Raport final

Silaghi Alex-Cristian
Universitatea de Vest din Timişoara
Facultatea de Matematică şi Informatică
Anul 2

Programare pe dispozitive mobile
16.05.2021
Abopress

Abstract

Acest raport prezintă o schiţă despre proiectul pe care l-am realizat şi anume o aplicaţie mobilă care are ca scop ajutarea factorilor poştali în gestionarea volumului de muncă. Raportul este structurat in două părţi. În prima parte voi prezenta anumite aspecte generale cum ar fi: scopul aplicaţiei şi potenţiali utilizatori împreuna cu aplicaţiile similare la momentul începerii proiectului, iar în partea a doua structura aplicaţiei şi direcţiile viitoare în procesul de dezvoltare a aplicaţiei.

Scop şi potenţiali utilizatori

Aplicatia se adreseaza in special postasilor si curierilor, dar ea poate fi adaptata de utilizator in a asa fel in cat aceasta sa ii deserveasca si in alte situatii. Scopul aplicatiei este da a ajuta utilizatorul in a-şi gestiona o parte din volumul de munca.

Introducere

Aplicaţia are rolul de a ajuta poştaşul să îşi gestioneze presa, ţinând evidenţa tipurilor de presă pe care acesta le are de distribuit pe sector (Renaşterea, Ziarul financiar, etc) şi ulterior putând să adauge o listă cu toţi abonaţii de pe sector care au abonament pentru ziarul respectiv împreună cu numărul de telefon al acestuia, astfel oferindu-i posibilitatea utilizatorului de a trimite simultan mesaje beneficiarilor de ziare, anuntandu-i că abonamentul acestora expiră

Contribuţia autorului

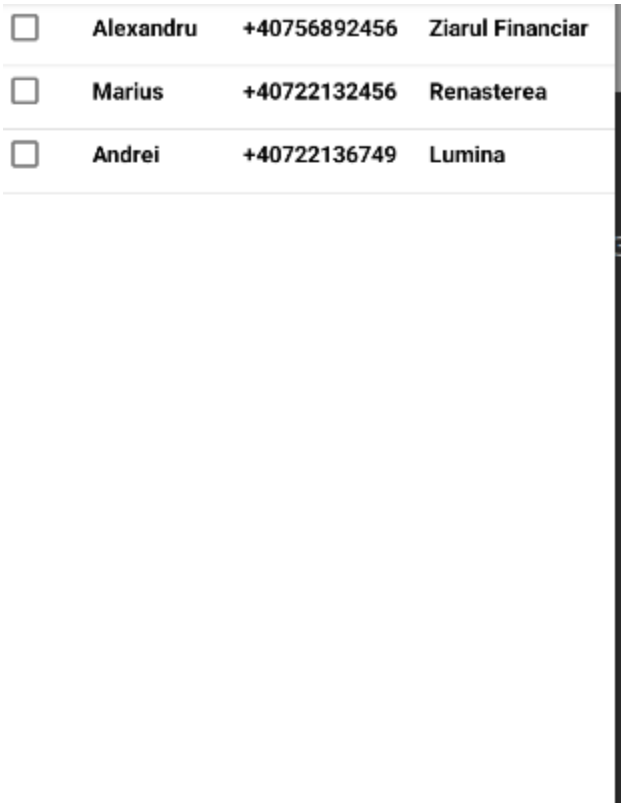
Contrubuţia personală ar fi combinarea celor 2 caracteristici si anume: oferirea unui catalog unde utilizatorul poate ţine evidenţa presei, dar şi trimiterea beneficiarilor de presa mesaje pentru a-i anunţa că abonamentul lor expiră şi intervalul în care poştaşul va trece pe la ei în cazul în care aceştia vor să-şi prelungească abonamentul la ziarul respectiv.

Funcţionalitate şi structura aplicaţiei

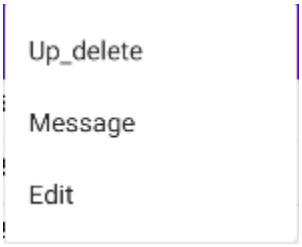
Aplicaţia a fost relizată în Android Studio şi a fost folosit limbajul Java, XML şi tehnologii precum SQLlite pentru a stoca informaţiile introduse de către utilizator. Structura aplicaţiei este una simplistă.

