УДК 004.428.2 DOI: 10.30748/soivt.2018.56.21

В.М. Федорченко[1], О.В. Сєвєрінов[2], С.В. Родіонов[3]

[1] *Харківський національний економічний університет ім. С. Кузнеця, Харків*
[2] *Харківський національний університет радіоелектроніки, Харків*
[3] *Український державний університет залізничного транспорту, Харків*

АНАЛІЗ ORM-БІБЛІОТЕК ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

Предметом дослідження в статті є процес розробки мобільних data-орієнтованих додатків для платформи Android. Стаття присвячена детальному аналізу сучасних методів управління базами даних в ОС Android. Мета роботи – оцінка ефективності різних бібліотек, що забезпечують взаємодію додатку з БД і реалізують ORM - технологію програмування. У статті вирішуються наступні завдання: розгляд і аналіз найбільш поширених ORM-бібліотек сторонніх розробників для платформи Android. Основний зміст дослідження становить порівняльний аналіз найбільш поширених ORM-бібліотек і результатів їх роботи в однакових умовах. Отримані наступні результати: проаналізовано складність використання API бібліотек при програмної реалізації ORM-моделі. Наведено результати дослідження швидкодії виконання CRUD-операцій в мобільному додатку із застосуванням розглянутих бібліотек. Висновки: проведений аналіз дозволяє підвищити ефективність процесу розробки і масштабування мобільних додатків в частині управління базами даних для платформи Android.

Ключові слова: *база даних, Android, ORM, ORM-бібліотека, ActiveAndroid, ORMLite, GreenDAO, SugarORM.*

Вступ

Постановка проблеми. Зберігання великих обсягів структурованих даних – дуже важливе завдання для сучасних систем управління. Крім зберігання, необхідно займатися обробкою цих даних, що вимагає постійних звертань до бази даних (БД). Природно, що все повинно виконуватися максимально швидко і при цьому формат представлення даних і код обробки для програміста повинен бути якомога більш легким для читання, і його написання не повинно займати багато часу. Очевидно, що ідеального підходу не існує, тому потрібно шукати компроміс між форматом подання оброблюваних об'єктів при написанні об'єктно-орієнтованих програм і можливістю максимально швидких звернень до необхідних таблиць бази даних. Саме це і є головною проблемою в створенні СУБД на основі принципу ORM. Особливо гостро стоїть питання зручності-швидкості при розробці додатків на мобільні пристрої. Слід зазначити, що в даний час 80% ринку мобільних операційних систем (ОС) займає платформа Android.

Аналіз останніх досліджень і публікацій. Рішенням проблеми зручності і швидкодії стала розробка механізму ORM (англ. Object-relational mapping, Об'єктно-реляційне відображення) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи "віртуальну об'єктну базу даних" [1–3]. На сьогоднішній день існують різні ORM-рішення, кожне з яких має свої сильні і слабкі сторони, а популярність залежить не тільки від вищевказаних показників, але і від успішності вирішення конкретних завдань, які ставляться перед розробниками.

Мета статті – провести порівняльний аналіз роботи найбільш поширених ORM-бібліотек для мобільних пристроїв в однакових умовах.

Виклад основного матеріалу

ORM-рішення для мобільної ОС Android

Використання стандартних методів управління даними при написанні програмного коду може бути не досить ефективним. Велика кількість різноманітних викликів та підготовчих етапів для кожної операції робить код перевантаженим і важким для читання. Ось тому для оптимізації рутинної частини цієї роботи по оголошенню полів і створенню структури бази з наступним швидким зверненнями до даних і були створені ORM-рішення. Розглянемо різні популярні рішення, якими найчастіше користуються на сьогоднішній день розробники мобільних додатків для платформи Android [1–4].

1) ActiveAndroid.

