

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Intelligent CV generator**

propusă de

**Petru Tanasă**

**Sesiunea: iulie, 2019**

Coordonator științific

**Prof. Colab. Florin Olariu**

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**

# **Intelligent CV generator**

**Petru Tanasă**

**Sesiunea: iulie, 2019**

Coordonator științific

**Prof. Colab. Florin Olariu**

Avizat,  
Îndrumător lucrare de licență,  
Prof. Colab. Florin Olariu.

Data: ..... Semnătura: .....

### **Declarație privind originalitatea conținutului lucrării de licență**

Subsemnatul **Tanasă Petru** domiciliat în **România, jud. Neamț, mun. Piatra Neamț, str. Dărmănești, nr. 48, bl. K5, et. 1, ap. 8**, născut la data de **04 iunie 1996**, identificat prin CNP **1960604270028**, absolvent al **Universității "Alexandru-Ioan Cuza" din Iași, Facultatea de Informatică**, specializarea **informatică**, promoția 2018, 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 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Intelligent CV generator** elaborată sub îndrumarea domnului **Prof. Colab. Florin Olariu**, pe care urmează să o susțin 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 ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă 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.

Data: .....

Semnătura: .....

### **Declarație de consimțământ**

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Intelligent CV generator**, 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 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.

Absolvent **Petru Tanasă**

Data: .....

Semnătura: .....

# Cuprins

<b>Introducere</b>	<b>2</b>
<b>1 Problema și soluția</b>	<b>3</b>
1.1 Scurt istoric . . . . .	3
1.2 Descrierea problemei . . . . .	3
1.3 Descrierea soluției . . . . .	4
1.4 Scenarii posibile . . . . .	4
<b>2 Despre aplicația Intelligent CV Generator</b>	<b>6</b>
2.1 Ce este un CV? . . . . .	6
2.2 Ce este un CV Inteligent? . . . . .	6
2.3 Ce aplicații similare mai există? . . . . .	6
2.4 Ce își propune această aplicație? . . . . .	7
2.5 Cu ce vine nou această aplicație? . . . . .	7
2.6 Ce poate fi dezvoltat în continuare? . . . . .	8
<b>3 Salvarea datelor</b>	<b>9</b>
3.1 Fișierele .xml . . . . .	10
3.2 Fișierele .pdf . . . . .	12
3.3 Template-uri . . . . .	12
<b>4 Tehnologii folosite</b>	<b>13</b>
4.1 Fragments . . . . .	13
4.2 ArrayAdapter . . . . .	16
4.3 API: iText PDF . . . . .	17
4.4 API: Speech-To-Text . . . . .	19
4.5 Speech Recognizer . . . . .	21

<b>5</b>	<b>Structura aplicatiei</b>	<b>22</b>
5.1	MainActivity . . . . .	22
5.2	ActivityCreateCV . . . . .	23
5.2.1	Contact Fragment . . . . .	24
5.2.2	Skill Fragment . . . . .	25
5.2.3	Education Fragment . . . . .	27
5.2.4	Experience Fragment . . . . .	28
5.2.5	Projects Fragment . . . . .	30
5.2.6	Communication Fragment . . . . .	31
5.3	ActivityCVList . . . . .	33
	<b>Concluzii</b>	<b>34</b>
	<b>Bibliografie</b>	<b>35</b>

# Introducere

Primul pas în obținerea job-ului mult visat în 2019 reprezintă un CV. CV-ul este un fel de bibliografie a candidatului, și cuprinde informații despre acesta, date de contact, informații despre abilitățile utilizatorului, informații despre educație, experiențe, proiecte personale, și chiar ce limbi străine cunoaște.

Aplicația **Intelligent CV Generator** este o aplicație pentru platforma Android, ce vine în ajutorul utilizatorului în generarea de CV-uri într-un mod inteligent, modern și ușor. Pentru a crea un CV, utilizatorul trebuie să completeze un formular cu informații ce vor fi așezate într-un fișier PDF. Datele pot fi introduse prin metoda clasică, 'de la tastatură', sau printr-o metodă modernă, și anume vocal. Datele introduse pot fi modificate sau șterse, în funcție de preferințele utilizatorului. După completarea formularului, datele trebuie salvate prin apăsarea unui buton. Pentru a vizualiza PDF-ul, utilizatorul merge în meniul CV List, și alege CV-ul ce vrea să îl vadă.

Pentru această aplicație, am folosit Android Studio și emulatorul pus la dispoziție de Android Studio pentru a o testa. Pe parcursul dezvoltării aplicației Intelligent CV Generator, am utilizat mai multe API-uri, printre care se numără iText PDF, Speech-To-Text și Text-To-Speech. Pentru formularul cu datele CV-ului, am lucrat cu Android Fragments și cu ArrayAdapter pentru o interfață mai prietenoasă.

# Capitolul 1

## Problema și soluția

### 1.1 Scurt istoric

23 Septembrie 2008 – o zi istorică în domeniul tehnologiei: Google lansează prima versiune de Android. Pe atunci, sistemul de operare permitea descărcarea și instalarea de aplicații dintr-un „market” virtual(Android Market). Accesul la servicii de e-mail precum POP3, IMAP4, SMTP se numărau printre avantajele Android’ului, alături de opțiuni precum sincronizarea contului google între telefon și alt dispozitiv (alt telefon/PC/...), și multe alte feature-uri. Din acea zi, sistemul de operare Android s-a dezvoltat și a devenit foarte popular, fiind prezent în 2018 în peste [2]70% din telefoanele mobile din toată lumea.

În ultimii ani, telefoanele s-au dezvoltat foarte puternic din punct de vedere hardware și software, permițând unui utilizator să-și desfășoare aproape orice activitate de pe laptop/pc pe telefon. De la scrierea unui simplu mesaj s-a ajuns la navigarea pe internet, chiar și la gaming-ul pe telefon.

### 1.2 Descrierea problemei

Puterea hardware-ului și software-ului din telefoanele mobile, alături de multitudinea de aplicații ce pot fi dezvoltate folosind toată această tehnologie ar trebui exploatată și utilizată la maxim, deoarece se pot face lucruri uimitoare, poate chiar ar putea salva vieți. Dar până la salvarea de vieți, cu această aplicație ne propunem să rezolvăm problema pe care o întâmpină un proaspăt absolvent de facultate, sau o persoană mai nouă în domeniul tehnologiei care ar dori să se angajeze, iar angajatorul i-a cerut un



CV. Utilizatorul, neavând prea multe cunoștințe despre ce ar trebui să conțină un CV, cum ar trebui structurat, sau cum se crează un document word sau fișier PDF, ar putea folosi această aplicație pentru a-și genera propriul CV.

## 1.3 Descrierea soluției

Această aplicație își propune să rezolve această problemă, și vine în ajutorul utilizatorilor cu o tehnologie modernă și ușor de rezolvat. Până la descrierea tehnologiei utilizate în aplicație, putem privi puțin în trecut, și observăm că marii producători de software, au dezvoltat din ce în ce mai multe produse care folosesc comenzi vocale. Siri de la Apple, Cortana de la Microsoft, Alexa de la Amazon sunt doar câteva exemple de aplicații care folosesc tehnologia Text-to-Speech și Speech-to-Text. Fiind o tehnologie care nu implică foarte multe cunoștințe în domeniul tehnologiei pentru un utilizator, am dezvoltat această aplicație care se folosește de Speech-to-Text pentru a crea CV-uri.

