



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Розрахунково-графічна робота

з дисципліни «Інтеграційні програмні системи»

Виконали:

Студенти IV курсу ФІОТ
Групи ІО-44
Андрусів А. С.
Козловський І. О.
Шевчук Б. М.
Зальотнов О.

Перевірів:

Асистент
Мазур Р. Ф.

1. Опис проекту

У проекті реалізовано базовий десктопний додаток для роботи з Твіттером. Проект написаний на Java з використанням JavaFx та бібліотеки для зв'язку з Twitter API – twitter4j (<http://twitter4j.org/en/>)
Проект збирається за допомогою Maven та проводяться тести за допомогою JUnit.

(GIT: <https://github.com/UnderDodge/TwitterProject>)

2. Принцип роботи

Для роботи з програмою необхідно мати профіль з правами розробника.
(детальніше: <https://apps.twitter.com/>)

Для тесту програми наша команда створила окремий профіль

При запуску програми перше що баче користувач це форма логіну саме в цей «профіль розробника».

Логін здійснюється за допомогою ключів які надаються після отримання прав розробника по ссилці вище.

(Ми надали ключі до нашого тестового профілю у git-репозиторії, дивитись у README.md)

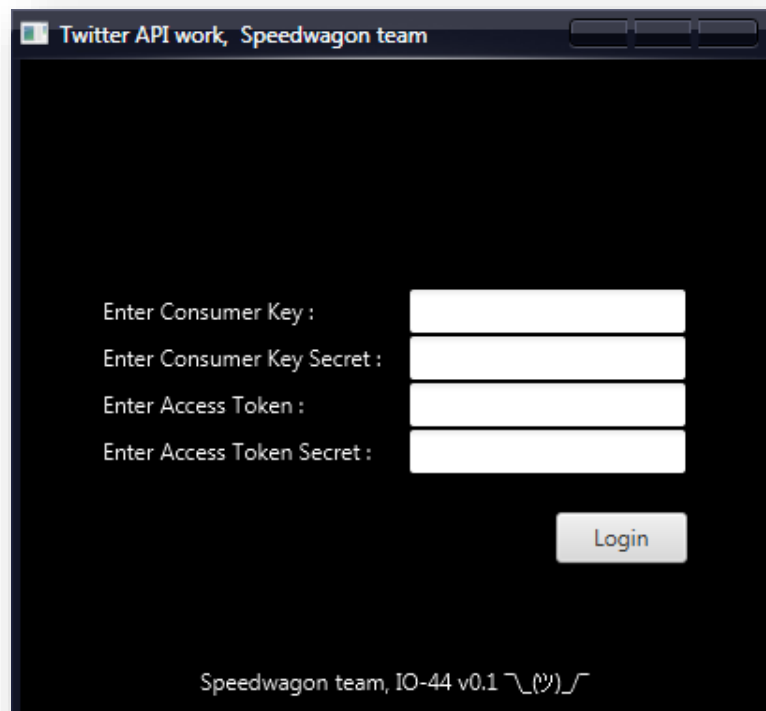


Рис №1: Логін

Після успішного логіну користувач переходить до «основного вікна» програми.

Програма відображає іконку, назву, та кількість послідовників профілю. Користувач має можливість зробити новий твіт, продивитись таймлайн, та зробити пошук твітів за тегом.
(була реалізована тільки базова функціональність необхідна для виконання лабораторних робіт)

