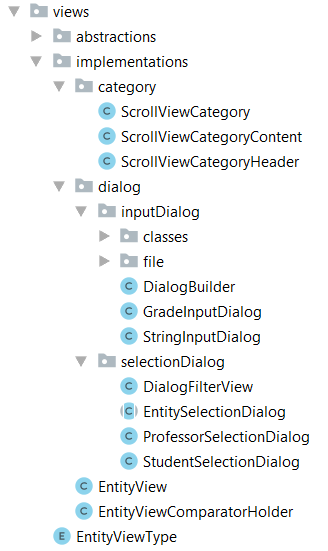


Figura 1 – Arhitectura Aplicatiei

## 2.1.1 Modulul Front End

Modulul de Front End expune utilizatorilor elementele de UI impreuna cu notificarile pe care acestia le vor primi, acesta reprezentand Presentation Layer-ul aplicatiei.

Elementul principal al acestui modul este Activitatea, care in cadrul sistemului Android reprezinta un proces, firul sau principal de executie avand responsabilitatea afisarii si modificarii elementelor grafice disponibile pe ecran. Elementele grafice disponibile in aplicatie sunt descrise atat programatic cat si utilizand fisiere de tip XML.

Fiecare element de UI (view) generat programatic reprezinta o clasa independenta ce respecta Single Responsability Principle, functionalitatile acesteia fiind restranse la propria sa entitate.

Sincronizarea si modificarea datelor in timp real la nivelul acestor componente se realizeaza cu ajutorul tehnologiei de Databinding, disponibila in SDK-ul Androidului, tehnologie ce se bazeaza pe Design Pattern-ul de Observer. Fiecare instanta a unui View reprezinta un observator asupra unui set de date (Observable), iar la modificarea acestor date, fiecare dintre elementele de UI ce le folosesc vor fi actualizate.

Este de mentionat ca majoritatea acestor elemente de UI sunt definite in cadrul fisierelor XML disponibile in aplicatie, acestea definind structura generala a componentei de UI si stilul ei, acest lucru crescand gradul de reutilizare a codului.

Figura 2 – View-uri generate programatic

Fisierele de tip XML ce ajuta la definirea elementelor grafice contin tag-uri si atribute specificie platformei Android (tag-uri precum LinearLayout, RadioButton si atribute precum android:layout\_height, android:onClick, android:background), aceste fisiere fiind parsate la compile time cu scopul validarii acestora, si la run time cu scopul crearii elementulelor grafice specificate in tag-urile XML-ului.

La nivelul acestui modul am implementat arhitectura MVC (Model-View-Controller) in urmatoarea maniera:

* Model – reprezinta seturile de date utilizate de catre Databinding, modele ce reflecta entitatile afisate in View-uri.
* View – element grafic definit in fisierele XML si / sau in propria sa clasa, actiunile si evenimentele efectuate asupra lui fiind delegate Activitatii ce contine respectivul view.
* Controller – este reprezentat de catre Activitate. aceasta reactionand la evenimentele transmise de catre View si efectuand actiunile necesare modificarii interfetei grafice in concordanta cu evenimentul transmis.

## 2.1.2 Modulul Back End

Modulul de Back End contine straturile de Business Logic si Data Access ale aplicatiei, acest modul fiind compus din o multitudine de servicii ce vor fi apelate de modulul prezentat anterior. Aceste servicii scriu si citesc date in/din Firebase Realtime Database, ascultand in acelasi timp schimbarile de date efectuate de catre ceilalti utilizatori, modificand datele expuse in interfata grafica.

Acest modul expune catre modulul de front-end 16 servicii, 13 dintre acestea modificand in Data Access Layer-ul lor cate un „tabel” al bazei de date (fiind o baza de date non-relationala, aceasta nu contine tabele precum o baza de date SQL, concepul de tabel fiind reprezentat de un nod de pe al doilea nivel al arborelui JSON). Fiecare apel catre servicii se realizanda asincron, datorita naturii prin care API-ul Firebase comunica cu aplicatia noastra, si anume prin Listeneri si metode de callback.

#### Business Logic Layer

Precum am mentionat anterior, 13 dintre cele 16 servicii prezinta un Data Access Layer propriu, iar la nivelul B.L.L. acestea asigura validitatea datelor, verificand campurile entitatilor, existenta entitatilor agregate prin chei straine, etc. De asemenea, un serviciu nu acceseaza D.A.L.-ul altui serviciu, intrucat asta ar incalca principiul Single Responsibility si principiul de Separation of Concerns (intrucat fiecare serviciu poate fi vazut drept un modul sau componenta autonoma). In situatia in care entitatile agregate transmise unui serviciu necesita validare la randul lor, acesta va comunica cu serviciile destinate acestora, astfel arhitectura acestui modul poate fi asemanata cu o arhitectura baza te microservicii.