Aplicaţia se deschide cu o pagină unde se poate adauga un abonat în listă. Pe această pagină sunt prezente mai multe câmpuri şi anume: un câmp pentru nume, unul pentru numărul de telefon şi unul pentru numele ziarului/revistei la care persoana respectivă s-a abonat. De asemenea utilizatorul mai are un buton pentru a adăuga persoana în lista şi un buton pentru a vizualiza lista de abonaţi.

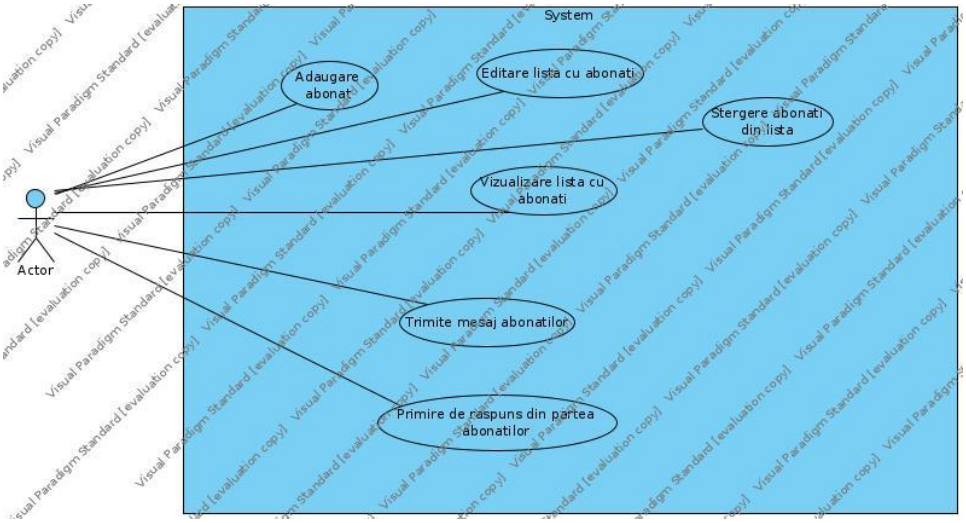
Butonul „Lista” afisează lista cu abonaţii care au fost înregistraţi în aplicaţie.



În această pagină sunt disponibile şi anumite actiuni bază, cum ar fi editarea, ştergerea sau trimiterea unui mesaj de înştiinţare cum că abonamentul persoanei respective va expira curânt.



Mai jos ataşat este diagrama cazurilor de utilizare



Codul

Aplicaţia foloseste un model numit CostumerModel:

```
public class CostumerModel implements Parcelable {
    private String name;
    private String Publicatie;
    private String NrTel;
    boolean selected=false;
    public void setSelected(boolean selected) { this.selected = selected; }
    public CostumerModel(String name, String nrTel,String Publicatie) {
        this.name = name;
        NrTel = nrTel;
        this.Publicatie = Publicatie;
    }
    public CostumerModel() {
    }
    protected CostumerModel(Parcel in) {
        name = in.readString();
        NrTel = in.readString();
        Publicatie = in.readString();
    }
    public static final Creator<CostumerModel> CREATOR = new Creator<CostumerModel>() {
        @Override
        public CostumerModel createFromParcel(Parcel in) { return new CostumerModel(in); }

        @Override
        public CostumerModel[] newArray(int size) { return new CostumerModel[size]; }
    };
    public String getName() { return name; }
    public String getNrTel() { return NrTel; }
```

```
        public CostumerModel[] newArray(int size) { return new CostumerModel[size]; }
    };
    public String getName() { return name; }
    public String getNrTel() { return NrTel; }
    public String getPublicatie(){return Publicatie;}
    public void setName(String name) { this.name = name; }
    public void setNrTel(String nrTel) { NrTel = nrTel; }
    @Override
    public String toString() {
        return "CostumerModel{" +
                "name='" + name + '\'' +
                ", Publicatie='" + Publicatie + '\'' +
                ", NrTel='" + NrTel + '\'' +
                '}';
    }

    public void setPublicatie(String Publicatie){this.Publicatie = Publicatie;}
    @Override
    public int describeContents() { return 0; }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeString(NrTel);
        dest.writeString(Publicatie);
    }
    public boolean getSelected() { return selected; }
}
```

