УДК 004.896.032.26(043.2)

І. Р. Арсенюк
В. В. Сабашна

ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ РЕСУРСІВ АКУМУЛЯТОРА ПРИ ВІДСТЕЖЕННЯ ГЕОЛОКАЦІЇ В МОБІЛЬНОМУ ДОДАТКУ REACT NATIVE ДЛЯ ОПЕРАЦІЙНИХ СИСТЕМ ANDROID ТА IOS

Вінницький національний технічний університет;

Анотація

В матеріалах розглядається спосіб покращення продуктивності акумулятора при розробці додатків для відстеження геолокації в операційній системі Android та IOS .

Ключові слова: акумулятор, ресурси, геолокація, місцезнаходження, Android.

Abstract

The materials consider a way to improve battery performance in the development of applications for tracking geolocation in the Android and IOS operating system.

Keywords: battery, resources, location, geolocation, Android.

Вступ

Ефективне використання ресурсів акумулятора є критично важливим аспектом для розробників мобільних додатків, особливо тих, які використовують геолокацію. Це стає актуальним як для платформи Android, так і для iOS. У цій роботі ми розглянемо найкращі практики використання служб локації, спрямовані на оптимізацію енергоспоживання, щоб забезпечити більш тривалу роботу мобільних пристроїв.

Результати дослідження

Спочатку, визначимо зміни, які були внесені у обмеження фонового розташування в Android 8.0 доставляється лише кілька разів на годину.

1. Збір місцевого розташування зменшується, а місце розташування обчислюється та доставляється лише кілька разів на годину.
2. Геофонічна чутливість змінюється від десятків секунд до приблизно двох хвилин. Ця зміна помітно покращує продуктивність акумулятора - до 10 разів краще на деяких пристроях [1].

Далі описано перелік найкращих практик для покращення продуктивності акумулятора на пристрої при розробці додатків для відстеження геолокації.

Видалення оновлення місцеположення

Поширеним джерелом непотрібного розряду акумулятора є неможливість видалення оновлень місцеположення, коли вони більше не потрібні. Це може статися, наприклад, коли методи життєвого циклу активності onStart() або onResume() містять виклик requestlocationUpdates() без відповідного виклику для видалення LocationUpdates() у методах життєвого циклу onPause() або onStop().

Встановлення тайм-аутів

Для захисту від розряду акумулятора можливе використання тайм-ауту, коли оновлення місцеположення повинні припинитися. Час очікування гарантує, що оновлення не триватиме нескінченно довго, і він захищає додаток у сценаріях, коли оновлення запитуються, але не видаляються (наприклад, через помилку в коді).

Для запиту в fused location provider кращим рішенням буде додавання таймаут, викликавши setExpirationDuration(), який отримує параметр, який представляє час у мілісекундах з моменту останнього виклику методу. Ви також можете додати тайм-аут, викликавши setExpirationTime(), який отримує параметр, який представляє час закінчення в мілісекундах з моменту останнього завантаження системи.

Пакетні запити

Для всіх випадків використання додатку не в передньому плані(фоновий режим) кращою практикує є виконання декількох запитів одночасно. Можна використовувати метод setInterval(), щоб вказати інтервал, через який буде обчислено розташування. Потім методом setMaxWaitTime(), щоб встановити інтервал, в якому розташування отримається додатком. Значення, передане методу setMaxWaitTime(), має бути кратним значенням, переданим методу setInterval().

Використання пасивного оновлення місцеположення

У фонових випадках використання корисно зменшити оновлення місцеположення. Ліміти Android 8.0 застосовують цю практику, але програми, що працюють на старих пристроях, повинні намагатися максимально обмежити розташування фону.

Під час пасивного споживання місцеположення не виникає витрата акумулятора, але, потрібно бути обережним у випадках, коли отримання даних про місцеположення викликає дорогі операції з процесором або введенням / виведенням. Щоб мінімізувати витрати акумулятора, інтервал, визначений у setFastestInterval(), не повинен бути занадто малим [2].

Зменшення точності даних про розташування та зменшення частоти збору даних про розташування та збільшення затримки швидкості оновлення даних. Ви можете вказати точність розташування за допомогою методу setPriority(), передаючи одне з наступних значень в якості аргументу: PRIORITY_HIGH_ACCURACY, PRIORITY_BALANCED_POWER_ACCURACY та PRIORITY_LOW_POWER

Дані практики оптимізації використання ресурсів акумулятора передбачають використання API локальних служб Google, які пропонують більш високу точність і накладають легше навантаження на акумулятор, ніж API-файли розташування.