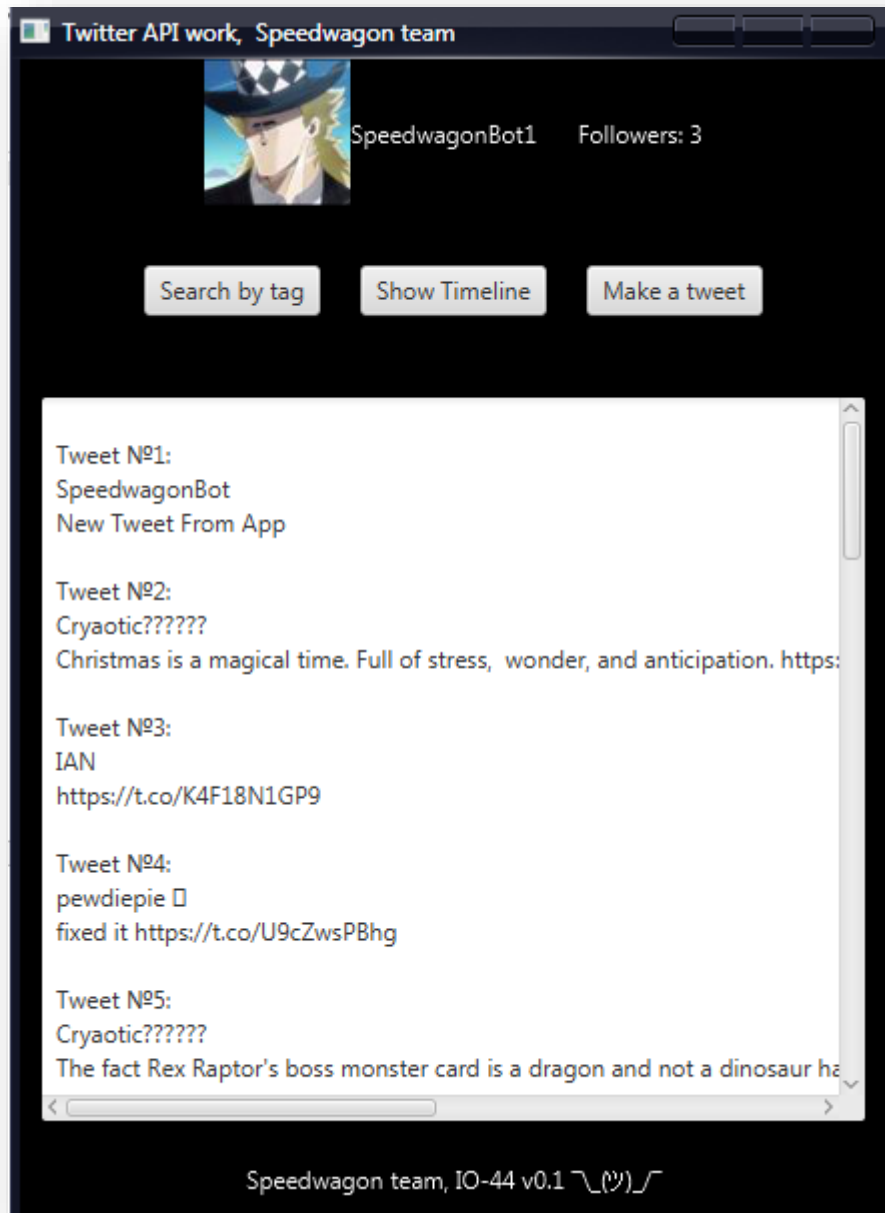


Рис №2: Основне вікно проекту

3. Система автоматичної збірки

У проєкті використовується система збірки Maven. Maven - це інструмент для збірки Java проєкту: компіляції, створення jar, створення дистрибутива програми, генерації документації. Прості проєкти можна зібрати в командному рядку. Якщо збирати великі проєкти з командного рядка, то команда для збірки буде дуже довгою, тому її іноді записують в bat / sh скрипт. Але такі скрипти залежать від платформи. Для того щоб позбутися від цієї залежності і спростити написання скрипта використовують інструменти для збірки проєкту.