ActiveAndroid (AA) – бібліотека для швидкого і зручного опису невеликих і простих таблиць. Як і будь-яке інше ORM-рішення, AA спрямований на створення зв'язків між java-класами і таблицями баз даних, полями java-класу і набором стовпців таблиці [5]. Кожен рядок є об'єктом класу. Розглянемо більш докладно, як вирішується головне завдання ORM. Після певних дій по компіляція бібліотеки AA через

gradle і зміни файлу AndroidManifest.xml, бібліотекою можна користуватися в робочому проекті. Основною особливістю роботи АА є необхідність наслідувати клас бази даних від спеціального класу Model бібліотеки АА, а для оголошення таблиці і колонок використовувати спеціальні інструкції @Table (name = "") і @Column (name = "" [, unique = true]) відповідно. Name позначає ім'я таблиці або стовпця і дається відповідно до правил SQL. Варто відзначити, що АА визначає для елементів локальний ID на додаток до стандартного ID, який дається на сервері бази даних. Для того, щоб звернутися до цього локального ID, необхідно скористатися командою getId() (рис. 1).

```
import com.activeandroid.Model;
import com.activeandroid.annotation.Column;
import com.activeandroid.annotation.Table;

@Table(name = "Items")
public class Item extends Model {
    // This is the unique id given by the server
    @Column(name = "remote_id", unique = true, onUniqueConflict = Column.ConflictAction.REPLACE)
    public long remoteId;
    // This is a regular field
    @Column(name = "Name")
    public String name;
    // This is an association to another activeandroid model
    @Column(name = "Category", onUpdate = ForeignKeyAction.CASCADE, onDelete = ForeignKeyAction.CASCADE)
    public Category category;

    // Make sure to have a default constructor for every ActiveAndroid model
    public Item(){
       super();
    }

    public Item(int remoteId, String name, Category category){
        super();
        this.remoteId = remoteId;
        this.name = name;
        this.category = category;
    }
}
```

Рис. 1. Приклад створення таблиці в ActiveAndroid

Далі, після оголошення всіх необхідних класів і їх полів, можна звичним способом створювати нові таблиці за заданим шаблоном і заповнювати їх даними. Для збереження описаних даних в таблицю використовується команда імя_классу.save (). Для видалення даних використовується вивантаження даних з таблиці з подальшим видаленням командами імя_классу.load () і імя_классу.delete (), або подобою SQL-запиту (рис. 2).

```
// Create an item
Item item = new Item();
item.remoteId = 1;
item.category = restaurants;
item.name = "Outback Steakhouse";
item.save();

// Deleting items
Item item = Item.load(Item.class, 1);
item.delete();
// or with
new Delete().from(Item.class).where("remote_id = ?", 1).execute();
```

Рис. 2. Створення і видалення записів БД в ActiveAndroid

Вибірка даних здійснюється за допомогою об'єкта Select () з перерахуванням умов, дуже схожим з SQL-query (рис. 3).

До додаткових можливостей пакету відносяться: заповнення стандартного ArrayAdapter даними з бази даних шляхом простого додавання результату виходу query в адаптер; виклик для користувача SQL без необхідності повернення результату або з заповненням даними об'єкта типу List; заповнення ListView; використання даних з АА в якості Content Provider і т.д.

```
@Table(name = "Items")
public class Item extends Model {
    // ...
    public static List<Item> getAll(Category category) {
        // This is how you execute a query
        return new Select()
          .from(Item.class)
          .where("Category = ?", category.getId())
          .orderBy("Name ASC")
          .execute();
    }
}
```

Рис. 3. Вибірка даних з таблиці в ActiveAndroid

2) ORMLite.

ORMLite – це загальнодоступне ORM-рішення, на ОС Android є адаптацією для мобільних платформ свого старшого Java-фреймворка. Даний набір бібліотек для роботи з базою даних активно використовує анотації та успадкування функцій від різних стандартних бібліотек системи.

Після завантаження та додавання бібліотек ORMLite в свій Android-проект, для початку роботи потрібно прописати їх імпорт в Activity додатки. Для оголошення бази даних використовується анотація @DatabaseTable (tablename = ""), де tablename – конкретна назва створюваної таблиці. Після цього оголошення йде опис класу таблиці.

