Aplikacja monitorująca budżet domowy

1.Opis projektu

Celem tego projektu było stworzenie aplikacji monitorującej przychody oraz wydatki gospodarstwa domowego. W aplikacji każdy użytkownik tworzy własne konto z przypisanym do niego adresem E-mail oraz hasłem. Każdy użytkownik może dodawać swoje przychody oraz wydatki podając kategorię oraz kwotę pieniędzy. Listę wydatków oraz przychodów można później przeglądać. Oprócz tego możemy przeglądać statystyczne zestawienie na wykresie kołowym ile pieniędzy zostało przeznaczone na jaką kategorię. Do przechowywania danych w bazie użyłem SQLite. Było to o tyle proste, że aplikacje android mają wbudowaną bazę SQLite oraz szereg funkcjonalności do jej obsługi.

2.Baza danych SQLite

Model bazy danych składa się z 5 tabelek z danymi, które połączone są relacjami. Są to tabelki: Osoba, Wydatek, Przychód, Kategoria wydatku i Źródło przychodu. Poniżej przedstawiam schemat bazy danej w programie DB Browser.

Nazwa	Rodzaj	Polecenie tworzące
Tabele (7)		
android_metadata		CREATE TABLE android_metadata (locale TEXT)
kategoria_wydatku		CREATE TABLE kategoria_wydatku(zrodlo_id INTEGER PRIMARY KEY AUTOINCREMENT,zrodlo_przychodu TEXT NOT NULL,zrodlo_opis TEXT)
zrodlo_id	INTEGER	"zrodlo_id" INTEGER
zrodlo_przychodu	TEXT	"zrodlo_przychodu" TEXT NOT NULL
zrodlo_opis	TEXT	"zrodlo_opis" TEXT
osoba		CREATE TABLE osoba(id_osoby INTEGER PRIMARY KEY AUTOINCREMENT,imie TEXT,Email TEXT,haslo TEXT)
id_osoby	INTEGER	"id_osoby" INTEGER
imie	TEXT	"imie" TEXT
Email	TEXT	"Email" TEXT
haslo	TEXT	"haslo" TEXT
przychod		CREATE TABLE przychod(id_przychodu INTEGER PRIMARY KEY AUTOINCREMENT,kwota_przychodu INTEGER,data_przychodu TEXT,zrodlo_prz
id_przychodu	INTEGER	"id_przychodu" INTEGER
kwota_przychodu	INTEGER	"kwota_przychodu" INTEGER
data_przychodu	TEXT	"data_przychodu" TEXT
zrodlo_przychodu	TEXT	"zrodlo_przychodu" TEXT
id_uzytkownika	INTEGER	"id_uzytkownika" INTEGER
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
wydatek		CREATE TABLE wydatek(wydatek_id INTEGER PRIMARY KEY AUTOINCREMENT,wartosc_wydatku INTEGER,data_wydatku TEXT,kategoria_wyd
wydatek_id	INTEGER	"wydatek_id" INTEGER
wartosc_wydatku	INTEGER	"wartosc_wydatku" INTEGER
data_wydatku	TEXT	"data_wydatku" TEXT
kategoria_wydatku	TEXT	"kategoria_wydatku" TEXT
id_uzytkownika	INTEGER	"id_uzytkownika" INTEGER
zrodlo_przychod		CREATE TABLE zrodlo_przychod(zrodlo_id INTEGER PRIMARY KEY AUTOINCREMENT,zrodlo_przychodu TEXT NOT NULL,zrodlo_opis TEXT)
zrodlo_id	INTEGER	"zrodlo_id" INTEGER
zrodlo_przychodu	TEXT	"zrodlo_przychodu" TEXT NOT NULL
zrodlo_opis	TEXT	"zrodlo_opis" TEXT

Rysunek 1 Układ bazy danych

Aby wygenerować tabelki w bazie danych należy w aplikacji należy napisać specjalne komendy, które zapisane są w kodzie SQL. Następnie za pomocą funkcji execSQL zostają one utworzone.

```
// create table sql query
private val CREATE_TABLE_USER = ("CREATE TABLE " + TABLE_USER + "("
        + COLUMN_USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN_USER_NAME + " TEXT," + COLUMN_USER_EMAIL + " TEXT," + COLUMN_U

private val CREATE_TABLE_SOURCE=( "CREATE TABLE "+ TABLE_SOURCE+"("
        + COLUMN_SOURCE_ID+" INTEGER PRIMARY KEY AUTOINCREMENT,"+ COLUMN_SOURCE_NAME+" TEXT NOT NULL,"+
        COLUMN_SOURCE_DESCRIBE+" TEXT);")
private val CREATE_TABLE_CATEGORY=( "CREATE TABLE "+ TABLE_CATEGORY+"("
        + COLUMN_CATEGORY_ID+" INTEGER PRIMARY KEY AUTOINCREMENT,"+ COLUMN_CATEGORY_NAME+" TEXT NOT NULL,"+
        COLUMN_CATEGORY_DESCRIBE+" TEXT);")
private val CREATE_TABLE_INCOME=( "CREATE TABLE " + TABLE_INCOME + "("
        + COLUMN_INCOME_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN_INCOME_VALUE + " INTEGER,"
        + COLUMN_INCOME_DATE + " TEXT," + COLUMN_INCOME_SOURCE + " TEXT, "
        + COLUMN_INCOME_USER + " TEXT, "+"FOREIGN KEY ("+ COLUMN_INCOME_USER+
        ") REFERENCES "+ TABLE_USER+"("+ COLUMN_USER_NAME+")"+"FOREIGN KEY ("+ COLUMN_INCOME_SOURCE+
    ") REFERENCES "+ TABLE_SOURCE+"("+ COLUMN_SOURCE_NAME+")"+ ");")
```