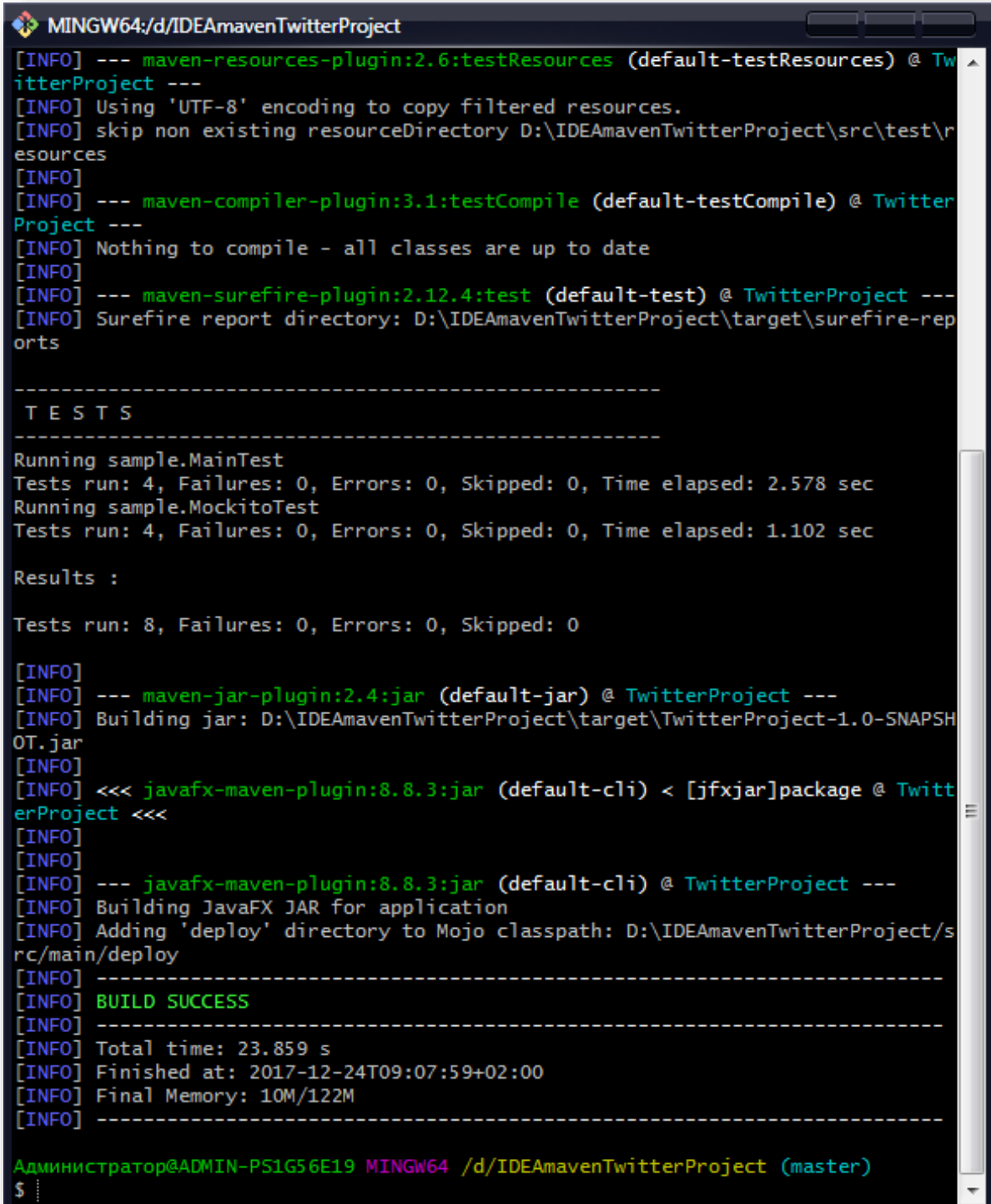
Основні переваги використання Maven:

- **Незалежність від OS.** Збірка проєкту відбувається в будь-якій операційній системі. Файл проєкту один і той же.
- **Управління залежностями.** Рідко які проєкти пишуться без використання сторонніх бібліотек (залежностей). Ці сторонні бібліотеки часто теж в свою чергу використовують бібліотеки різних версій. Мавен дозволяє управляти такими складними залежностями. Що дозволяє вирішувати конфлікти версій і в разі потреби легко переходити на нові версії бібліотек.
- **Можлива збірка з командного рядка.** Таке часто необхідно для автоматичного складання проєкту на сервері (Continuous Integration).
- **Гарна інтеграція з середовищами розробки.** Основні середовища розробки на java легко відкривають проєкти які збираються з допомогою maven. При цьому найчастіше проєкт налаштовувати не потрібно - він відразу готовий до подальшої розробки.
- **Декларативний опис проєкту.**

Для опису програмного проєкту який потрібно побудувати (*build*), Maven використовує конструкцію відому як [Project Object Model](#) (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач - таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).

Двигун ядра може динамічно завантажувати [плагіни](#) з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних [Java](#)-проєктів з відкритим кодом, від Apache та інших організацій та окремих розробників. Цей репозиторій та його реорганізований наступник, - Maven 2 репозиторій, - намагається бути де-факто механізмом для дистрибуції [Java](#) програм, але прийняття його в такій якості йде повільно.

