Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота

з дисципліни: «Інтеграційні програмні системи »

Виконали:
студенти 4 курсу
ФІОТ гр. ІО-41
Дайко І. В.
Калитенко А. О.
Курганський С. С.
Назарівський Д. В.

Київ 2017

1. Короткий опис проекту

Реалізованим проектом у даному курсі є Android додаток, що дозволяє легко та швидко отримувати та переглядати відеозаписи з популярного відеохостингу Vidme. Додаток надає доступ до трьох типів стрічок відеозаписів: Популярні, Нові та Відеозаписи підписок користувача (рис.1). Останню стрічку може переглядати лише зареєстрований користувач.

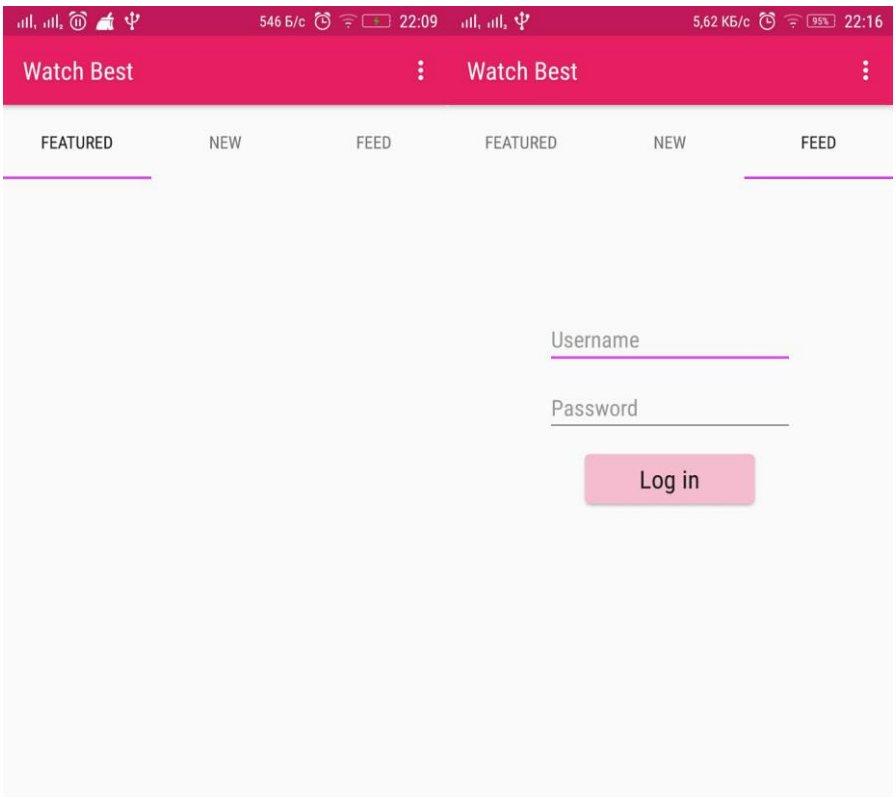

Рис.1 Скріншоти інтерфейсу додатку

Робота з сервером відбувається через API Vidme за допомогою бібліотек Retrofit та RxJava. Для парсингу JSON рядка використовується бібліотека Gson.

Для авторизації користувач повинен бути зареєстрований на сайті https://vid.me/. У випадку успішної авторизації, на девайсі в приватному сховищі додатку зберігається токен, отриманий від серверу. При наступному запуску додатку користувачеві вже не потрібно вводити логін та пароль, доки не буте натиснута кнопка Logout.

2. Система автоматизації збірки

У якості системи для автоматизації збірки проекту використовується Gradle - система, яка далі розвиває принципи, закладені в Apache Ant та Apache Maven і використовує предметно-орієнтовану мову (DSL) на основі мови Groovy замість традиційної XML-подібної форми представлення конфігурації проекту. На відміну від Apache Maven, заснованого на концепції життєвого циклу проекту, і Apache Ant, в якому порядок виконання задач (targets) визначається відносинами залежностей (depend-on), Gradle використовує спрямований ациклічний граф для визначення порядку виконання задач.

Gradle був розроблений для багатопроектних збірок, які розраховані на подальше розширення і підтримує інкрементальні збірки, визначаючи, які компоненти дерева збірки не змінилися і які завдання, залежать від цих частин, не вимагають перезапуску.