Стосовно операційної системи IOS діють практичні ті ж самі поради, але з використанням можливостей мов програмування Swift та Objective-C. Також можна зазначити додаткові методи оптимізації використання акумулятора при використанні локації девайсу для IOS:

1. Увімкніть служби місцезнаходження лише тоді, коли вони потрібні. Потім залиште їх увімкненими просто достатньо довго, щоб отримати фіксацію місцезнаходження, і знову вимкніть їх. Якщо користувач не знаходиться в рухомому транспортному засобі, поточне місцезнаходження не повинно змінюватися достатньо часто, щоб стати проблемою. Ви завжди можете знову увімкнути служби місцезнаходження пізніше, якщо вам потрібне інше оновлення [3].
2. Якщо вашому додатку не надходять оновлення з очікуваною точністю, йому слід перевірити отримані оновлення та визначити, чи покращується точність часом, чи залишається приблизно такою ж. Якщо точність не покращується, можливо, потрібний рівень точності просто недоступний на даний момент. У цьому випадку припиніть оновлення місцезнаходження і спробуйте знову пізніше, щоб ваш додаток не продовжував займати енергію місцезнаходження [4].
3. Переконайтеся, що властивість pausesLocationUpdatesAutomatically об'єкта менеджера місцезнаходження встановлена в YES, щоб допомогти економити енергію. Встановіть властивість activityType, щоб сповістити Core Location, який тип діяльності з місцезнаходження виконує ваш додаток у конкретний момент часу, наприклад, якщо ваш додаток виконує відстеження фітнесу або автомобільну навігацію. Дивіться CLActivityType для списку типів діяльності. Вказання цих налаштувань допомагає менеджеру місцезнаходження визначити найбільш відповідний час для виконання оновлень місцезнаходження. Наприклад, оновлення місцезнаходження у фоновому режимі можуть автоматично призупинятися, якщо система визначить, що користувач не рухається [4].
4. Відкладіть оновлення місцезнаходження при роботі в фоновому режимі. Ви можете використовувати deferredLocationUpdatesAvailable, щоб визначити, чи підтримує пристрій відкладені оновлення місцезнаходження. Щоб відкласти оновлення, викличте метод allowDeferredLocationUpdatesUntilTraveled:timeout: об'єкта менеджера місцезнаходження і передайте йому відстань і час, який може пройти, перш ніж буде отримано наступне оновлення місцезнаходження. Цей метод зазвичай викликається в методі делегата locationManager:didUpdateLocations:, щоб знову відкласти, якщо це необхідно, коли отримано відкладене оновлення місцезнаходження [3].

Висновки

У цій роботі ми детально розглянули найкращі практики використання служб локації на платформах Android та iOS з метою оптимізації ресурсів акумулятора. Ефективне управління енергоспоживанням стає ключовим чинником для забезпечення тривалої роботи мобільних пристроїв, особливо у випадку

додатків, які активно використовують геолокацію. Шляхом впровадження рекомендацій, наведених у цій роботі, розробники можуть підвищити ефективність своїх додатків, забезпечуючи одночасно задоволення потреб користувачів та збереження заряду батареї їхніх пристроїв.

.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. developer.android: [Електронний ресурс] - режим доступу: https://developer.android.com/guide/topics/location/battery.
2. Muthumurugesan D., Nalini S., Vinodini R. Smart Way to Track the Location in Android Operating System/ Muthumurugesan D., Nalini S., Vinodini R. – IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661
3. developer.apple: [Електронний ресурс] - режим доступу: https://developer.apple.com/documentation/Performance/Energy/LocationBestPractices
4. Abdul Ali Bangash, Daniil Tiganov, Karim Ali Abram Hindle – 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Сабашна Вікторія Валеріївна — студентка групи 1кн-20б, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, Вінниця, e-mail: vikasab77@gmail.com

Науковий керівник: ***Арсенюк Ігор Ростиславович*** — кандидат технічних наук, доцент кафедри “Комп’ютерних наук”, Вінницький національний технічний університет, м. Вінниця