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє як первісний засіб синхронізації виходу проектів на локальній системі.



```
MINGW64:/d/IDEAmavenTwitterProject
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ TwitterProject ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\IDEAmavenTwitterProject\src\test\resources
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ TwitterProject ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ TwitterProject ---
[INFO] Surefire report directory: D:\IDEAmavenTwitterProject\target\surefire-reports

-----
T E S T S
-----
Running sample.MainTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.578 sec
Running sample.MockitoTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.102 sec

Results :

Tests run: 8, Failures: 0, Errors: 0, Skipped: 0

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ TwitterProject ---
[INFO] Building jar: D:\IDEAmavenTwitterProject\target\TwitterProject-1.0-SNAPSHOT.jar
[INFO] <<< javafx-maven-plugin:8.8.3:jar (default-cli) < [jfxjar]package @ TwitterProject <<<
[INFO] --- javafx-maven-plugin:8.8.3:jar (default-cli) @ TwitterProject ---
[INFO] Building JavaFX JAR for application
[INFO] Adding 'deploy' directory to Mojo classpath: D:\IDEAmavenTwitterProject\src/main/deploy
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 23.859 s
[INFO] Finished at: 2017-12-24T09:07:59+02:00
[INFO] Final Memory: 10M/122M
[INFO] -----

Администратор@ADMIN-PS1G56E19 MINGW64 /d/IDEAmavenTwitterProject (master)
$
```

Рис №3: Приклад команди `mvn jfx:jar`

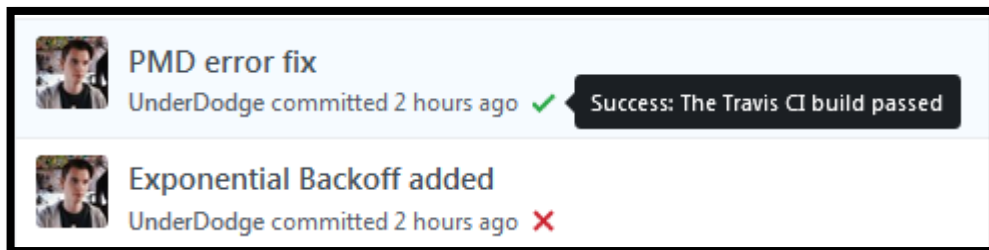
Також Maven працює з JUnit для перевірки програми тестами. Тестування це один з необхідних сходинок в циклі збірки Maven.

Особливість тестування в Maven яка найбільш подобається нашій команді це те що Maven проганяє усе що знаходиться в файлах де частиною назви файлу є «Test». Тобто для організації тестів достатньо всього лиш створити файл з тестами і частиною назви вказати «Test» і Maven побачить його і при наступній збірці пройде усі тести які знаходяться в цьому файлі на етапі тестування.

4. Задачі, що виконуються на сервері безперервної інтеграції

GitHub репозиторій проекту зв'язаний з Travis.

Який виконує необхідні задачі та дає нам знати чи була збірка успішною:



Для кожного нового коміту виконуються наступні дії:

1. Тестування за допомогою тестів, які є частиною збірки проекту за допомогою Maven.

2. Перевірка коду за допомогою PMD.

PMD - статичний аналізатор вихідних кодів Java із відкритим вихідним кодом, який повідомляє про проблему в коді програми. PMD містить вбудовані набори правил і підтримує можливість писати власні правила. PMD не повідомляє помилки компіляції, оскільки він може обробляти добре сформовані вихідні файли. Питання, про які повідомляється PMD, є досить неефективним кодом або поганими звичками програмування, які можуть знизити продуктивність і зручність роботи програми, якщо вони накопичуються.

3. Збірка Jar файлу за допомогою Maven.

4. Побудова проекту в docker

5. Експоненційна витримка

Twitter API має свою особисту «витримку» а точніше ліміт, він обмежує використання get запитів до 15 за 15 хвилин. Що дуже мало для роботи... Але не дивлячись на цей ліміт, в ньому мало сенсу бо перенагрузка сервісу все одно можлива, бо всі ці 15 запитів можуть буду здійснені одночасно...

Як би там не було, в проекті в основному є 2 причини які можуть визвати Twitter exeption:

- 1) Було досягнуто ліміту get запитів
- 2) Пропав зв'язок з Twitter API

Тут якраз і можна використати експоненційну витримку.

При отриманні twitter exeption запит (будь то get чи set) буде виконаний знову з затримкою у 1 секунду. Якщо exeption не пропав то запит буде виконаний знову але затримка до нього буде збільшена до 4 секунд.

Таким чином запит буде з затримкою 1, 4, 9, 16, 25 секунд і якщо він не виконається за цей період то буде скинутий.

