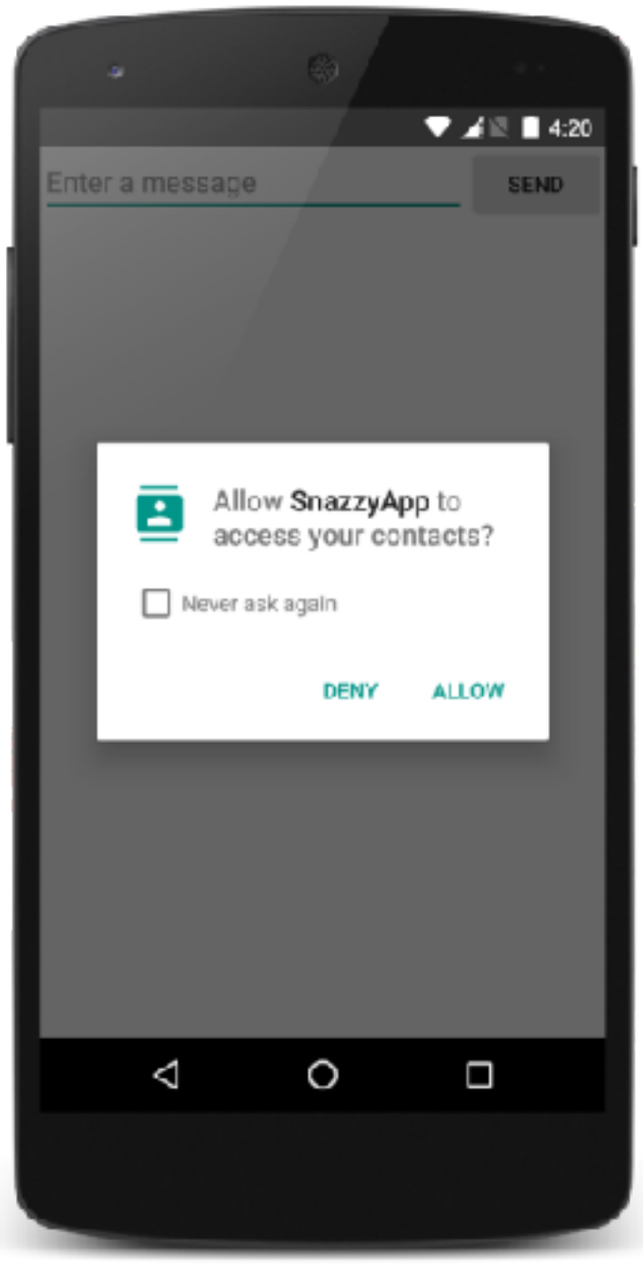
# CV3

# Permissions

- API < 23 vs API >= 23
- AndroidManifest.xml
- Úroveň ochrany:
  - **Normal** - vibrace, BT, internet - automatické povolení systémem
  - **Dangerous** - čtení sms, kontakty, poloha - vyžaduje interakci uživatele

# Dangerous Permissions




**CALENDAR**
READ_CALENDAR
WRITE_CALENDAR

**CONTACTS**
READ_CONTACTS
WRITE_CONTACTS
GET_ACCOUNTS

**SENSORS**
BODY_SENSORS

**CAMERA**
CAMERA

**LOCATION**
ACCESS_FINE_LOCATION
ACCESS_COARSE_LOCATION

**MICROPHONE**
RECORD_AUDIO

**STORAGE**
READ_EXTERNAL_STORAGE
WRITE_EXTERNAL_STORAGE

**PHONE**
READ_PHONE_STATE
CALL_PHONE
READ_CALL_LOG
WRITE_CALL_LOG
ADD_VOICEMAIL
USE_SIP
PROCESS_OUTGOING_CALLS

**SMS**
SEND_SMS
RECEIVE_SMS
READ_SMS
RECEIVE_WAP_PUSH
RECEIVE_MMS

# Implementace

```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity,
                  Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {

    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
            Manifest.permission.READ_CONTACTS)) {

        // Show an explanation to the user *asynchronously* -- don't block
        // this thread waiting for the user's response! After the user
        // sees the explanation, try again to request the permission.

    } else {

        // No explanation needed, we can request the permission.

        ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.READ_CONTACTS},
                MY_PERMISSIONS_REQUEST_READ_CONTACTS);

        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
        // app-defined int constant. The callback method gets the
        // result of the request.
    }
}
```