Кожен стовпець таблиці супроводжується анотацією @DatabaseField з подальшим описом типу змінної і її назви, які повинні бути private. Кожна анотація може супроводжуватися набором аргументів, але це не обов'язково. У разі відсутності таких їх значення будуть встановлені за замовчуванням. У класі обов'язково оголошення конструктора без аргументів. Далі можна прописувати різні методи класу, при виклику яких будуть повертатися ті чи інші дані з таблиці бази. Первинним ключем може служити будь-яке поле – досить вказати у нього id = true, але рекомендується зробити автогенероване значення id і поставити йому generatedId = true. ORMLite сама призначить йому унікальний номер. Також крім анотацій ORMLite можна використовувати розповсюджені javax.persistence анотації (рис. 4).

Існує кілька способів отримання доступу до даних БД за допомогою ORMLite в ОС Android.

Можна наслідувати кожну Activity від ORMLiteBaseActivity і тоді не доведеться стежити за життєвим циклом з'єднання з БД, але при цьому ми не зможемо отримати доступу з інших класів.

Найчастіше на практиці використовують другий підхід – за допомогою створення класу, який інстанціює (створює екземпляр класу) помічника в створенні та при роботі з базою даних.

```
@DatabaseTable(tableName = "goals")
public class Goal{

   public final static String GOAL_NAME_FIELD_NAME = "name";

   @DatabaseField(generatedId = true)
   private int Id;

   @DatabaseField(canBeNull = false, dataType = DataType.STRING, columnName = GOAL_NAME_FIELD_NAME)
   private String name;

  @DatabaseField(dataType = DataType.DATE)
   private Date lastEditDate;

   @DatabaseField()
   private String notes;

   public Goal(){
       scheduleList = new ArrayList<Shedule>();
       priorities = new ArrayList<PrioritySchedule>();
   }
}
```

Рис. 4. Опис таблиці в ORMLite

Відбувається це за допомогою створення в новому класі об'єкта класу DatabaseHelper (рис. 5) і опису декількох методів, які будуть брати, запускати і видаляти помічника.

```
public class HelperFactory{

   private static DatabaseHelper databaseHelper;

   public static DatabaseHelper getHelper(){
       return databaseHelper;
   }
   public static void setHelper(Context context){
       databaseHelper = OpenHelperManager.getHelper(context, DatabaseHelper.class);
   }
   public static void releaseHelper(){
       OpenHelperManager.releaseHelper();
       databaseHelper = null;
   }
}
```

Рис. 5. Реалізація класу-помічника в ORMLite

Звернення до помічника відбувається на початку і кінці життя додатку. Таким чином запобігають витоку пам'яті через незакрите з'єднання з БД. У свою чергу клас DatabaseHelper, що успадковує клас OrmLiteSqliteOpenHelper, відповідає за створення БД через отримання посилань на Data access objects (DAO).

Клас для роботи з DAO потрібно писати окремо, він буде успадковувати клас BaseDaoImpl (рис. 6).

```
public class RoleDAO extends BaseDaoImpl<Role, Integer>{

   protected RoleDAO(ConnectionSource connectionSource,
           Class<Role> dataClass) throws SQLException{
       super(connectionSource, dataClass);
   }

   public List<Role> getAllRoles() throws SQLException{
       return this.queryForAll();
   }
}
```

Рис. 6. Клас-обробник DAO

У цьому ж класі можна прописати обробку специфічних складних запитів до БД, можна також оновлювати і видаляти таблиці.

Також ORMLite підтримує роботу з вкладеними сутностями, тобто зі зв'язаними об'єктами.

Необхідно зауважити, що ORMLite, виходячи з офіційної документації, при доволі широкому функціоналі, не приносить великої зручності в роботі з базами даних, часом роблячи код нехай і більш структурованим, але достатньо заплутаним. Необхідність створювати кілька додаткових класів-помічників і постійна залежність одних класів від інших роблять цю реалізацію не найпривабливішою для розробників. Але, так як це одна з перших реалізацій ORM на Android, вона поки ще залишається однією з найпопулярніших.

3) GreenDAO.

GreenDAO вважають одним з найшвидших ORM-рішень серед всіх популярних бібліотек, але його освоєння є не дуже простим процесом. Спочатку, для успішної роботи GreenDAO потрібно створити окремий виконуваний Java (не Android) проект, який буде бібліотекою-генератором цього ORM. Після успішного його заповнення даними пакету, моделювання полів і виклику генерації коду можна приступати до використання бібліотеки в основному Android-проекті, попередньо прикріпивши її до нього.

