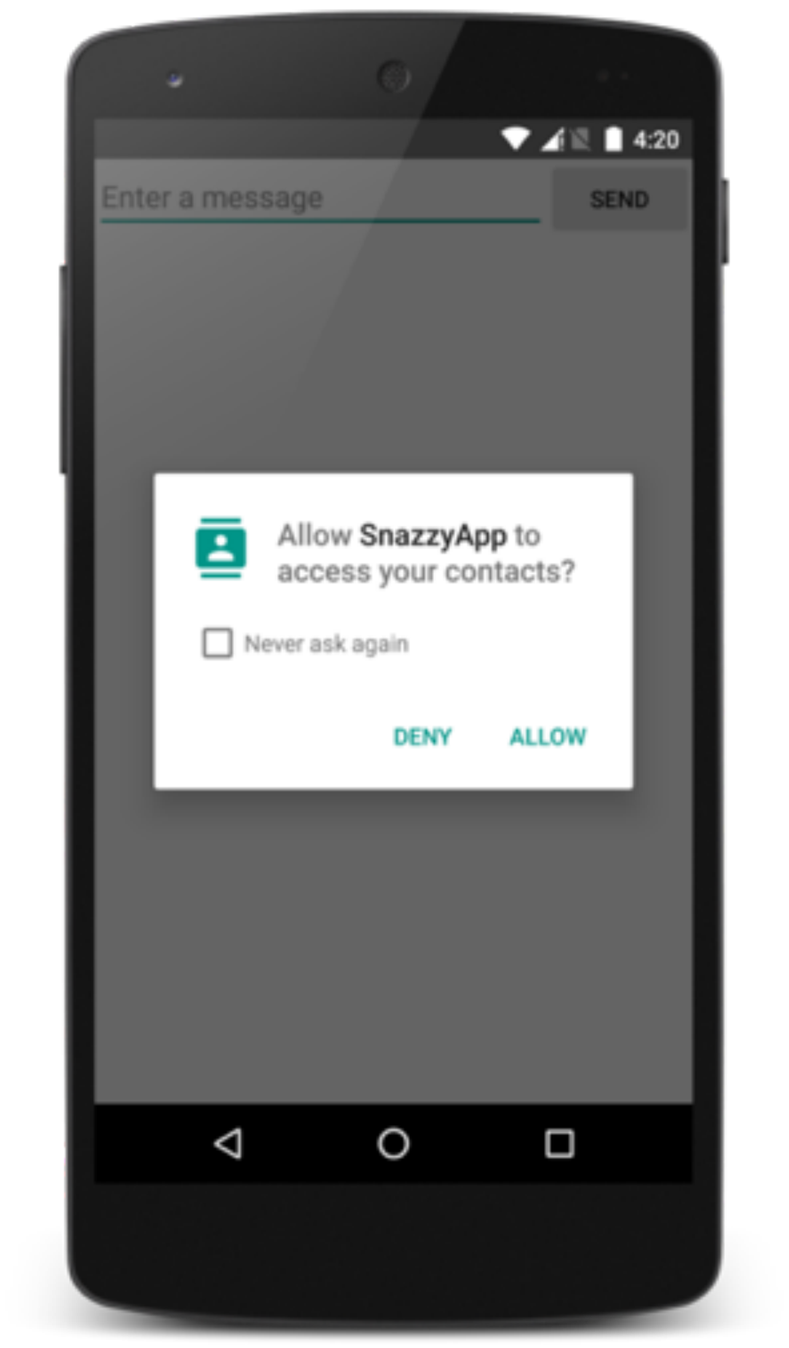
# CV3

## 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



Allow SnazzyApp to access your contacts?

**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

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

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

SharedPreferences.Editor 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

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

int count = sharedPrefs.getInt(“count”, 0);

textview.setText(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

```
String FILENAME = “hello_file";
String string = "hello world!";


FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
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(); …
```

# Realm

https://realm.io

- ORM
- lazy-loaded
- Android (Java), iOS (Swift, Obj-C), Tamarin

# Realm - objekt

```
public class Dog extends RealmObject {
    private String name;
    private int age;
}

public class Person extends RealmObject {
    @PrimaryKey
    private long id;
    private String name;
    private RealmList<Dog> dogs;
}
```

# Realm - inicializace

```
Realm.init(context);

// Get a Realm instance for this thread
Realm realm = Realm.getDefaultInstance();
```

# Realm - insert

```
realm.beginTransaction();

User user = realm.createObject(User.class, primaryKey)
user.setName(“Radim”);

realm.commitTransaction();
```

nebo

```
User user = new User();
user.setName(“Radim");

realm.beginTransaction();
User realmUser = realm.copyToRealm(user);
realm.commitTransaction();
```

# Realm - async

```
realm.executeTransactionAsync(new Realm.Transaction() {
            @Override
            public void execute(Realm r) {
                User user = r.createObject(User.class);
                user.setName("John");
                user.setEmail("john@corporation.com");
            }
        }, new Realm.Transaction.OnSuccess() {
            @Override
            public void onSuccess() {
                // Transaction was a success.
            }
        }, new Realm.Transaction.OnError() {
            @Override
            public void onError(Throwable error) {
                // Transaction failed and was automatically canceled.
            }
        });
```

# Realm - query

```
RealmResults<User> result = realm.where(User.class)
    .findAll();
```

```
RealmResults<User> result = realm.where(User.class)
    .equalTo("name", "John")
    .or()
    .equalTo("name", "Peter")
    .findAll()
    .sort(“age", Sort.DESCENDING);
```

# Realm - delete

```
realm.where(User.class)
  .findAll()
  .deleteAllFromRealm();
```

## Zadání

- Doplňte chybějící funkce Realmu pro přidání, mazání a získání dat

# Řešení

```
public void onClick(View view) {
  realm.beginTransaction();
  Todo user = realm.createObject(Todo.class, UUID.randomUUID().toString());
  user.setName(input.getText().toString());
  realm.commitTransaction();
}
```

```
RealmResults<Todo> todos = realm.where(Todo.class).findAll();
adapter = new TodoAdapter(todos);
```

```
Todo todo = adapter.getItem(i);
realm.beginTransaction();
realm.where(Todo.class)
  .equalTo("uuid", todo.getUuid())
  .findAll()
  .deleteAllFromRealm();
realm.commitTransaction();
```