# Storage

- SharedPreferences, Hawk
- Soubor
- Databaze
- ORM

# SharedPreferences

https://developer.android.com/reference/android/content/SharedPreferences.html

- Jednoduchý key-value storage
- String, Boolean, Float, Int, Set<String>
- context.getSharedPreferences(“my”, 0)

# SharedPreferences

```
val sharedPrefs: SharedPreferences =
context.getSharedPreferences(“my”, 0)

sharedPrefs.getString(“name”, “”) // druhy param je default
sharedPrefs.getInt(“id”, 0)
sharedPrefs.contains(“name”) // true-false


val editor: SharedPreferences.Editor = sharedPrefs.edit()
editor.putString(“name”, “Radim”)
editor.commit(); // nebo editor.apply()
```

# Zadání

- Vytvořte aplikaci, která bude zobrazovat počet spuštění - číslo při každém spuštění aplikace

# Řešení

```
val sharedPrefs: SharedPreferences =
context.getSharedPreferences(“my”, 0)

val count = sharedPrefs.getInt(“count”, 0)

textview.text = String.valueOf(count)

sharedPrefs.edit().putInt(“count”, count + 1).apply();
```

# Hawk

https://github.com/orhanobut/hawk

```
Hawk.init(context).build()

Hawk.put(key, T)

T value = Hawk.get(key)

Hawk.delete(key)

Hawk.contains(key)

compile 'com.orhanobut:hawk:2.0.1'
```

# Soubor

- `getFilesDir(), getCacheDir(), getExternalFilesDir()`
- `android.permission.WRITE_EXTERNAL_STORAGE`
- `File – FileInputStream, FileOutputStream`

# Soubor - příklad

```
val filename: String = “hello_file"
val data: String = "hello world!"

val fos: FileOutputStream =
  openFileOutput(filename, Context.MODE_PRIVATE)
fos.write(data.bytes)
fos.close()
```