GreenDAO оперує 4 основними класами, робота яких пов'язана і які є його основними інтерфейсами. DaoMaster – відправна точка для роботи з GreenDAO. Цей клас містить об'єкт бази даних SQLiteDatabase і управляє класами DAO для певного шаблону (schema). У ньому також містяться класи-помічники, взяті з SQLiteOpenHelper. DaoSession – управляє всіма доступними DAO-об'єктами певного шаблону. Доступ до них забезпечується вже прописаними в бібліотеці методами. Клас також містить основні методи для управління конкретними записами, такими як insert, load, update, refresh і delete. До всього, саме цей клас стежить за унікальністю необхідних записів. Клас DAOs – управляє властивостями записів. Клас Entities – управляє загальним видом записів і типами даних [6].

Для початкової ініціалізації роботи GreenDAO в коді необхідно прописати декілька об'єктів вищеописаних класів, за допомогою яких і буде відбуватися управління базою (рис. 7).

```
helper = new DaoMaster.DevOpenHelper(this, "notes-db", null);
db = helper.getWritableDatabase();
daoMaster = new DaoMaster(db);
daoSession = daoMaster.newSession();
noteDao = daoSession.getNoteDao();
```

Рис. 7. Ініціалізація роботи GreenDAO

Далі для роботи з БД через GreenDAO необхідно визначити структуру тих даних, якими вона буде заповнюватися. Дана ORM слідує своєї певній структурі, яка вже визначена в Java-коді бібліотеки. Тому потрібно лише створити Java-проект, який би базувався на проекті DaoExampleGenerator, який можна завантажити на офіційному сайті (рис. 8).

Основні сутності, якими оперує дана схема, це:

– шаблон (Schema). Перший об'єкт, що визначається в класі структури БД, прототип загального об'єкта бази даних. На шаблоні розміщуються всі записи (Entities).

```
Schema schema = new Schema(1, "de.greenrobot.daoexample");

Entity note= schema.addEntity("Note");
note.addIdProperty();
note.addStringProperty("text").notNull();
note.addStringProperty("comment");
note.addDateProperty("date");

new DaoGenerator().generateAll(schema, "../DaoExample/src-gen");
```

Рис. 8. Загальна структура, представлена в DaoExampleGenerator

Шаблон містить в собі 2 прапора, особливості яких ще не задокументовані творцями пакета. Виклик шаблону відбувається наступним чином:

Schema schema = new Schema (1, "de.greenrobot.daoexample");

– записи (Entities). Є прототипами таблиць бази даних з декількома методами і параметрами. Методом add_tip_Property можна додавати різні стовпці з їх назвами. Виклик Entity:

Entity user = schema.addEntity ("User");

Основними функціями GreenDAO є: створення зв'язків між елементами; успадкування від не Entity класів; інтерфейси; виклик генерації коду певних шаблонів-записів; робота з різноманітними запитами за допомогою класу QueryBuilder; наявність власних list-методів і т.д. (рис. 9).

```
Query query = userDao.queryBuilder().where(
Properties.FirstName.eq("Joe"), Properties.YearOfBirth.eq(1970))
.build();
List joesOf1970 = query.list();

query.setParameter(0, "Maria");
query.setParameter(1, 1977);
List mariasOf1977 = query.list();
```

Рис. 9. Створення query-запросу в GreenDAO

Для додавання полів в клас-сутності, необхідно помістити їх в блоки коментарів // KEEP FIELDS ... // KEEP FIELDS END. Пакет досить різноманітний і функціональний, але при всій його швидкості йому не вистачає ґрунтовної документації, багато функцій доводиться досліджувати "на дотик". До того ж, необхідна певна підтримка коду і не дуже вдало реалізоване API може знизити придатність даного ORM-пакета.

4) SugarORM.