Rysunek 2 Wyrażenia SQLite tworzące tabelki

```
override fun onCreate(db: SQLiteDatabase) {
    db.execSQL(CREATE_TABLE_USER)
    db.execSQL(CREATE_TABLE_SOURCE)
    db.execSQL(CREATE_TABLE_CATEGORY)
    db.execSQL(CREATE_TABLE_INCOME)
    db.execSQL(CREATE_TABLE_EXPENSE)
```

Rysunek 3 Wywołanie wyrażeń SQLite

Oprócz tego mamy również funkcje, dynamicznie dodające dane do bazy danych oraz wyszukujące dane w bazie. Aby tego dokonać należało w odpowiedni sposób skonfigurować wyrażenie query w funkcji.

```
@SuppressLint( ...value: "Range")
fun getAllUser(): List<User> {
    // array of columns to fetch
    val columns = arrayOf(COLUMN_USER_ID, COLUMN_USER_EMAIL, COLUMN_USER_NAME, COLUMN_USER_PASSWORD)
    // sorting orders
    val sortOrder = "$COLUMN_USER_NAME ASC"
    val userList = ArrayList<User>()
    val db = this.readableDatabase
    // query the user table
    val cursor = db.query(TABLE_USER, //Table to query
        columns,            //columns to return
        selection: null,     //columns for the WHERE clause
        selectionArgs: null,  //The values for the WHERE clause
        groupBy: null,      //group the rows
        having: null,       //filter by row groups
        sortOrder)          //The sort order
    if (cursor.moveToFirst()) {
        do {
            val user = User(id = cursor.getString(cursor.getColumnIndex(COLUMN_USER_ID)).toInt(),
                name = cursor.getString(cursor.getColumnIndex(COLUMN_USER_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_USER_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_USER_PASSWORD)))
            userList.add(user)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return userList
}
```

Rysunek 4 Funkcja zwracająca listę zarejestrowanych użytkowników

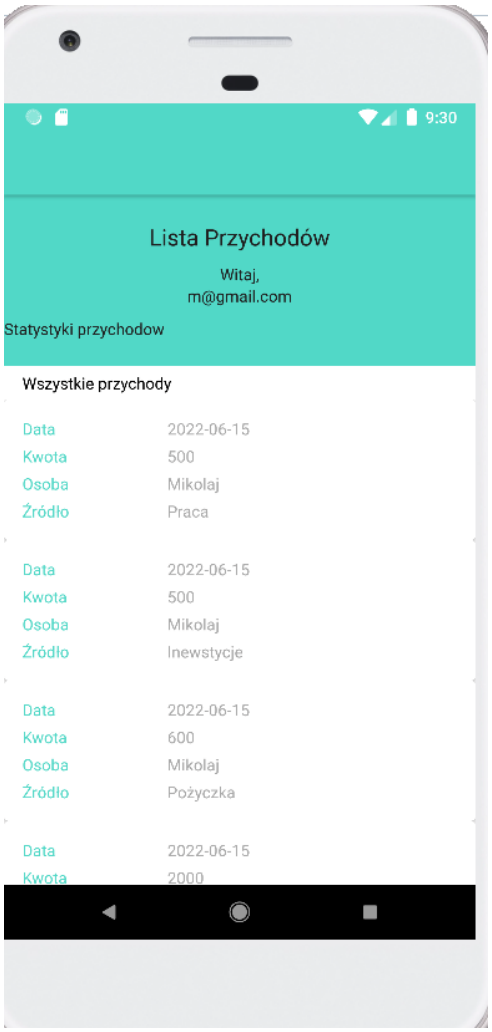
Bardzo istotną rzeczą podczas tworzenia bazy danych w aplikacji androidowej jest to aby za każdym razem gdy wprowadzamy jakąś zmianę zmienić numer wersji bazy danych. Zostaną wtedy usunięte wszystkie rekordy i zajdą zmiany w bazie. Jeżeli nie zmienimy wersji albo nie zresetujemy telefonu nie zajdą żadne zmiany.