# Database SQL

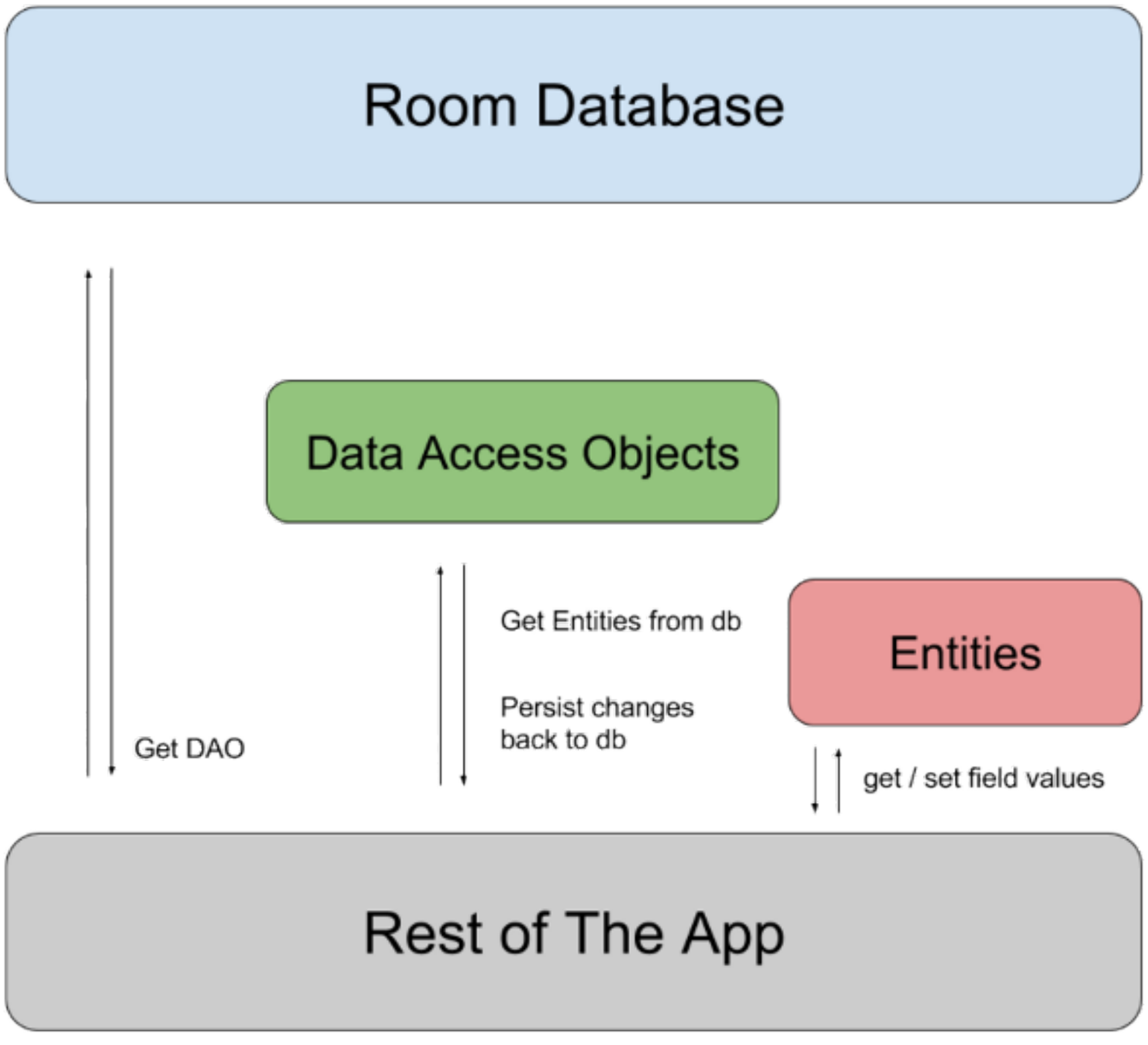
https://developer.android.com/training/basics/data-storage/databases.html

```
public class MySQLiteOpenHelper extends SQLiteOpenHelper {
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = “MyDb.db";

    public MySQLiteOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}

MySQLiteOpenHelper myDbHelper = new MySQLiteOpenHelper(getContext());
SQLiteDatabase db = myDbHelper.getReadableDatabase();
db.beginTransaction(); …
```

# Room


Room Database
Data Access Objects
Get DAO
Get Entities from db
Persist changes
back to db
Entities
get / set field values
Rest of The App

# Room - Entity

```
@Entity(tableName = "user")
data class Person(
        @PrimaryKey
        @ColumnInfo(name = "name")
        var name: String = ""
)
```

# Room - DAO

```
@Dao
interface PersonDao {
    @Query("SELECT * FROM person")
    fun findAll(): List<Person>

    @Delete
    fun remove(person: Person)

     @Insert
    fun insert(person: Person)
}
```

# Room - AppDatabase

```
@Database(entities = arrayOf(Person::class), version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun personDao(): PersonDao
}
```

# Room - Použití

```
Room.databaseBuilder(this, AppDatabase::class.java, "db")
        .allowMainThreadQueries()
        .build()
```

```
db.personDao().insert(Person(“Radim”))
db.personDao().delete(Person(“Radim”))
db.personDao().findAll()
```