Для початку роботи з SugarORM необхідно запустити команду compile 'com.github.satyan: sugar: 1.4' і прописати SugarApp в AndroidManifest. Це крайнє в списку розглянутих ORM рішення відрізняється від всіх попередніх тим, що воно розроблялося спочатку тільки для Android. Ніяких портів з інших платформ, ніяких додаткових Java-бібліотек. SugarORM є дуже простим, але з обмеженим функціоналом рішенням.

Для створення таблиці потрібно просто створити клас і успадковувати його від класу SugarRecord, або використовувати анотацію @Table, але в цьому випадку необхідно самому визначити поле id типу private. Далі, як і в будь-якому класі, слід визначити конструктор. Об'єкти таблиці визначаються як звичайні поля класу. Функції save, delete і findById максимально спрощують роботу з таблицею і внесенням в неї даних. Створення зв'язку також просте – після визначення другого класу необхідно в першому створити запис типу створеного другого класу (рис. 10).

Рис. 10. Основні операції в SugarORM

Також в бібліотеці присутні функції QueryBuilder і міграції БД. На цьому функціонал даної ORM закінчується. Ніяких надмірностей, тільки те, що і вимагала він неї початкова задача, що можна вважати як плюсом, так і мінусом даної бібліотеки. В Інтернеті його називають "слабким клоном ActiveAndroid".

Порівняння ORM-бібліотек

Для порівняння розглянутих ORM-рішень для управління базами даних в Android скористаємося результатами тестів, проведених компанією Sebbia.

За основну оцінку при тестуванні бібліотек були взяті не особливості кожного з ORM, а швидкість виконання аналогічних завдань, поставлених перед кожною з них. Було вирішено, що не так важливо, яким чином кожен з представлених проектів справляється зі своїм завданням в плані лаконічності коду і прийомів роботи з базою, а важливо те, наскільки швидко це відбувається, і як можна порівняти швидкість роботи проекту зі швидкістю обробки даних при використанні стандартних засобів Android.

Отже, проведений тест перевіряє, як швидко та чи інша бібліотека може зберігати тестові сутності в ході SQLite транзакції і здійснювати зворотну їй операцію. У новостворену базу даних додаються 1000 тестових об'єктів, які після очищення кешу в пам'яті ORM зчитуються і перевіряються на "коректність" даних. Кожна бібліотека (ORM та стандартні методи) перевіряється 10 разів, за кінцевий результат береться середній час виконання операцій. Тестові об'єкти складаються з двох текстових полів фіксованої довжини, одного поля дати і одного масиву байт, отриманого з серіалізуємого (Serializable) об'єкта (рис. 11).

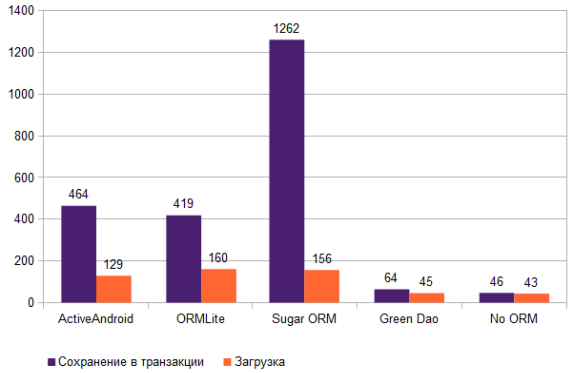

Рис. 11. Діаграма результатів порівняння рішень управління БД

Проект і всі версії використаних при тесті бібліотек можна завантажити на Github-сторінці компанії.

Попередньо беручи до уваги особливості всіх обраних для тесту проектів можна було вже зробити певні висновки щодо можливих результатів тесту. Практичні ж дослідження тільки підтверджують їх. Найшвидшим виявилося рішення на основі стандартних методів Android (що використовує SQLiteOpenHelper і скомпільовані SQLiteStatement). Після нього, не сильно відстаючи, GreenDAO, завдяки своєму підходу кодогенерації через окремий клас, скомпільовані SQLiteStatement і при оптимізації процесу роботи з БД. По ньому йдуть ActiveAndroid, ORMLite, і в самому кінці, з найгіршим результатом, SugarORM. Останній за всіма параметрами значно програє своїм старшим побратимам. Причому, як уже було видно з огляду, його програш по швидкості доповнюється дуже урізаним функціоналом. Що ж стосується зручності кожного з пакетів, то тут кожен розробник вибирає сам, з чим йому приємніше працювати, будь то успадковані бібліотеки, анотація або кодогенерація, а також в залежності від поставлених завдань. Однак, кінцевий користувач середнього сегмента, на якого спрямована більша частина розроблюваних додатків, швидше за все, не побачить різниці при роботі швидкого або повільного ORM. Крім нього, в додатку завжди знайдеться безліч речей, які будуть навантажувати апарат сильніше, ніж робота БД. Тому в більшості випадків вибір зручного інструменту лягає на плечі того, хто пише код.