Основні плагіни призначені для розробки та розгортання програм на мовах Java, Groovy, Scala та ін. Для збірки Android додатків також призначений спеціальний плагін, що розробляється разом з Android SDK. Плагін для Android для Gradle працює з набором інструментів для створення процесів та настроюваних параметрів, специфічних для створення та тестування додатків для Android. Gradle та плагін Android для Gradle працюють незалежно від Android Studio. Це означає, що можливо створювати програми для Android з Android Studio, командного рядк або на машинах, де Android Studio не встановлений (наприклад, безперервні сервери інтеграції).

Кожна конфігурація збірки може визначати власний набір коду та ресурсів, одночасно повторно використовувати частини, загальні для всіх версій одного додатку.

Gradle не має власних сховищ і як джерело залежностей використовує Maven- та Ivy-репозиторії. В останніх версіяї плагіну основні репозиторії підключаються командами google() та jcenter().

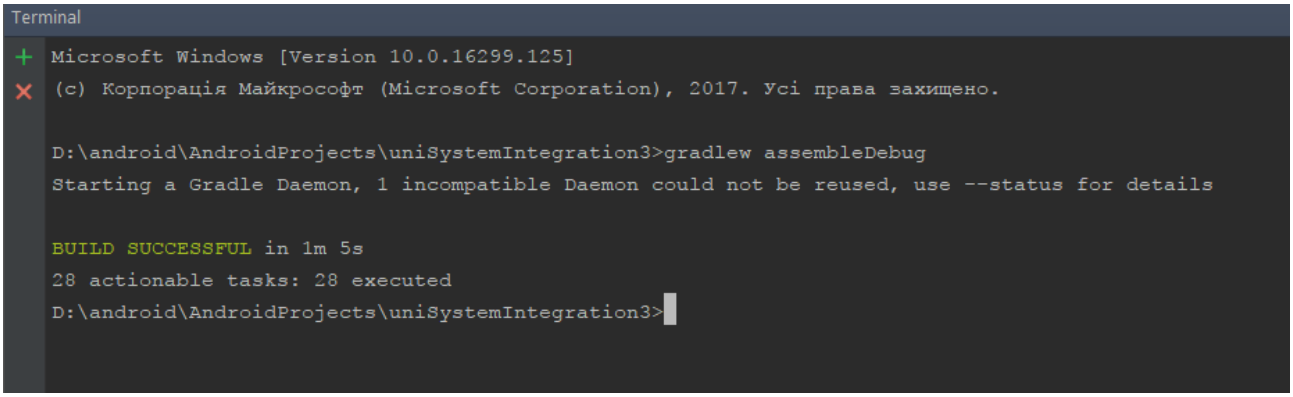

Рис.2 Результат збірки проекту за допомогою Gradle

3. Безперервна інтеграція

Безперервна інтеграція - це практика розробки програмного забезпечення, яка полягає в об'єднанні робочих копій у спільну основну гілку розробки та виконанні автоматизованих збірок проектів для швидкого виявлення потенційних дефектів та вирішення інтеграційних проблем. Перехід до безперервної інтеграції дозволяє знизити трудомісткість інтеграції та зробити її більш прогнозованою за рахунок швидкого виявлення та усунення помилок і, але основною перевагою є скорочення вартості виправлення дефектів за рахунок раннього їх виявлення.

Для реалізації безперервної інтеграції використовувався розподілений веб-сервіс Travis CI (рис.3).

На Travis CI виконуються наступні задачі:

./gradlew test – запускає усі тести проекту;

./gradlew lint – запускає статичний аналізатор коду.