3.Funkcjonalności aplikacji

W aplikacji zastosowane zostało kilka funkcjonalności takie jak mechanizm logowania oraz rejestracji użytkownika, dodawanie wydatków oraz przychodów i przeglądanie listy wydatków, przychodów i użytkowników a także podgląd w statystyki. Przy mechanizmie rejestracji zostały zaimplementowane metody walidacji danych polegające na sprawdzeniu czy dane pole nie jest puste, podczas wpisywania e-maila sprawdzane jest czy e-mail zawiera domenę a także czy dwukrotnie wpisano to samo hasło. Podczas rejestracji dane umieszczane są w bazie danych za pomocą funkcji służącej do dodawania rekordów do tabelki. Pomysł na system logowania i rejestracji został zaczerpnięty z http://www.androidtutorialshub.com/android-login-and-register-with-sqlite-database-tutorial/.

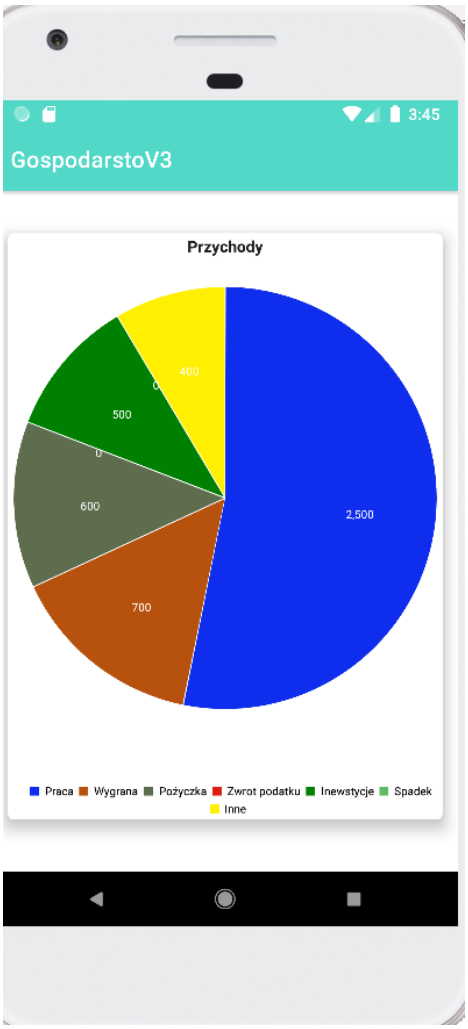
Dodawanie wydatków i przychodów odbywa się w taki sposób, że wybieramy kategorię z rozwijanej listy oraz wpisujemy kwotę pieniędzy. Jeżeli wszystko przeprowadzimy poprawnie to wybieramy dodaj i nasza informacja zostanie dodana do bazy danych wraz z imieniem osoby aktualnie zalogowanej oraz aktualną datą.

Rysunek 5 Dodawanie przychodu



Dodatkowo stworzony zostaje wykres kołowy, pokazujący statystyki naszych wydatków oraz przychodów. Biblioteka wykresów nie jest częścią kotlina i została dodana z repozytorium

https://github.com/PhilJay/MPAndroidChart. Do reprezentacji statystyk zastosowałem wykres kołowy wraz z legendą. Aby wykres ten mógł powstać dla każdego rodzaju kategorii należy pobrać z bazy danych wartość kwoty dla której została ona wykonana, zsumować wszystkie kwoty z ta samą kategorią a następnie zamknąć w tablicy i podać do wykresu.



Rysunek 6 Wykres przychodów

```
fun getPieChartArrays()
{
    val databaseHandler: DatabaseHelper = DatabaseHelper( context: this)
    val sourceArray=resources.getStringArray(R.array.Sources)

    val valueArray = IntArray( size: 7){0}
    var i = 0
    for(s in sourceArray)
    {
        valueArray[i]=databaseHandler.getSumIncomeBySource(s)
        i++
    }
    populatePieChart(valueArray, sourceArray)
```

Rysunek 7 Funkcja odpowiedzialna za stworzenie tablicy z danymi

4.Problemy

W czasie tworzenia projektu powstało sporo problemów, z którymi musiałem sobie poradzić podczas implementacji kodu. Musiałem również zrezygnować z niektórych rzeczy takich jak przydzielanie uprawnień użytkownikom. Jednym z problemów było wspomniane odświeżanie bazy danych. Oprócz tego w kotlinie często pojawiały się

problemy z kompatybilnością danych z czym również musiałem sobie radzić. Podczas tworzenia statystyk zdecydowałem się dodać „na sztywno” listę kategorii w liście rozwijanej tak aby wyniki były miarodajne.