Висновки

Зберігання даних в базах даних – один з найбільш поширених і зручних способів роботи з даними, незалежно від платформи. У кожній платформі є свої засоби створення і управління базами даних, і ОС Android не є винятком. В її складі є вже готові інструменти з управління базами даних SQLite, які надають великі можливості доступу до даних будь-яким зручним способом. Мінусом цих засобів є їх складність для розуміння, широта написання коду при здійсненні специфічних операцій і загальна його не лаконічність. Тому було прийнято рішення створити технологію, яка була б прошарком між програмістом і базою даних, надаючи першому можливість у звичній об'єктно-орієнтованої манері працювати з базою, при цьому не спотворюючи її загальну структуру і принципи зберігання інформації. Безліч різних ORM-рішень, які існують в Інтернеті і повсюдно використовуються, якісно спрощують роботу з БД, перетворюючи її в оперування об'єктами, полями і методами класів, що представляють структуру таблиць бази. Кожне рішення по-своєму зручно, але має свої недоліки, тому кожен програміст має право сам вибирати, яким саме йому користуватися або не користуватися взагалі.

Список літератури

1. Sommerhoff Peter. Kotlin for Android App Development / Peter Sommerhoff. – Addison-Wesley Professional, 2018. – 432 p.
2. Späth Peter. Pro Android with Kotlin: Developing Modern Mobile Apps / Peter Späth. – Apress, 2018. – 485 p.
3. Stroud Adam. Android Database Best Practices / Adam Stroud. – Addison-Wesley Professional, 2016. – 288 p.
4. Save data in a local database using Room [Electronic resource]. – Available at: https://developer.android.com/training/data-storage/room/.
5. ActiveAndroid Guide [Electronic resource]. – Available at: https://guides.codepath.com/android/activeandroid-guide.
6. Introduction to GreenDAO [Electronic resource]. – Available at: http://greendao-orm.com/documentation/introduction/.
7. Ешин С.С. Сравнительный анализ API ORM-библиотек для платформы Android / С.С. Ешин // Информатика: проблемы, методология, технологии, 2015.
8. Titze D. Preventing Library Spoofing on Android / Titze, D., & Schutte, J. // 2015 IEEE Trustcom/BigDataSE/ISPA. https://doi.org/10.1109/trustcom.2015.494.
9. Petrucha J. Design of database applications in mobile devices with OS Android / J. Petrucha // 16th International Multidisciplinary Scientific GeoConference SGEM2016, Informatics, Geoinformatics and Remote Sensing, 2016. https://doi.org/10.5593/sgem2016/b21/s07.023.
10. Есин В.И. Возможности различных схем баз данных по представлению и работе с данными / В.И. Есин // Системи обробки інформації. – 2012. – № 2(100). – С. 195-198.
11. Третяк В.Ф. Использование технологии репликации в системе управления распределенными базами данных / В.Ф. Третяк, И.Е. Кужель, В.М. Приходько // Збірник наукових праць Харківського національного університету Повітряних Сил. – 2010. – № 2(24). – С. 109-114.
12. Танянский С.С. Условия сохранения корректного состояния базы данных при модификации структуры отношений / С.С. Танянский // Системи обробки інформації. – 2012. – № 9(107). – С. 211-214.

13. Nooh Taha Nasif Development of a hybrid mobile application for android with the function of processing and scanning of bar-codes / Nasif Nooh Taha // Системи озброєння і військова техніка. – 2017. –№. 2(50). – С. 159-167.