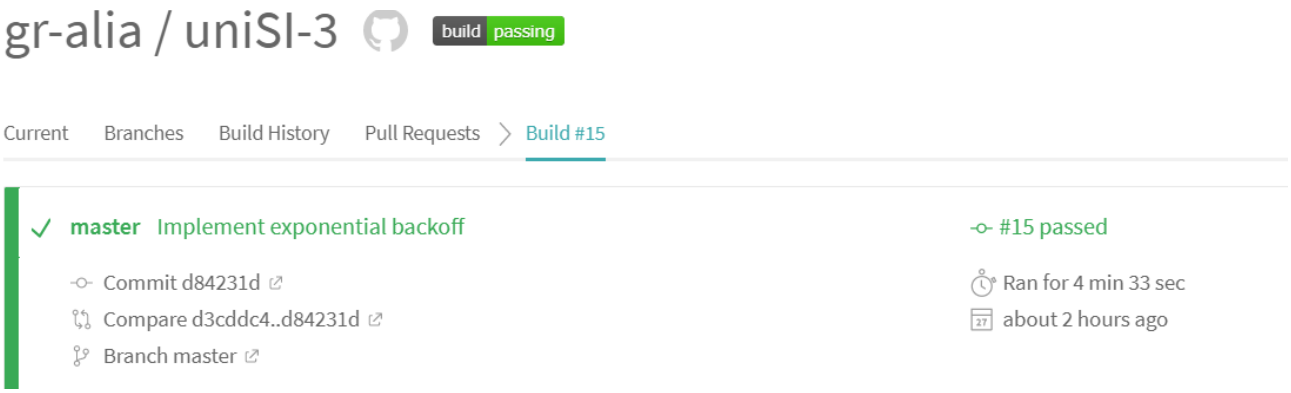

Рис.3 Безперервна інтеграція на Travis CI

4. Експоненціальна витримка

Для вирішення проблем з'єднання з сервером був використаний метод експоненціальної витримки.

За повторну спробу підключення до серверу відповідає метод бібліотеки RxJava – retryWhen(). У випадку виникнення помилки retryWhen() повторно підписується на Observable, сподіваючись, що він буде завершений без помилок.

Для формули витримки було взято мінімальний час 4 секунди у степені кількості спроб. Графік, який ілюструє вибрані інтервали для повтору спроб при експоненціальній витримці зображено на рис.4.

Для запобігання пікових станів навантаження на сервер отримане значення витримки було розподілене в межах плюс-мінус половина інтервалу витримки.

```
double delay = Math.pow(4, attempt);
double min = delay - delay/2;
double max = delay + delay/2;
delay = (min + Math.random() * ((max -min)+1));
Log.i( tag: "Exponential Backoff",  msg: "Attempt to reconnect number: " + attempt
return   Observable.timer((long) delay, TimeUnit.SECONDS);
```

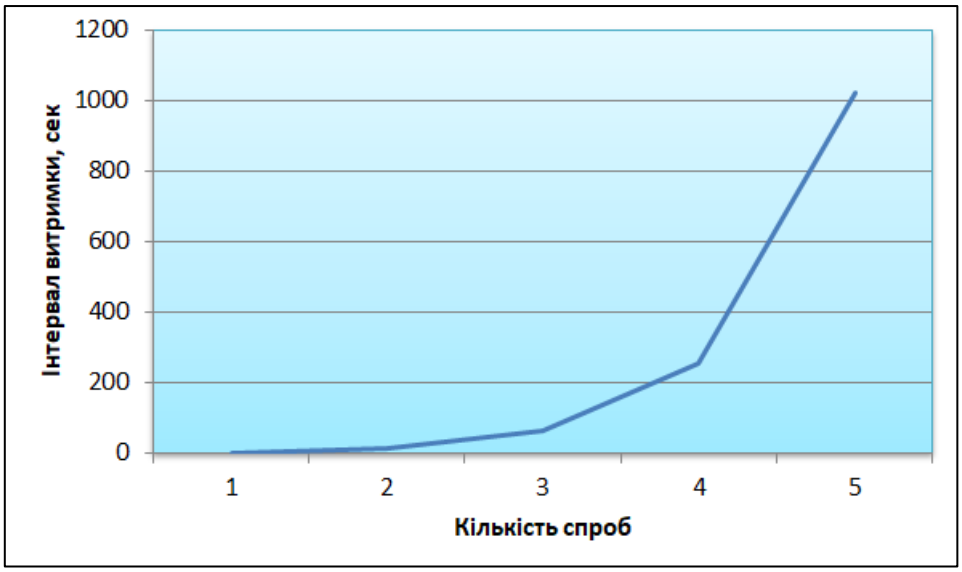

Рис.4. Графік експоненціальної витримки