De asemenea modul cum baza de date a fost implementată:

```
public class CostumerModel implements Parcelable {
    private String name;
    private String Publicatie;
    private String NrTel;
    boolean selected=false;
    public void setSelected(boolean selected) { this.selected = selected; }
    public CostumerModel(String name, String nrTel,String Publicatie) {
        this.name = name;
        NrTel = nrTel;
        this.Publicatie = Publicatie;
    }
    public CostumerModel() {
    }
    protected CostumerModel(Parcel in) {
        name = in.readString();
        NrTel = in.readString();
        Publicatie = in.readString();
    }
    public static final Creator<CostumerModel> CREATOR = new Creator<CostumerModel>() {
        @Override
        public CostumerModel createFromParcel(Parcel in) { return new CostumerModel(in); }

        @Override
        public CostumerModel[] newArray(int size) { return new CostumerModel[size]; }
    };
    public String getName() { return name; }
    public String getNrTel() { return NrTel; }
```

```
        public CostumerModel[] newArray(int size) { return new CostumerModel[size]; }
    };
    public String getName() { return name; }
    public String getNrTel() { return NrTel; }
    public String getPublicatie(){return Publicatie;}
    public void setName(String name) { this.name = name; }
    public void setNrTel(String nrTel) { NrTel = nrTel; }
    @Override
    public String toString() {
        return "CostumerModel{" +
                "name='" + name + '\'' +
                ", Publicatie='" + Publicatie + '\'' +
                ", NrTel='" + NrTel + '\'' +
                '}';
    }

    public void setPublicatie(String Publicatie){this.Publicatie = Publicatie;}
    @Override
    public int describeContents() { return 0; }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeString(NrTel);
        dest.writeString(Publicatie);
    }
    public boolean getSelected() { return selected; }
}
```

```
}
public boolean addOne (CostumerModel costumerModel){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(CUSTOMER_NAME,costumerModel.getName());
    cv.put(NUMARTEL,costumerModel.getNrTel());
    cv.put(PUBLICATIE, costumerModel.getPublicatie());

    long insert = db.insert(CUSTOMER_TABLE,  nullColumnHack: null, cv);
    return insert != -1;
}

public boolean deleteOne(CostumerModel costumerModel ){

    SQLiteDatabase db = this.getWritableDatabase();
    String queryString = String.format("DELETE FROM %s WHERE NUMARTEL='%s'", CUSTOMER_TABLE,costumerModel.getNrTel());
    Cursor cursor = db.rawQuery(queryString, selectionArgs: null);
    return cursor.moveToFirst();
}

public List<CostumerModel> getEverybody(){

    List<CostumerModel> returnList = new ArrayList<>();
    String queryString = "SELECT * FROM " + CUSTOMER_TABLE;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(queryString,  selectionArgs: null);
    if (cursor.moveToFirst()) {
        do{
```

```
}

public List<CostumerModel> getEverybody(){

    List<CostumerModel> returnList = new ArrayList<>();
    String queryString = "SELECT * FROM " + CUSTOMER_TABLE;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(queryString, selectionArgs: null);
    if (cursor.moveToFirst()) {
        do{
            String cursorPublicatie = cursor.getString( columnIndex: 2);
            String cursorName = cursor.getString( columnIndex: 1);
            String cursorTel = cursor.getString( columnIndex: 0);

            CostumerModel costumerModel = new CostumerModel(cursorName,cursorTel,cursorPublicatie);
            returnList.add(costumerModel);

        }while(cursor.moveToNext());
    }
    cursor.close();
    db.close();
    return returnList;
}

public void Edit(CostumerModel OldCostumerModel, CostumerModel NewCostumerModel ) {
    String queryString = String.format("UPDATE %s SET CUSTOMER_NAME='%s', NUMARTEL='%s', PUBLICATIE='%s' WHERE CUSTOMER_NAME='%s' AND NUMARTEL='%s' AND
    SQLiteDatabase db = this.getWritableDatabase();
```

Directii viitoare

- Implementarea unui facilităţi prin care utilizatorul să primească recomandări de ziare, recomandări bazate pe produsele la care este deja abonat
- Implementarea unei modalităţi de achitare a abonamentului prin intermediul aplicaţiei

Referinţe

https://www.youtube.com/watch?v=4GYKOzgQDWI

https://www.youtube.com/watch?v=eSM_YkWeS7k

https://developer.android.com/docs