14. Дубровський, М.С. Comparative overview of basic cybervulnerabilities of mobile applications for Android operating system / М.С. Дубровський, С.Г. Семенов// Системи обробки інформації. – 2016. – № 3(140). – С. 18-20.

15. Ільїна І.В. Аналіз можливостей впровадження спеціального програмного забезпечення в ОС Android / І.В. Ільїна, О.С. Петленко // Системи озброєння і військова техніка. – 2015. – № 2(42). – С. 90-91.

References

1. Sommerhoff, Peter (2018), *Kotlin for Android App Development*, Addison-Wesley Professional, 432 p.
2. Späth, Peter (2018), *Pro Android with Kotlin: Developing Modern Mobile Apps*, Apress, 485 p.
3. Stroud, Adam (2016), *Android Database Best Practices*, Addison-Wesley Professional, 288 p.
4. *Save data in a local database using Room*, available at: https://developer.android.com/training/data-storage/room/.
5. *ActiveAndroid Guide*, available at: https://guides.codepath.com/android/activeandroid-guide.
6. *Introduction to GreenDAO*, available at: http://greendao-orm.com/documentation/introduction/.
7. Eshin, S.S. (2015), "Sravnytelnyi analyz API ORM-byblyotek dlia platformy Android" [Comparative analysis of API ORM-libraries for the Android platform], *Informatyka: problemy, metodolohyia, tekhnolohyy*.
8. Titze, D. and Schutte, J. (2015), Preventing Library Spoofing on Android, *2015 IEEE Trustcom/BigDataSE/ISPA*. https://doi.org/10.1109/trustcom.2015.494.
9. Petrucha, J. (2016), Design of database applications in mobile devices with OS Android, *16th International Multidisciplinary Scientific GeoConference SGEM2016, Informatics, Geoinformatics and Remote Sensing*. https://doi.org/10.5593/sgem2016/b21/s07.023.
10. Esin, V.I. (2012), "Vozmozhnosti razlichnykh skhem baz dannykh po predstavleniiu i rabote s dannymi" [Possibilities of various schemes of databases on presentation and work with data], Information Processing Systems, Vol. 2(100), pp. 195-198.
11. Tretiak, V.F., Kuzhel, I.E. and Prikhodko, V.M. (2010), "Ispolzovanie tekhnologii replikatsii v sisteme upravleniia raspredelennymi bazami dannykh" [Use of technology of replication in control the system by the distributed databases], Scientific Works of Kharkiv National Air Force University, Vol. 2(24), pp. 109-114.
12. Tanianskii, S.S. (2012), "Usloviia sokhraneniia korrektnogo sostoianiia bazy dannykh pri modifikatsii struktury otnoshenii" [Tanianskii, S.S. (2012), "Usloviia sokhraneniia korrektnogo sostoianiia bazy dannykh pri modifikatsii struktury otnoshenii", Information Processing Systems, Vol. 9(107), pp. 211-214.], Information Processing Systems, Vol. 9(107), pp. 211-214.
13. Nooh Taha Nasif, (2017), Development of a hybrid mobile application for android with the function of processing and scanning of bar-codes, Systems of Arms and Military Equipment, No. 2(50), pp. 159-167.
14. Dubrovskii, M.S. and Semenov, S.G. (2016), Comparative overview of basic cybervulnerabilities of mobile applications for Android operating system, Information Processing Systems, Vol. 3(140), pp. 18-20.
15. Ilina, I.V. and Petlenko, O.S. (2015), "Analiz mozhlyvostei vprovadzhennia spetsialnoho prohramnoho zabezpechennia v OS Android" [Analysis of the possibility of introducing special software in the OS Android], Systems of Arms and Military Equipment, No. 2(42), pp. 90-91.