## 1.4 Scenarii posibile

Initial utilizatorul are doua optiuni: de a crea un CV nou, și de a vizualiza lista cu CV-uri, implicând și generarea fișierului .PDF.

Pentru a crea un CV nou, utilizatorul introduce un nume pentru CV, și apasă butonul **Create CV**. După apăsarea butonului, interfața aplicației se schimbă, iar utilizatorul poate naviga prin formular apăsând pe diferite obiecte, sau introducând comenzi vocale. La apăsarea butonului **ADD BY TEXT**, utilizatorul trebuie să completeze date în fereastra ce apare pe ecran, pentru a adăuga un nou obiect (Skill / Education / Experience / Projects / Communication, în funcție de fereastra în care se afla). De asemenea, în cazul unei greșeli, utilizatorul poate modifica / șterge datele dintr-o listă, apăsând pe elementul ce se vrea modificat.

Pentru a vizualiza lista de CV-uri, utilizatorul trebuie să apese butonul **CV List** din meniul principal. La apăsarea butonului, apare o listă cu CV-urile create (numele și data ultimei modificări). Folosind meniul de tip dropdown de sus, utilizatorul poate ordona lista cu CV-uri după nume și după data ultimei modificări; ordinea poate fi ascendentă, sau descendentă. La apăsarea scurtă pe un element din listă, se generează fișierul PDF, iar CV-ul este gata de utilizare. La o apăsare mai lungă pe un element din listă, utilizatorului i se deschide un menu de unde poate genera CV-uri folosind

template-ul 1, template-ul 2, sau poate șterge CV-ul. După ștergerea unui CV, lista este reactualizată folosind ultimul filtru ales.

Mai jos am adăugat și o diagrama UseCase:

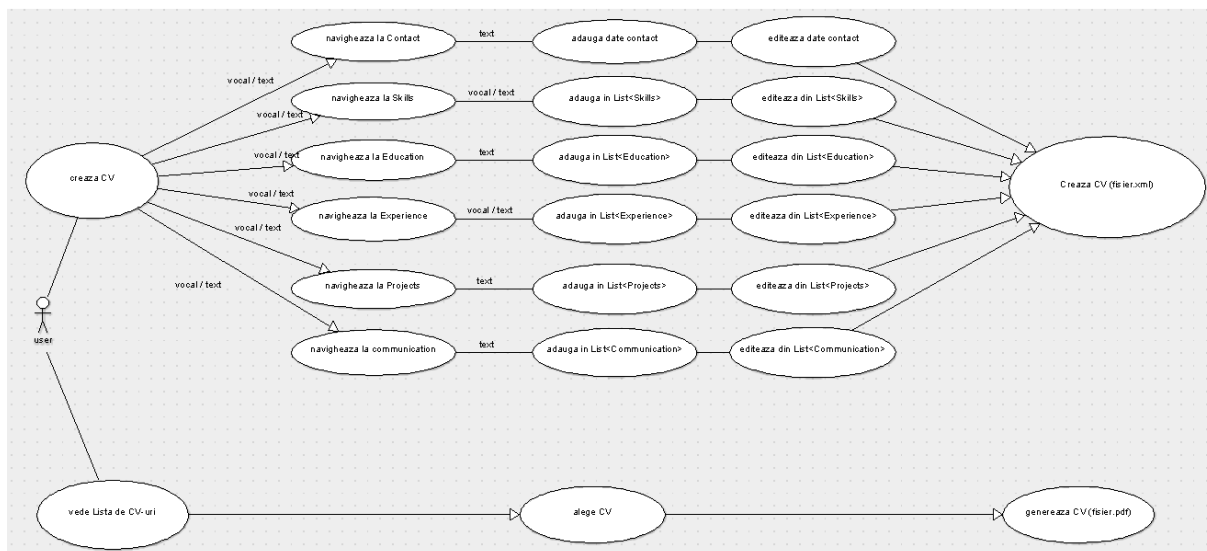


Figura 1.1: Diagrama Use Case

## Capitolul 2

# Despre aplicația Intelligent CV Generator

### 2.1 Ce este un CV?

Conform Dicționarului Explicativ Român, termenul de CV reprezintă abrevierea pentru Curriculum Vitae.

[1] *"CURRICULUM VITAE loc. Scurtă autobiografie (scrisă) care cuprinde date referitoare la studii, pregătirea profesională, carieră și situația familială;"*

### 2.2 Ce este un CV Intelligent?

Un CV inteligent, reprezintă un CV modern, curat, ușor de citit, ce cuprind doar informații strict relevante pentru jobul la care s-a aplicat. Un CV inteligent este creat într-un mod inteligent, folosind tehnici moderne și ușor de folosit.

### 2.3 Ce aplicații similare mai există?

Deoarece aplicația prezentată în această lucrare se bazează pe platforma Android, vom analiza top 3 aplicații similare de pe aceeași platforma, din PlayStore:

- Resume Builder App Free CV maker CV - Prima, și probabil cea mai modernă aplicație din PlayStore; este o aplicație cu un design plăcut, are la bază foarte multe template-uri pentru CV, interacțiunea cu meniul și opțiunile aplicației se

fac în mod clasic, prin folosirea butoanelor din aplicație. Datele se introduc tot într-un mod clasic, prin introducerea textului de la tastatură;

- Smart Resume Builder - ce de a doua aplicație din PlayStore, permite, la fel ca aplicația noastră, generarea unui CV. Template-urile au un aspect plăcut, dar navigarea prin aplicație și introducerea datelor se fac în același mod ca în aplicația descrisă mai sus;
- CV Builder for Smart Resumes - ultima aplicație din topul nostru, este foarte asemănătoare cu cea prezentată mai sus, și folosește aceleași metode de a interacționa cu aplicația.

Din câte am putut observa, primele 3 aplicații din PlayStore, folosesc metode vechi de a interacționa cu aplicația, spre deosebire de aplicația Intelligent CV Generator, care folosește metode moderne de a interacționa cu aplicația.

## **2.4 Ce își propune această aplicație?**

Această aplicație vine în ajutorul proaspăt absolvenților, sau în ajutorul oricărei persoane ce urmează să aplice la un job. Aplicația "Intelligent CV Generator" ajută utilizatorul să își creeze un CV curat, modern, și eficient din punct de vedere al informațiilor ce se vor afla în el. Interfața aplicației este foarte simplă și intuitivă, astfel încât orice persoană o poate folosi cu ușurință. Design-ul CV-ului este simplu și foarte bine structurat pentru a fi ușor de citit și înțeles. Aplicația oferă mai multe variante de a introduce date în aplicație pentru a fi mai comod de folosit.