Este de mentionat ca toate metodele expuse de servicii sunt asincrone, din motivul elaborat anterior, insa, neutilizand JDK 8+, interfetele functionale prezente in acesta nu sunt disponibile (interfete precum Future, Function, Comparable, Consumer, etc). In aceasta situatie s-a dovedit a fi utila utilizarea Language Level-ului 8, ce permite definirea de interfete functionale proprii, implementand astfel interfetele functionale necesare facilitarii procesului de dezvoltare (aceleasi interfete enumerate mai sus).

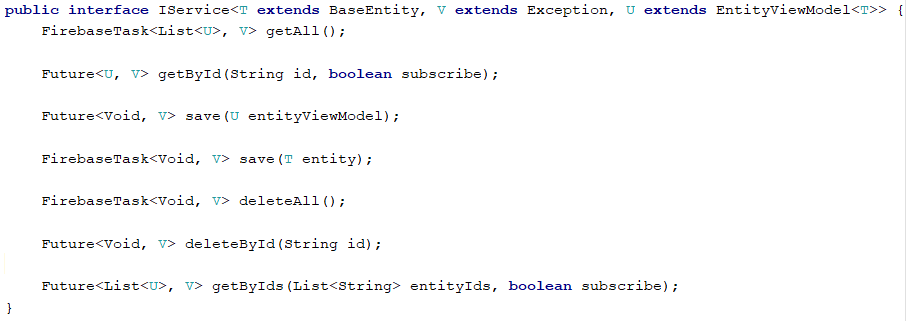


Figura 3 – Interfata implementata de serviciile cu D.A.L.

Din figura 3 se poate observa faptul ca serviciile nu expun entitatile propriu-zise, ci un ViewModel al acestora, pentru a nu dezvalui detaliile de reprezentare a acestora din baza de date, ci a transmite doar date relevante consumatorilor.

De asemenea se poate observa absenta unei metode de update, lucru datorat faptului ca Firebase Realtime Database trateaza metoda de save si cea de update similar.

#### Data Access Layer

Data Access Layer sau D.A.L. este stratul inferior al serviciilor, acesta intervenind intre B.L.L. si baza de date, acesta ocupandu-se de operatiile C.R.U.D. si de setarea Listenerilor asupra entitatilor din baze de date, entitati referentiate prin path-ul absolut ce il au in baza de date (Spre exemplu „/Files/Metadata/{uid}” este path-ul absolut catre metadata cu id-ul „uid” al unui fisier).

Pentru implementarea D.A.L. am optat pentru o abordare generica, intrucat metodele utilizade de acesta sunt similare, indiferent de entitatea asupra carora se realizeaza. In ceea ce priveste partea de Design Patterns, nu am optat pentru Repository, intrucat memorarea datelor in acesta ar fi cauzat probleme de sincronizare, crescand gradul de dificultate al procedurii de actualizare a datelor in timp real. De asemenea, alegerea de a nu implementa pattern-ul de Repository nu are consecinte negative asupra performantei, a latimii de banda utilizate de aplicatie sau a cantitatii de date transmise intre aplicatie si baza de date, deoarece API-ul expus de Firebase utilizeaza caching local, astfel interogarile multiple asupra aceleiasi resurse vor rezulta intr-o singura interogare la baza de date, la primul apel, si citiri din cache la urmatoarele, in conditiile in care resursa in cauza nu se modifica.

In cadrul D.A.L. se seteaza Listeneri asupra entitatilor din baza de date, Listeneri ce vor fi apelati in momentul modificarii resursei respective, asigurand astfel sincronizarea in timp real a datelor. In acest sens s-a utilizat Language Level 8, Listener reprezentand o interfata functionala ce poate fi usor inlocuita cu o expresie lambda, ceea ce a crescut lizibilitatea codului.

## 2.1.3 Firebase Cloud Functions

In cazul functionalitatilor a caror implementare in aplicatia Android ar fi intampinat obstacole, s-a utilizat Firebase Cloud Functions. Acestea reprezinta functii JavaScript carora li s-a facut deploy in Cloud si sunt disponibile fie drept end-point-uri accesibile prin HTTPS, fie drept triggere la scriere pe baza de date.

Principalele functionalitati ce folosesc Firebase Cloud Functions sunt transmiterea de emailuri ce contin tokenul de inregistrare al profesorului sau administratorului si transmiterea de notificari, ce a fost realizata prin intermediul triggerelor.



