УДК 004.051

МЕТОДИ ОПТИМІЗАЦІЇ RECYCLERVIEW НА ANDROID ПРИСТРОЯХ

Мовчан О. Ю.[1], студент; **Плахтій Є. Г.**[2], ст. виклад.
Придніпровська державна академія будівництва та архітектури
[1] movchan.olexandr@gmail.com; [2] plakhtii.yevhen@pdaba.edu.ua

Постановка проблеми. Використання RecyclerView на платформі Android регулярно викликає проблеми з продуктивністю, через що застосунок починає повільно працювати. Особливо це стосується, коли верстка ViewHolder складна і не оптимізована.

Мета дослідження. Дослідити методи оптимізації і покращення роботи списків при високому навантаженні. Оскільки майже всі мобільні застосунки потребують використання списків – питання оптимізації їх роботи досить актуальне і поширене.

Результати дослідження. Для дослідження було взято xml з 19 дочірніми елементами, який мав також кілька ступенів вкладеності. В першу чергу було досліджено вплив антипаттернів. Сам список був «безкінечним», з динамічним підвантаженням елементів по 20 шт., коли користувач догортав до низу.

1. Використання RecyclerView всередині ScrollView [1]. В такому випадку ViewHolder не перевикористовується, і система рендерить весь список. Якщо RecyclerView «безкінечний», і підвантаження елементів відбувається динамічно, то з кожним додаванням елементів відбувається перемальовування всього списку, що суттєво навантажує систему. В результаті після відмальовки 100−120 елементів починає ловитися помилка «Application Not Responding» при додаванні або видаленні елементів.

2. Використання великої вкладеності компонентів в xml-розмітці. Це призводить до збільшення кількості операцій і часу вираховування розмірів кожного з компонентів. Після перекомпонування xml-розмітки, і винесення компонентів в один рівень вкладеності, приріст продуктивності склав 14 %.

3. Використання компоненту CardView. Цей компонент визнаний не оптимізованим, тож використання не рекомендоване. Для експерименту, після відмови від CardView, і заміни його стандартним View з спеціальним фоном, приріст продуктивності склав 4 %.

Наступним кроком були розглянуті додаткові способи покращення продуктивності.

1. Заміна Visibility.Gone на Alpha 0, коли в залежності від стану потрібно показати, або приховати елемент, при тому не зміщуючи інші. Таким чином, при зникненні View не доведеться перераховувати всі розміри. Маючи 8 таких компонентів, пришвидшення перемальовки при оновленні даних склало 11 %.

2. Використання фіксованих розмірів замість wrap_content. Якщо ми точно знаємо висоту елемента (наприклад, TextView з текстом в 1 рядок), або його ширину, то можна спробувати одразу їх задати. Таким чином спрощується обчислення розмірів і час рендерингу. Приріст невеликий, 3 %, але він є.

3. Попереднє завантаження картинок. В даному експерименті використовувалось завантаження картинок по url. Після заміни підходу на попереднє завантаження, і встановлення картинки з пам’яті телефону, по відчуттях список став працювати краще. Але об’єктивного способу заміряти різницю не було, тож цей пункт не включаємо в загальний підрахунок покращення.

4. Відмова від xml [2]. Для того, щоб створити View, система має розпарсити xml, а потім створити дочірні компоненти програмним методом. Якщо дати системі змогу

цього не робити, і зробити це самим – це також пришвидшить час створення View. По замірах – це найефективніший метод, час створення ViewHolder скоротився на 32 %. Із недоліків – цей код стає складніше підтримувати.

В таблиці можна подивитися і порівняти результати оптимізацій. Середній час вираховувався із 100 операцій.

Таблиця

Порівняння результатів оптимізації списків на Android

Метод оптимізації	onCreateViewHolder			onBindViewHolder			Загальний приріст, %
	Час до (мс)	Час після (мс)	Приріст, %	Час до (мс)	Час після (мс)	Приріст, %	
Зменшення вкладеності xml	130,024	116,093	12	11,808	11,576	2	14
Відмова від CardView	116,093	110,565	5	Суттєво не впливає			5
Заміна Visibility.Gone на Alpha	Суттєво не впливає			11,576	10,429	11	11
Використання фіксованих розмірів	110,565	107,345	3	Суттєво не впливає			3
Відмова від xml	107,375	73,015	32	Суттєво не впливає			32

Дані отримані з девайсу Redmi 12 на Android 14. Була також додатково протестовано на інших пристроях, в тому числі: Xiaomi Redmi Note 4 (Android 7), Samsung M14 (Android 13), Android Emulator API34. На всіх пристроях дані корелюються, і приріст продуктивності у відсотках близький до наведених даних у таблиці.

Висновки. Найбільш ефективно себе показала відмова від xml в цілому, але, як видно із результатів, кожна дрібниця, яка впливає на 3−5 %, в купі з іншими може зменшити або покращити продуктивність на 30−35 %.

Список використаних джерел

1. Android Programming: The Big Nerd Ranch Guide, 3nd Edition. Big Nerd Ranch. Guides, Bill Phillips, Kristin Marsicano, Chris Stewart. 2015.

2. Create dynamic lists with RecyclerView [Електронний ресурс]. URL: https://developer.android.com/develop/ui/views/layout/recyclerview