## **2.5 Cu ce vine nou această aplicație?**

Pentru introducerea datelor în CV, aplicația dispune de recunoaștere vocală, astfel utilizatorul poate vorbi cu aplicația, iar aceasta transformă vorbitul în text. Mai mult de atât, interacțiunea cu aplicația se poate face și vocal, acceptând comenzi vocale pentru a naviga rapid prin meniuri.

## 2.6 Ce poate fi dezvoltat în continuare?

Un viitor feature ce ar atrage utilizatorii să folosească aplicația, ar fi posibilitatea de a alege din mai multe template-uri pentru .pdf-uri. Multitudinea de template-uri, ar însemna și posibilitatea de a încărca poze în pdf, ceea ce ar putea fi un alt update interesant pentru clienții aplicației. A înregistra vocal tot conținutul CV-ului ar fi un feature bine primit de utilizatori și interesant de dezvoltat în următoarele versiuni, și ar face aplicația foarte populară.

# Capitolul 3

## Salvarea datelor

În funcție de nevoile aplicației, pe platforma android se diferențiază mai multe tipuri de “data storage”:

- Internal Data Storage
- External Data Storage
- Shared Preferences
- Databases

Internal Data Storage este folosit pentru a salva date la care utilizatorul nu are acces în mod direct, adică date private. Informațiile salvate aici, vor fi șterse odată cu deinstalarea aplicației.

External Data Storage este folosit pentru a salva date la care utilizatorul are acces în mod direct, adică informația este publică, deci datele salvate aici vor putea fi accesate și din alte aplicații. Aceste informații nu vor fi șterse în cazul deinstalării aplicației.

Shared Preferences - informațiile salvate aici vor fi de tipul chaie-valoare. De regulă este indicată salvarea datelor pentru setările aplicației(exemple: limba folosită în aplicație EN/RO, tema aplicației, ...). În mod direct, utilizatorul nu poate interacționa cu aceste date.

Databases - informațiile sunt structurate într-o bază de date și nu pot fi accesate în mod direct de către utilizator.

Aplicația noastră folosește Internal Data Storage pentru a salva fișiere .xml, adică informațiile din CV, și External Data Storage pentru a salva fișiere .pdf, adică CV-ul propriu zis.

## 3.1 Fișierele .xml

Fișierele .xml conțin informațiile ce vor fi adăugate în CV. Aceste fișiere se generează la apăsarea butonului **Create CV**. Un exemplu de fișier, poate fi PeTa\\_CV\.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<CV name="PeTa\_CV\.xml">

  <data dataType="contact">
    <contactData dataContactType="name">
      <first_name>Petru</first_name>
      <middle_name/>
      <last_name>Tanasa</last_name>
    </contactData>

    <contactData dataContactType="address">
      <country>Romania</country>
      <city>Iasi</city>
      <cod_postal>700668</cod_postal>
    </contactData>

    <contactData dataContactType="phoneNumber">
      <phoneNumber>0758990801</phoneNumber>
    </contactData>

    <contactData dataContactType="email">
      <email>tanasapetrut@gmail.com</email>
    </contactData>
  </data>

  <data dataType="skills">
    <skillData id="0">
      <name>Software Testing</name>
    </skillData>
  </data>
```

```

<data dataType="education">
  <educationData id="0">
    <type>Facultate</type>
    <name>Facultatea de Informatica Iasi</name>
    <profile>Informatica</profile>
    <start_date>2015</start_date>
    <end_date>2018</end_date>
  </educationData>
</data>

<data dataType="experience">
  <experienceData id="0">
    <name>Continental</name>
    <position>Software Testing</position>
    <start_date>04.12.2017</start_date>
    <end_date>Present</end_date>
  </experienceData>
</data>

<data dataType="projects">
  <projectData id="0">
    <name>Online Horse Bet</name>
    <description>Website for betting.</description>
  </projectData>
</data>

<data dataType="communication">
  <communicationData id="0">
    <name>English</name>
    <level>B1</level>
  </communicationData>
</data>

</CV>

```

Listing 1: Fromat fișier .xml



## 3.2 Fișierele .pdf

Fișierele .pdf conțin designul CV-ului completat cu datele din fișierul .xml. Acest fișier se generează la selectarea unui CV din lista CV-urilor existente. Formatul fișierului poate diferi, în funcție de tema aleasă de utilizator.

## 3.3 Template-uri

În momentul de față, aplicația dispune de două template-uri. Aceasta poate fi ales de către utilizator, din opțiunile aplicației.

Template 1 (Left):

- firstName middleName secondName
- 700673, Iasi, Romania
- Telefon: 0758990801
- E-mail: tanasapetrut@hotmail.com
- Aptitudini
- Experienta
- Studii

Template 2 (Right):

- fName mName lName
- address
- e-mail
- phone
- Skills
- Education
- Experience

Figura 3.1: Template 1 și 2

# Capitolul 4

## Tehnologii folosite

În acest capitol vom detalia tehnologiile folosite în dezvoltarea aplicației.

- Fragments
- ArrayAdapter
- API: iText PDF
- API: Speech-To-Text
- Speech Recognizer

### 4.1 Fragments

Un fragment este o clasă reutilizabilă care implementează o parte a unei activități. Un fragment reprezintă de obicei interfață pentru utilizator. Acesta nu poate rula independent de activitate, deci trebuie inclus într-o activitate.

Fragmentul se aseamănă foarte mult cu o activitate, deoarece ambele reprezintă o combinație dintre fișierul .xml și clasa .java, iar ambele au un ciclu de viață asemănător.

Pentru a înțelege mai bine diferența dintre fragmente și activități, putem spune că activitățile sunt folosite pentru navigarea prin aplicație, iar fragmentele se folosesc pentru view-uri și logică.

În continuare vom exemplifica modul în care am folosit Fragmentele în aplicația noastră:

```
public class Fragment_Skills extends Fragment {

    // avem nevoie de un constructor public, gol.
    public Fragment_Skills() {
    }

    // Suprascriem clasa onCreate()
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    // Suprascriem onCreateView(), si returnam view-ul curent
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent,
Bundle savedInstanceState) {
        View rootView = inflater.inflate(
R.layout.fragment_fragment__skills, container, false);
        return rootView;
    }
}
```

Listing 2: Implementare Fragments

Fragmentele au fost declarate și instanțiate în clasa ActivityCreateCV:

```
public class ActivityCreateCV extends AppCompatActivity {
    private ViewPager viewPager;
    private Fragment_Contact fragContact;
    private Fragment_Skills fragSkill;
    private Fragment_Education fragEducation;
    private Fragment_Experience fragExperience;
    private Fragment_Projects fragProject;
    private Fragment_Communication fragCommunication;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_cv);
        // ...
        viewPager = (ViewPager) findViewById(R.id.viewpager);
        setupViewPager(viewPager);
    }
    // ...
    private void setupViewPager(ViewPager viewPager) {
        fragContact = new Fragment_Contact();
        fragSkill = new Fragment_Skills();
        fragEducation = new Fragment_Education();
        fragExperience = new Fragment_Experience();
        fragProject = new Fragment_Projects();
        fragCommunication = new Fragment_Communication();

        adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFragment(fragContact, "Contact");
        adapter.addFragment(fragSkill, "Skills");
        adapter.addFragment(fragEducation, "Education");
        adapter.addFragment(fragExperience, "Experience");
        adapter.addFragment(fragProject, "Projects");
        adapter.addFragment(fragCommunication, "Communication");
        viewPager.setAdapter(adapter);
    }
}
```

Listing 3: Utilizare Fragments

## 4.2 ArrayAdapter

ArrayAdapter-ul este folosit, în general, atunci când avem nevoie de un ListView scrolabil. În cazul nostru, în fragmentele Skills, Education, Experience, Projects, Communication avem câte o listă scrolabilă, deci vom implementa câte un ArrayAdapter pentru a face legătura între ListView și ArrayList.

Mai departe vom încerca să arătăm cum am folosit în aplicația noastră acest ArrayAdapter:

```
public class SkillListAdapter extends ArrayAdapter<Skill> {
    private ArrayList<Skill> skillsAdapter;
    public SkillListAdapter(Context context, ArrayList<Skill> _skills) {
        super(context, 0, _skills);
        this.skillsAdapter = _skills;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent){

        // Get the view
        View view = convertView;
        if(view == null) {
            view = LayoutInflater.from(context).inflate(
R.layout.list_skills, parent, false);
        }

        if(skillsAdapter.size() == 0) return view;
        Skill skill = skillsAdapter.get(position); //Get the Skill Objects
        TextView line1 = view.findViewById(R.id.tv_SkillsList_line1);
        line1.setText(skill.getNum());

        return view;
    }
}
```

Listing 4: Implementare ArrayAdapter

Mai jos avem un exemplu despre cum este folosit acest SkillListAdapter, care extinde clasa ArrayAdapter. Următoarele linii de cod sunt din `Fragment\_Skill.java`:

```
// ...  
adapterSkills = new SkillListAdapter(getActivity().  
getApplicationContext(), skills);  
  
Skill newSkill = new Skill("My Temp Skill");  
adapterSkills.add(newSkill);  
adapterSkills.notifyDataSetChanged();  
  
// ...
```

Listing 5: `Fragment_Skill.java`

## 4.3 API: iText PDF

iText este o librărie PDF pentru Java ce permite dezvoltarea de programe Java pentru crearea și manipularea fișierelor .pdf. În continuare vom descrie cum am folosit iText în aplicația **Intelligent CV generator**. Pentru o modularizare mai bună a codului, am creat câteva metode eficiente pentru a desena linii și a scrie text. Pentru a putea crea fișiere, aplicația are nevoie de permisiuni asupra `DataStorage`-ului. Pentru a-i da aplicației această permisiune, se adaugă în `AndroidManifest.xml` următoarele linii: `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>` și `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>`

```

Document document = new Document();

File customPrivateDir = context.getExternalFilesDir("CV2");
File mPdfFile = new File(customPrivateDir, cvName + ".pdf");
PdfWriter writer = PdfWriter.getInstance(document,
new FileOutputStream(myPdfFile));

document.open();

```

Listing 6: Creare fișier PDF

Pentru a desena linii in fișierul PDF, am folosit funcția `drawLine(PdfWriter, float, float, float, float, BaseColor)`:

```

private void drawLine(PdfWriter writer, float width_1, float height_1,
float width_2, float height_2, BaseColor color) {
    PdfContentByte canvas = writer.getDirectContent();
    Rectangle rect = new Rectangle(width_1, height_1,
width_2, height_2);
    rect.setBorder(Rectangle.BOX);
    rect.setBorderWidth(0);
    rect.setBackgroundColor(color);

    canvas.rectangle(rect);
}

```

Listing 7: Implementare metoda `drawLine()`

Pentru a scrie text in fișierul PDF, am folosit funcția `writeLongText (PdfWriter, String, float, float, int, float, float, float, int)`:

```
private void writeLongText (PdfWriter writer, String text,
float paddingLeft, float paddingRight, int fontSize, float spaceAfter,
float spaceBefore, float firstLineIndent, int align) {
    writer.setSpaceCharRatio (PdfWriter.NO_SPACE_CHAR_RATIO);
    Paragraph paragraph = new Paragraph();
    paragraph.setIndentationLeft (paddingLeft);
    paragraph.setIndentationRight (paddingRight);
    paragraph.getFont ().setSize (fontSize);
    paragraph.setAlignment (align);
    paragraph.setFirstLineIndent (firstLineIndent);
    paragraph.setSpacingAfter (spaceAfter);
    paragraph.setSpacingBefore (spaceBefore);
    paragraph.add (text);

    document.add (paragraph);
}
```

Listing 8: Implementare metoda `writeLongText`

## 4.4 API: Speech-To-Text

Acest API l-am folosit pentru a lasă aplicația să înregistreze ce spune vocal utilizatorul, apoi să convertească înregistrarea în text, astfel încât să obținem date pentru informațiile din CV. Folosind acest API, aplicația are nevoie de câteva perisiuni mai speciale pentru a putea funcționa. Este vorba despre permisiunea de a înregistra vocal, și de a se conecta la internet pentru a converti înregistrarea în text. Acest permisiuni trebuie adăugate în `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Pentru o organizare mai bună a codului, am creat o clasă `SpeechToText`, în care se folosește un obiect de tip `SpeechRecognizer` (pentru a asculta), și un `Intent` (pentru a primit textul).



```

private void createInstanceForSpeechRecognizer(Context context) {
    mSpeechRecognizer = SpeechRecognizer.createSpeechRecognizer(
context);

    mSpeechRecognizerIntent = new Intent(
RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    mSpeechRecognizerIntent.putExtra(
RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

    mSpeechRecognizerIntent.putExtra(
RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());

    mSpeechRecognizer.setRecognitionListener(new RecognitionListener() {
        // ...
    });
}

```

Listing 9: Implementare SpeechRecognizer

Atunci când vrem ca aplicația să înceapă să asculte 'date', vom folosi metoda `startListening(SpeechRecognizer, Intent)`, și `stopListening(SpeechRecognizer)` pentru a ne opri din înregistrat:

```

public void startListening(SpeechRecognizer mSpeechRecognizer,
Intent mSpeechRecognizerIntent) {
    mSpeechRecognizer.startListening(mSpeechRecognizerIntent);
}