Figura 4 – Exemplu de functie din cloud, disponibila prin HTTPS

## 2.2 Tehnologii folosite

In elaborarea lucrarii de licenta am utilizat tehnologii moderne, de actualitate, ce au adus valoare proiectului, sporind gradul de utilitate a acestuia si facilitand procesul de dezvoltare prin functionalitatile oferite de acestea. Printre aceste tehnologii se numara:

* Android SDK 17+
* Java JDK 1.7 impreuna cu Language Level 8
* Firebase Authentication
* Firebase Realtime Database
* Firebase Cloud Functions

De asemenea, in cadrul elaborarii acestei lucrari s-au utilizat si urmatoarele instrumente:

* Android Studio 3.5.1
* Android Emulator 29.3.4 cu Android SDK 24
* Visual Studio Code 1.36.1
* Postman
* Git Bash

## 2.2.1 Android SDK 17+

Android SDK 17+ este unul dintre cele mai utilizate SDK-uri din randul dezvoltatorilor, datorita gradului de compatibilitate cu dispozitivele ce se pot gasi deja pe piata, asigurand, conform statisticilor Google o compatibilitate de peste 70%.

In acelasi timp SDK 17+ asigura componente importante precum RadioGroup, Constraint Layout, o gama variata de obiecte de tip Listener si multe alte elemente de UI.

Fiind o versiune relativ veche de Android, acesta foloseste, din pacate Java 7 cu Language Level 8, acesta fiind motivul alegerii urmatoarelor tehnologii.

## 2.2.2 Java JDK 1.7 impreuna cu Language Level 8

Din motivul elaborat mai sus, si anume acela de a pastra compatibilitatea cu cat mai multe dispozitive Android, am ales sa implementez aplicatia cu ajutorul versiunii 1.7 de Java, impreuna cu Language Level 8.

Acest nivel de limbaj asigura posibilitatea utilizarii de lambda expressions, definirea interfetelor functionale proprii, interfete precum Function<T,V>, Predicate<T>, Consumer<T>, interfete ce au fost implementate in cadrul aplicatiei, intrucat acestea faciliteaza procesul de development, oferind o paleta mai larga de situatii in care se pot utiliza Lambda Expressions. Utilizarea acestora imbunatateste drastic gradul de lizibilitate a codului, multe obiecte disponibile in SDK 17+, in special cele de tip Listener, putand fi inlocuite prin lambda expressions, reducand numarul de clase si clase anonime ale aplicatiei.

## 2.2.3 Firebase Authentication

Dorind a elabora o aplicatie server-less, pentru gestionarea conturilor, a utilizatorilor si a procesului de login, de inregistrare si de resetare a parolei, am optat pentru integrarea API-ului de Firebase in cadrul aplicatiei.

Acesta asigura confidentialitatea datelor conturilor de utilizator, unicitatea si validarea emailurilor, procesul de resetare a parolei prin email, toate acestea reducand gradul de complexitate a aplicatiei in aceste functionalitati.

## 2.2.3 Firebase Realtime Database

Dupa lungi considerente, din dorinta transmiterii datelor in timp real, mentinand toti utilizatorii conectati in permanenta si optand pentru o aplicatie server-less, am ales ca baza de date Firebase Realtime Database.

Aceasta este o baza de date No-SQL, datele fiind memorate sub forma unui JSON, oferind posibilitatea de a actualiza datele aproape instant de pe toate dispozitivele conectate la aceasta.

O alta motivatie in aceasta alegere sunt regulile de securitate expuse de catre Firebase Realtime Database, putand seta reguli de citire si scriere asupra fiecarui nod din arborele JSON, asigurand astfel confidentialitatea si integritatea datelor aplicatiei.

Structura bazei de date:

## 2.2.4 Firebase Cloud Functions

Un alt produs folosit din cadrul pachetului oferit de catre Firebase este Firebase Cloud Functions. Acestea expun functii scrise in Javascript, disponibile aplicatiei sub forma unor endpoint-uri accesate prin HTTPS de catre API-ul Firebase integrat in cadrul aplicatiei Android.

O alta folosinta a Firebase Cloud Functions este aplicarea de triggere asupra nodurilor din arborele JSON, putand executa functii customizate la schimbarea datelor din baza de date. Acest tip de functionalitate asigura transmiterea notificarilor catre dispozitivelor utilizatorilor. Un exemplu pentru aceasta functionalitate

## 2.3 Scenarii de utilizare a aplicației

# Concluziile lucrării

# Bibliografie

# Anexe