Надійшла до редколегії 20.09.2018
Схвалена до друку 5.11.2018

Відомості про авторів:

Федорченко Володимир Миколайович
кандидат технічних наук доцент
доцент кафедри
Харківського національного
економічного університету ім. С. Кузнеця,
Харків, Україна
https://orcid.org/0000-0001-7359-1460

Сєвєрінов Олександр Васильович
кандидат технічних наук доцент
доцент кафедри
Харківського національного
університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0002-6327-6405

Родіонов Сергій Вікторович
кандидат технічних наук доцент
доцент кафедри
Українського державного
університету залізничного транспорту,
Харків, Україна
https://orcid.org/0000-0002-9511-3950

Information about the authors:

Volodymyr Fedorchenko
Candidate of Technical Sciences Associate Professor
Senior Lecturer
of Simon Kuznets Kharkiv National
University of Economics,
Kharkiv, Ukraine
https://orcid.org/0000-0001-7359-1460

Oleksandr Sievierinov
Candidate of Technical Sciences Associate Professor
Senior Lecturer
of Kharkiv National University
of Radio Electronics,
Kharkiv, Ukraine
https://orcid.org/0000-0002-6327-6405

Sergii Rodionov
Candidate of Technical Sciences Associate Professor
Senior Lecturer
of Ukrainian State University
of Railway Transport,
Kharkiv, Ukraine
https://orcid.org/0000-0002-9511-3950

АНАЛИЗ ORM-БИБЛИОТЕК ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

В.Н. Федорченко, А.В. Северинов, С.В. Родионов

Предметом исследования в статье является процесс разработки мобильных data-ориентированных приложений для платформы Android. Статья посвящена детальному анализу современных методов управления базами данных в ОС Android. Цель работы – оценка эффективности различных библиотек, обеспечивающих взаимодействие приложения с БД и реализующих ORM-технологию программирования. В статье решаются следующие задачи: рассмотрение и анализ наиболее распространённых ORM-библиотек сторонних разработчиков для платформы Android. Основное содержание исследования составляет сравнительный анализ наиболее распространённых ORM-библиотек и результатов их работы в одинаковых условиях. Получены следующие результаты: проанализирована сложность использования API библиотек при программной реализации ORM-модели. Приведены результаты исследования быстродействия выполнения CRUD-операций в мобильном приложении с применением рассмотренных библиотек. Выводы: проведенный анализ позволяет повысить эффективность процесса разработки и масштабирования мобильных приложений в части управления базами данных для платформы Android.

Ключевые слова: *база данных, Android, ORM, ORM-библиотека, ActiveAndroid, ORMLite, GreenDAO, SugarORM.*

ANALYSIS OF ORM-LIBRARIES FOR ANDROID OPERATING SYSTEM

V. Fedorchenko, O. Sievierinov, S. Rodionov

The Android operating system is based on the Linux kernel and Google's own Java Virtual Machine implementation. For most mobile applications, especially business applications, a local relational database is used for data storage. The most common technology for organizing interaction with the database is ORM (Object-Relational Mapping) - a programming technology that connects the database with the concepts of object-oriented programming languages, creating a "virtual object database". ORM solutions are used to optimize the work of writing software code when creating a database. The issue to be research in the article is the development process of the data-driven mobile applications for android platform. Also the article dealt with the detailed analysis of the modern database management techniques for Android. The ORM analysis focuses on the efficiency evaluation of the different libraries that provide the database and application interaction. Under the publication, the following tasks are carried out: the analysis of the most widely used ORM-libraries, built by the third-party developers for OS Android, namely: ActiveAndroid, ORMLite, GreenDAO and SugarORM. The essence of the study is a comparison of the most common ORM-libraries and the results of its performance. To compare the considered ORM solutions for database management in Android, the results of tests conducted by Sebbia are used. For the basic evaluation, when testing libraries, not the features of each ORM were taken, but the speed of performing similar tasks assigned to each of them.The following results have been achieved: It was examined the usage complexity of API-libraries with software implementation of the ORM model and studied the operating speed of the application's CRUD-operations with the ORM above mentioned as well. The best results for the performance of standard CRUD database operations, which corresponds to the use of standard Android tools, are shown by the GreenDAO library. Each ORM solution is convenient in its own way, but has its disadvantages, so each programmer has the right to choose exactly which ORM solution to use it. Conclusions: the analysis allows to improve the efficiency of the development and scaling of mobile applications in terms of database management for the Android platform.

Keywords: *database, Android, ORM, ORM-library, ActiveAndroid, ORMLite, GreenDAO, SugarORM.*