public String stopListening(SpeechRecognizer mSpeechRecognizer) {
    mSpeechRecognizer.stopListening();
    return getText();
}

```

Listing 10: Implementare ascultare vocală

Atunci când aplicația ascultă date, se apelează diferite metode, pe care le-am su-

prascris în event-ul `setRecognitionListener()`:

- `onReadyForSpeech()` - `SpeechRecognizer`-ul este gata să asculte;
- `onBeginningOfSpeech()` - `SpeechRecognizer`-ul a început să asculte;
- `onRmsChanged()` - intensitatea vocii a fost schimbată;
- `onBufferReceived()` - primește un buffer;
- `onEndOfSpeech()` - `SpeechRecognizer`-ul s-a oprit din a asculta;
- `onError()` - `SpeechRecognizer`-ul a detectat o eroare (`ERROR_NETWORK_TIMEOUT`, `ERROR_AUDIO`, `ERROR_SERVER`, `ERROR_CLIENT`, `ERROR_SPEECH_TIMEOUT`, `ERROR_NO_MATCH`, `ERROR_RECOGNIZER_BUSY`, `ERROR_INSUFFICIENT_PERMISSION`);
- `onResults()` - `SpeechRecognizer`-ul a primit rezultatele înregistrării;
- `onPartialResults()` - s-au primit o parte din rezultate;

## 4.5 Speech Recognizer

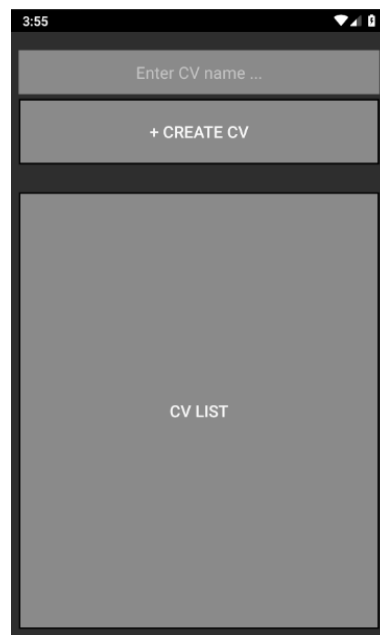
`SpeechRecognizer`-ul este tot un `Speech-To-Text`, doar că aici, aplicația procesează textul primit, și în funcție de acesta, execută diferite comenzi. La fel ca în capitolul de mai sus, am suprascries metoda `onResult()`, unde am apelat metoda `processResult(List<String>)`. Aplicația va parsea datele din lista cu string-uri, și va decide ce să facă în continuare. Spre exemplu, dacă în listă se află și stringul "add skill", aplicația va vorbi ("Please insert a new skill"), va deschide fragmentul `Skills`, și va afișa dialogul pentru a introduce un nou Skill.

# Capitolul 5

## Structura aplicatiei

Aplicația dispune de 3 activități (MainActivity, ActivityCreateCV, ActivityCV-List), fiecare dintre ele având un scop precis.

### 5.1 MainActivity



Oferă un menu principal, prin care utilizatorul poate naviga simplu, rapid și intuitiv către celelalte activități (creare CV, vizualizare CV-uri).

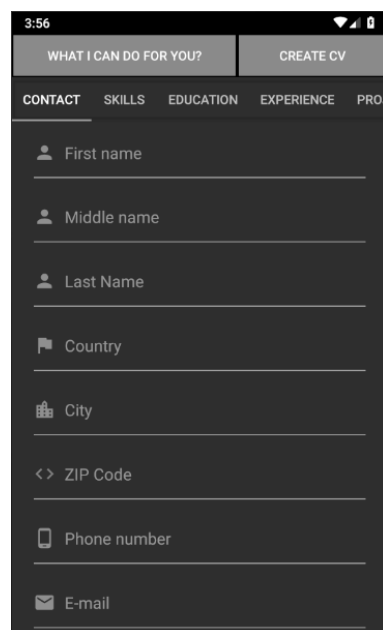
Meniul principal conține 2 Butoane și un TextBox, astfel:

- btn\_MainActivity\_createCV - la apăsarea butonului, se afișează activitatea cu formularul pentru a introduce datele din CV;

- `btn_MainActivity_CVList` - la apăsarea butonului, se afișează activitatea cu toate CV-urile create până în acel moment. Prin 'CV-urile create' ne referim la fișierele .xml.
- `et_CVName` - textbox în care se introduce numele CV-ului. Prin 'numele CV-ului' ne referim la numele fișierului .xml.

Dacă textbox-ul este completat, la apăsarea butonului `btn_MainActivity_createCV`, se transmite către activitatea `ActivityCreateCV` un obiect de tip `String` în care se reține conținutul textbox-ului pentru a da un nume fișierului .xml. În cazul în care textbox-ul nu este completat, numele fișierului va fi `default.xml`.

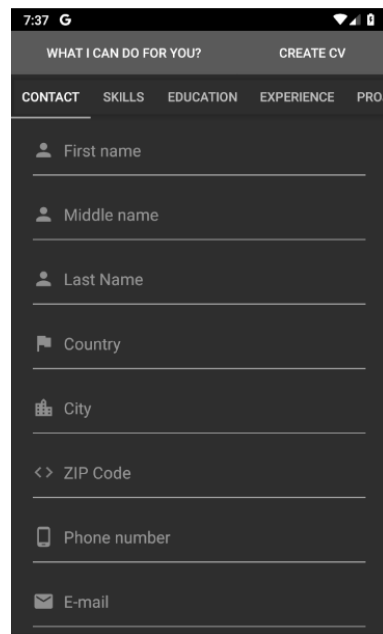
## 5.2 ActivityCreateCV



Pune la dispoziție un meniu complex prin care se completează datele CV-ului, un buton pentru salvarea datelor și un buton pentru a introduce comenzi vocale(exemple: "Open Skill" / "Add education"). Meniul este format din șase fragmente, fiecare ocupându-se de câte un capitol din CV (Contact, Skills, Education, Experience, Projects, Communication).

Pentru fiecare fragment a fost suprascrisă clasa `onStop()` pentru a salva datele introduse în CV. Prin urmare, atunci când utilizatorul schimbă fragmentul, datele din acel fragment sunt salvate în CV.

## 5.2.1 Contact Fragment



7:37 G

WHAT I CAN DO FOR YOU? CREATE CV

CONTACT SKILLS EDUCATION EXPERIENCE PROJ

First name

Middle name

Last Name

Country

City

<> ZIP Code

Phone number

E-mail

Conține o multitudine de `textBox`-uri pentru diferite informații de contact ale utilizatorului. La fiecare `textBox` se află iconiță, dar și un 'Hint' prin care se sugerează ce informație ar trebui completată în câmpul respectiv.

- `et_contact_firstname` - (String, litere, maxim 24 caractere)
- `et_contact_middlename` - (String, litere, maxim 24 caractere)
- `et_contact_lastname` - (String, litere, maxim 24 caractere)
- `et_contact_country` - (String, litere, maxim 30 caractere)
- `et_contact_city` - (String, litere, maxim 30 caractere)
- `et_contact_zip` - (String, litere și cifre, maxim 16 caractere)
- `et_contact_phoneNumber` - (Număr, maxim 14 caractere)
- `et_contact_email` - (String, email, maxim 50 caractere)

Suprascrierea metodei `onStop()` din `Fragment\_Contact.java`, s-a facut în felul următor:

```

@Override
public void onStop() {
    super.onStop();
    cv.setFirstName(et_cv_firstname.getText().toString());
    cv.setMiddleName(et_cv_middlename.getText().toString());
    cv.setLastName(et_cv_lastname.getText().toString());

    cv.setCountry(et_cv_country.getText().toString());
    cv.setCity(et_cv_city.getText().toString());

    cv.setCod_postal(et_cv_zip.getText().toString());
    cv.setPhoneNumber(et_cv_phone.getText().toString());
    cv.setEmail(et_cv_email.getText().toString());
}

```

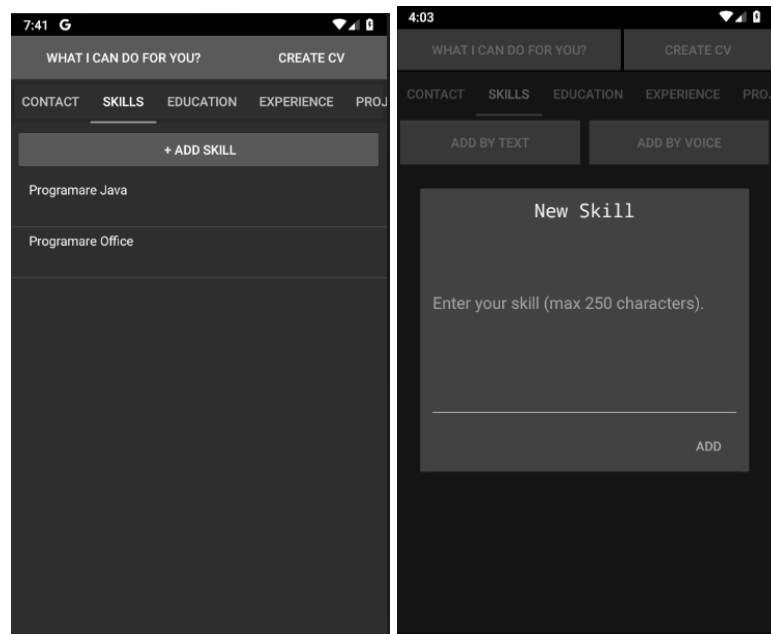


Figura 5.1: *Fragment Skill (stânga) și dialog Add Skill (dreapta)*

### 5.2.2 Skill Fragment

Conține două butoane pentru a adăuga un nou Skill (*btnAddSkillByText* și *btnAddSkillByVoice*), și o listă (*lv\_currentSkills*) cu skill-urile introduse în CV.

La apăsarea butonului *btnAddSkillByText*, utilizatorului i se afișează un dialog în care i se cer informații despre skill-ul ce urmează a fi adăugat. În *dialog\_new\_skill* se

găsesc două `textBox`-uri(*tbAddSkill\_skillName* și *tbAddSkill\_skillDescription*), și un buton(*Add*). După completarea informațiilor și apăsarea butonului *ADD*, dialogul se închide, iar în lista cu abilități, va apărea noua abilitate adăugată.

La apăsarea butonului *btnAddSkillByVoice*, utilizatorul trebuie să introducă verba un skill. Cuvintele trebuie rostite tare și clar, pentru a înregistra date corecte. Datele vor fi înregistrate imediat după ce aplicația scoate un sunet(un beep).

De asemenea, în cazul unei erori sau a unei greșeli, se pot modifica abilitățile din lista, dând click pe elementul din lista ce se vrea a fi modificat/șters. Atunci când se face click pe un element din listă, dialog-ul *dialog\_new\_skill* re apare gata completat cu informațiile skill-ului selectat, dar de data aceasta avem butoanele **Update** și **Remove**. **Update** pentru a modifica abilitatea, și respectiv '**Remove**' pentru a o elimina.

Suprascrierea metodei `onStop()` din `Fragment\_Skill.java`, s-a făcut în felul următor:

```
@Override
public void onStop() {
    super.onStop();

    if (adapterSkills != null) {
        ArrayList<Skill> skills = new ArrayList<Skill>();
        for (int i = 0; i < adapterSkills.getCount(); i++) {
            skills.add(adapterSkills.getItem(i));
        }

        cv.setSkills(skills);
    }
    else {
        cv.setSkills(null);
    }
}
```

### 5.2.3 Education Fragment

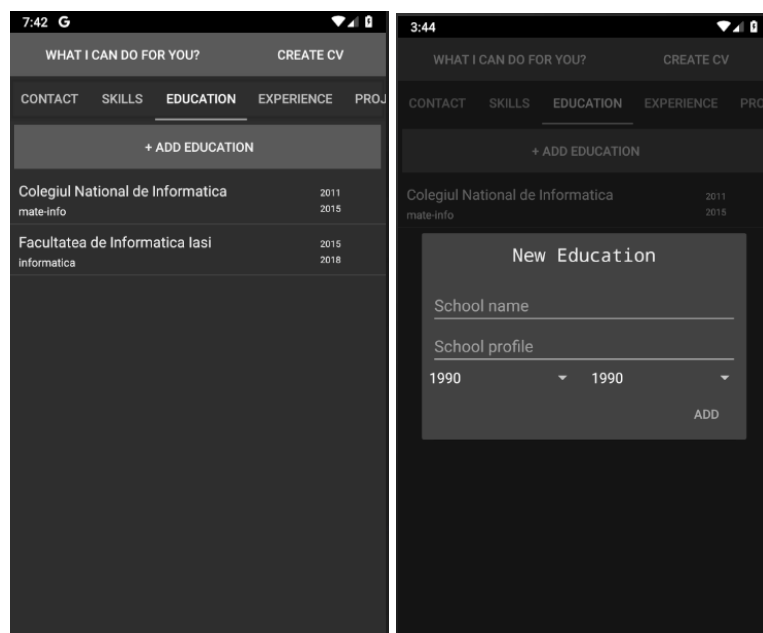


Figura 5.2: *Fragment Education (stânga) și dialog Add Education (dreapta)*

Fragmentul Education este alcătuit dintr-o listă(*lv\_currentEducations*) cu datele introduse în capitolul Education, și un buton(*btnAddEducation*) pentru adăugarea unui nou element în listă.

La apăsarea butonului *btnAddEducation*, se afișează dialogul *dialog\_new\_education*, în care se introduc datele ce urmează a fi adăgate în listă.

În cazul unei greșeli într-un element din lista *lv\_currentEducations*, se poate modifica sau șterge, dând click pe asupra căruia se dorește modificarea. La atingerea elementului, *dialog\_new\_education* va re-apărea pe ecran cu datele completate, dar butoanele vor fi Update și Remove, fiecare ocupându-se de modificarea, respectiv ștergerea elementului.

Suprascrierea metodei *onStop()* din *Fragment\\_Education.java*, s-a facut în felul următor:

```
@Override
public void onStop() {
    super.onStop();
}
```



```

if (adapterEducations != null) {
    ArrayList<Education> educations = new ArrayList<Education>();
    for (int i = 0; i < adapterEducations.getCount(); i++) {
        educations.add(adapterEducations.getItem(i));
    }

    cv.setEducation(educations);
}
else {
    cv.setEducation(null);
}
}

```

## 5.2.4 Experience Fragment

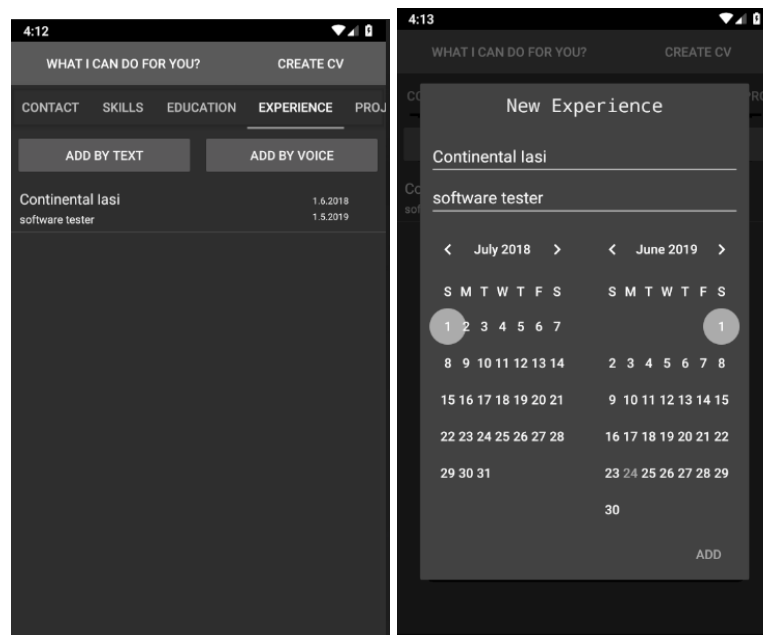


Figura 5.3: *Fragment Experience (stânga) și dialog Add Experience (dreapta)*

Acest fragment se ocupă de introducerea datelor din capitolul **Experice**. Aici se pot adăuga date folosind metoda clasică, *de la tastatură*, sau se pot introduce date folosind metode mai moderne, și anume *vocal*.

Introducerea datelor prin metoda clasică se poate face prin apăsarea butonului *btnAddExperienceByText*, moment în care se afișează dialogul *dialog\_new\_experience*. În acest dialog se introduc datele pentru Experiența ce se vrea adăugată: numele

instituției, poziția ocupată în cadrul acelui job, și prin intermediul celor două calendare, se poate alege perioada în care s-a acumulat experiența respectivă.

Introducerea datelor prin comenzi vocale, se poate face apăsând butonul *btnAddExperienceByVoice*. Imediat după apăsarea butonului, telefonul va scoate un sunet asemanator unui bipăit, iar aplicația așteaptă să primească informații noi. Când aplicația nu mai primește date noi, lista cu experiențe se va modifica, și se vor adăuga noile informații. În cazul în care datele nu sunt foarte bine recepționate și parsate de către aplicație, acestea se pot modifica manual, prin selectarea elementului ce se dorește a fi corectat.

Suprascrierea metodei `onStop()` din `Fragment\_Experience.java`, s-a făcut în felul următor:

```
@Override
public void onStop() {
    super.onStop();

    if (adapterExperiences != null) {
        ArrayList<Experience> experiences = new ArrayList<Experience>();
        for (int i = 0; i < adapterExperiences.getCount(); i++) {
            experiences.add(adapterExperiences.getItem(i));
        }

        cv.setExperiences(experiences);
    }
    else {
        cv.setExperiences(null);
    }
}
```

## 5.2.5 Projects Fragment

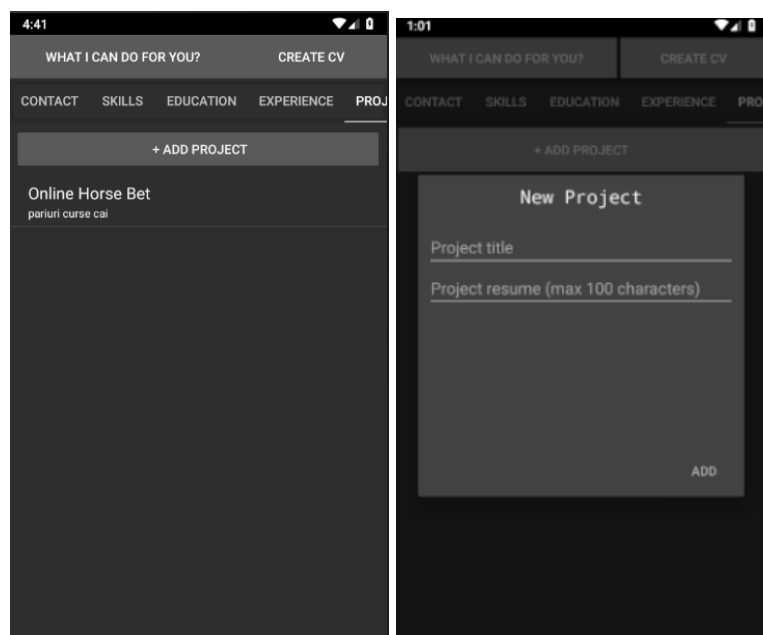


Figura 5.4: *Fragment Projects (stânga) și dialog Add Project (dreapta)*

Fragmentul Projects conține o listă(*lv\_currentProjects*) cu proiectele utilizatorului. La fel ca și în celelalte fragmente, pe aceeași listă se pot face modificări, cum ar fi inserarea unui element nou, modificarea unui element existent, sau ștergerea unui element.

Inserarea unui nou proiect, se face prin apăsarea butonului *btnAddProject*, moment la care se afisează dialogul *dialog\_new\_project*. În acest dialog se completează informații despre proiect, cum ar fi titlul și un rezumat:

Pentru modificarea, sau ștergerea unui element, se alege proiectul din lista *lv\_currentProjects*, se atinge elementul respectiv, iar dialogul *dialog\_new\_project* apare din nou, cu datele deja completate, dar cu butoane pentru modificarea elementului(*Update*) și pentru ștergerea proiectului(*Remove*).

Suprascrierea metodei `onStop()` din `Fragment\_Project.java`, s-a făcut în felul următor:

```

@Override
public void onStop() {
    super.onStop();

    if (adapterProjects != null) {
        ArrayList<Project> projects = new ArrayList<Project>();
        for (int i = 0; i < adapterProjects.getCount(); i++) {
            projects.add(adapterProjects.getItem(i));
        }
        cv.setProjects(projects);
    }
    else {
        cv.setProjects(null);
    }
}
}

```

## 5.2.6 Communication Fragment

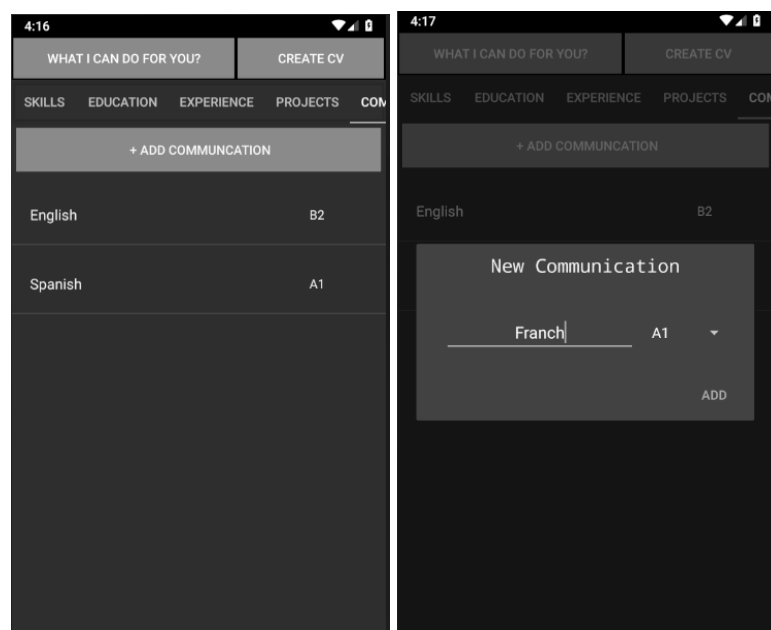


Figura 5.5: *Fragment Communication (stânga) și dialog Add Communication (dreapta)*

În acest Fragment, utilizatorul își poate adăuga limbile de circulație internațională ce le cunoaște. Fragmentul conține un buton(*btnAddComm*) pentru a adăuga o nouă limba străină cunoscută, și o listă(*lv\_currentComm*).

Adăugarea unei noi limbi străine, se face prin apăsarea butonului *btnAddComm*, și completarea formularului din *dialog\_new\_communication*. Informațiile se pot modifica / șterge, prin selectarea elementului din listă, modificarea datelor din formularul ce apare, și apăsarea butonului Update(pentru a modifica) sau Remove(pentru a șterge).

Suprascrierea metodei `onStop()` din `Fragment\_Communication.java`, s-a facut în felul următor:

```
@Override
public void onStop() {
    super.onStop();

    if (adapterCommunications != null) {
        ArrayList<Communication> communications = new
ArrayList<Communication>();
        for (int i = 0; i < adapterCommunications.getCount(); i++) {
            communications.add(adapterCommunications.getItem(i));
        }

        cv.setCommunications(communications);
    }
    else {
        cv.setCommunications(null);
    }
}
```

## 5.3 ActivityCVList

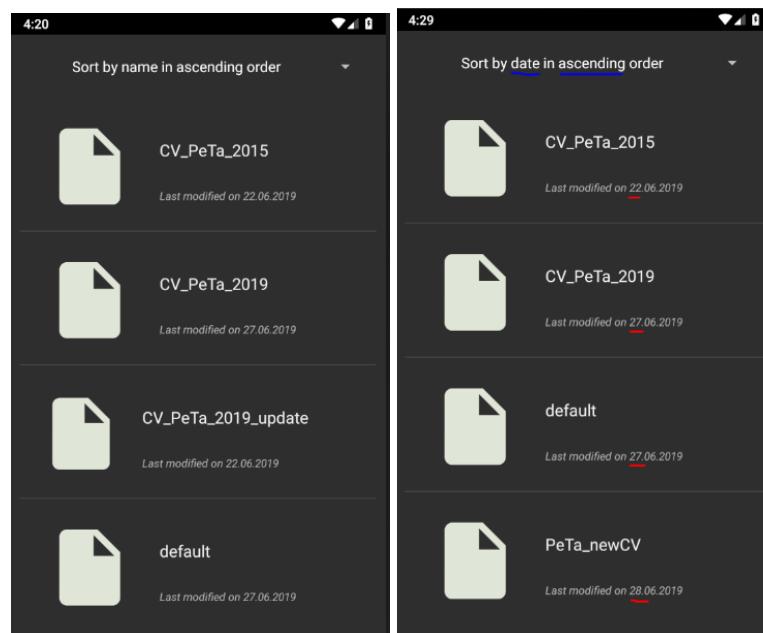


Figura 5.6: *Activity CV List*

Această activitate conține o listă cu toate CV-urile create. Lista conține alături de numele CV-ului și data la care au fost introduse informațiile în CV. Aici este și locul în care, la selectarea unui CV, se generează PDF-ul.

Pentru a naviga mai repede prin listă, utilizatorul poate ordona lista după nume sau dată, folosind ordonare crescătoare, sau descrescătoare. Imediat după selectarea unei ordonări, lista este re-aranjată.

La apăsarea scurtă pe un item din listă, se generează fișierul PDF ce conține datele CV-ului, dar și design-ul acestuia.

La apăsarea mai lungă pe un item, utilizatorului îi apare un meniu, prin care poate genera PDF-ul, folosind template-ul 1 sau 2, sau poate șterge acel CV

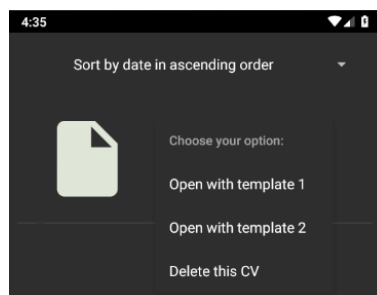


Figura 5.7: *Menu apăsare lungă*

# Concluzii

Ideea acestei aplicații a venit în momentul în care mi-am amintit cum am aplicat la primul job. Nu aveam nicio idee despre ce ar trebui să conțină un CV bun sau despre cum ar trebui să arate. Inițial m-am folosit de aplicații online de generat CV-uri, dar acestea conțineau informații nefolositoare job-ului pentru care aplicam.

Cu această aplicație sper să vin în ajutorul persoanelor care au nevoie un generator de CV-uri intuitiv, eficient și care folosește metode inteligente de a genera fișierul. Chiar dacă partea de UI a CV-ului este mai slabă în comparație cu top 3 aplicații asemănătoare din PlayStore, această aplicație folosește tehnici moderne de a introduce date în aplicație, iar plaja de viitoare update-uri este foarte largă, după cum urmează:

- o interfață mai prietenoasă a aplicației;
- mai multe template-uri pentru CV;
- posibilitatea de a încărca poze în CV;
- înregistrare vocală completă a CV-ului;
- în timpul înregistrării vocale, crearea unui filmuleț cu utilizatorul;
- generarea unei scrisori de intenție;
- posibilitatea de a distribui CV-ul (fișierul PDF) pe diferite platforme de recrutare (LinkedIn, ...);

Pentru mine, aceasă aplicație a fost o provocare și mă bucur că am reușit să o aduc la un stadiu de utilizare. Privind tot efortul depus în dezvoltarea ei, consider că m-a ajutat foarte mult să-mi dezvolt cunoștințele în programarea pe platforma Android, și aștept cu nerabdare să dezvolt cât mai multe aplicații moderne.

# Bibliografie

- [1] <https://dexonline.ro/definitie/cv>
- [2] <http://gs.statcounter.com/os-market-share/mobile/worldwide>
- [3] <https://profs.info.uaic.ro/acf/java/>
- [4] <https://sites.google.com/site/fiiandroidprogramming/courses>
- [5] [https://www.tutorialspoint.com/java\\_xml/java\\_xml\\_parsers.htm](https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm)
- [6] <https://itextpdf.com/en/resources/examples>
- [7] <https://developer.android.com/guide/components/fragments>
- [8] <https://guides.codepath.com/android/creating-and-using-fragments>
- [9] <https://developer.android.com/reference/android/widget/ArrayAdapter>
- [10] <https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView>
- [11] <https://www.androidtutorialpoint.com/material-design/android-speech-text-tutorial/>
- [12] <https://stackoverflow.com/questions/1337424/android-spinner-get-the-selected-item-change-event>
- [13] <https://stackoverflow.com/questions/17207366/creating-a-menu-after-a-long-click-event-on-a-list-view>