

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»

КАФЕДРА АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА
УПРАВЛІННЯ

КУРСОВА РОБОТА

з дисципліни «Технології паралельних та розподілених обчислень - 1»
на тему: «Github-Livetracker. Розробка системи для відстеження в Github
діяльності розробників з наборами ключових слів»

Студентки 4 курсу групи ІП-71

спеціальності 121 "Інженерія

програмного забезпечення"

Каспрук А.А.

Керівник _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2020 рік

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизованих систем обробки інформації і управління
Дисципліна «Технології паралельних та розподілених обчислень - 1»
Спеціальність 121 "Інженерія програмного забезпечення"

Курс 4

Група ІІІ-71

Семестр 7

ЗАВДАННЯ
на курсову роботу студентки
Каспрук Анастасії Андріївни

1. Тема роботи Github-Livetracker. Розробка системи для відстеження в Github
діяльності розробників з наборами ключових слів.

2. Строк здачі студентом закінченої роботи 30.12.2020

3. Вихідні дані до роботи діаграма архітектури системи, сервіс реєстрації користувача,
сервіс обробки ключових слів, база даних для Github-Livetracker.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
Постановка задачі, опис архітектури програмної системи, опис мікросервісу реєстрації
“Auth Service”, опис мікросервісу обробки ключових слів “Processing Keywords Service”,
тестування програмного забезпечення.

5. Дата видачі завдання 01.12.2020

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	01.12	
2.	Підготовка ТЗ	06.12	
3.	Аналіз предметної області	08.12	
4.	Проектування архітектури програмної системи	10.12	
5.	Розробка сценарію роботи програми	11.12	
6.	Розробка програмного забезпечення	23.12	
7.	Тестування програми	25.12	
8.	Підготовка пояснювальної записки	28.12	
9.	Здача курсової роботи на перевірку	30.12	
10.	Захист курсової роботи	30.12	

Студент _____
(підпис)

Керівник _____
(підпис)

(прізвище, ім'я, по батькові)

" 30 " грудня 2020 р

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 20 сторінок, 9 рисунків, 7 посилань.

Об'єкт дослідження: діяльність розробників з наборами ключових слів у Github.

Мета роботи: розробити систему для відстеження в Github діяльності розробників з наборами ключових слів.

У даній курсовій роботі було спроектовано та реалізовано два мікросервіси та спільну для них базу даних.

Для розробки було обрано наступні технології:

- а) Серверна частина – ASP.NET Core 3.1;
- б) База даних – SQL Server.

Результатом виконаної роботи стала система, що дозволяє відслідковувати роботу інших розробників у Github з вказаними ключовими словами у режимі реального часу.

ЗМІСТ

ВСТУП	6
1. ПОСТАНОВКА ЗАДАЧІ.....	7
2. ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ	8
2.1. ОПИС МІКРОСЕРВІСУ РЕЄСТРАЦІЇ “AUTH SERVICE”	12
2.2. ОПИС МІКРОСЕРВІСУ ОБРОБКИ КЛЮЧОВИХ СЛІВ “PROCESSING KEYWORDS SERVICE”	14
3. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	17
ВИСНОВОК.....	19
ПЕРЕЛІК ПОСИЛАНЬ.....	20

ВСТУП

Ціль: розробити систему, що дозволяє відслідковувати роботу інших розробників у Github з вказаними ключовими словами у режимі реального часу.

Призначення курсової роботи: закріпити, поглибити та узагальнити базові теоретичні знання з паралельних обчислень, реактивного програмування та проектування архітектури проекту.

Загальна постановка завдання: розробити систему з можливістю реєстрації користувача, підписки та відписки на сповіщення у режимі реального часу по ключових словах у Github, отримання статистики з використання ключових слів.

Під час виконання даної курсової роботи був застосований реактивний підхід для забезпечення розсилки сповіщень у режимі реального часу.

Реактивне програмування – парадигма програмування, що побудована на потоках даних та взаємному розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Реактивне програмування — програмування з асинхронними потоками даних. Це поняття також називають подієво-орієнтованим програмуванням.

1. ПОСТАНОВКА ЗАДАЧІ

Завдання - розробити систему, що дозволяє відслідковувати роботу інших розробників у Github з вказаними ключовими словами у режимі реального часу, де ключовим словом може бути назва модулю, збірки, бібліотеки, інструменту тощо.

Для реалізації слід використати відповідний [Github API](#), який дозволяє шукати певне ключове слово в індексованому коді. Система повинна надсилати оновлення та сповіщення в режимі реального часу для конкретного ключового слова.

Користувач системи повинен мати можливість передати набір ключових слів, які служба повинна відстежувати.

Користувач системи повинен мати доступ до сповіщень у режимі реального часу для певного ключового слова чи набору ключових слів.

Користувач системи повинен мати змогу отримати інформацію про найбільш активні проекти, що використовують ключове слово.

Послуга повинна забезпечити механізми реєстрації та аутентифікації користувача.

Система у якості однієї зі своїх складових частин повинна реалізувати “workers” механізм, який буде періодично опитувати Github API та отримувати оновлення. Лише один “worker” повинен одночасно опитувати Github API для певного ключового слова, навіть якщо декілька користувачів відслідковують дане слово.

Сервіс повинен мати можливість запускати “worker”-и на одній машині або на декількох та збирати інформацію.

2. ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

Система реалізовує мікросервісну архітектуру.

Мікросервіси — архітектурний стиль, за яким єдиний застосунок будується як сукупність невеликих сервісів, кожен з яких працює у своєму власному процесі та спілкується з рештою, використовуючи прості та швидкі протоколи передачі даних, зазвичай HTTP. Ці сервіси будуються навколо бізнес-потреб і розгортаються незалежно один від одного зазвичай з використанням автоматизованого середовища. Централізоване керування цими сервісами досягає мінімуму. Самі по собі вони можуть бути написані з використанням різних мов програмування і технологій зберігання даних.

Мікросервіси допомагають розробникам доставляти зміни швидше, безпечніше і з більш високою якістю, тобто зберігати швидкість розвитку продукту. Адже не тісно зв'язані сервіси дають можливість проводити зміни з більшою частотою ітерацій мінімізуючи вплив змін на решту частин системи.

Основні властивості:

- високий рівень незалежності: незалежна розробка, незалежне розгортання;
- незалежне масштабування;
- невелика кодова база зменшує кількість конфліктів та дозволяє швидко залучати нових розробників;
- простота заміни однієї реалізації сервісу іншою;
- простота додавання нового функціоналу в систему;
- ефективне використання ресурсів
- еластичність: вихід з ладу одного сервісу зазвичай не призводить до виходу з ладу всієї системи;

- сервіси організовані відносно бізнес логіки яку вони виконують;
- кожен сервіс незалежно від інших може бути реалізований за допомогою будь-якої мови програмування, СУБД, та ін.

Архітектуру розробленої системи можна зобразити за допомогою наступної діаграми.

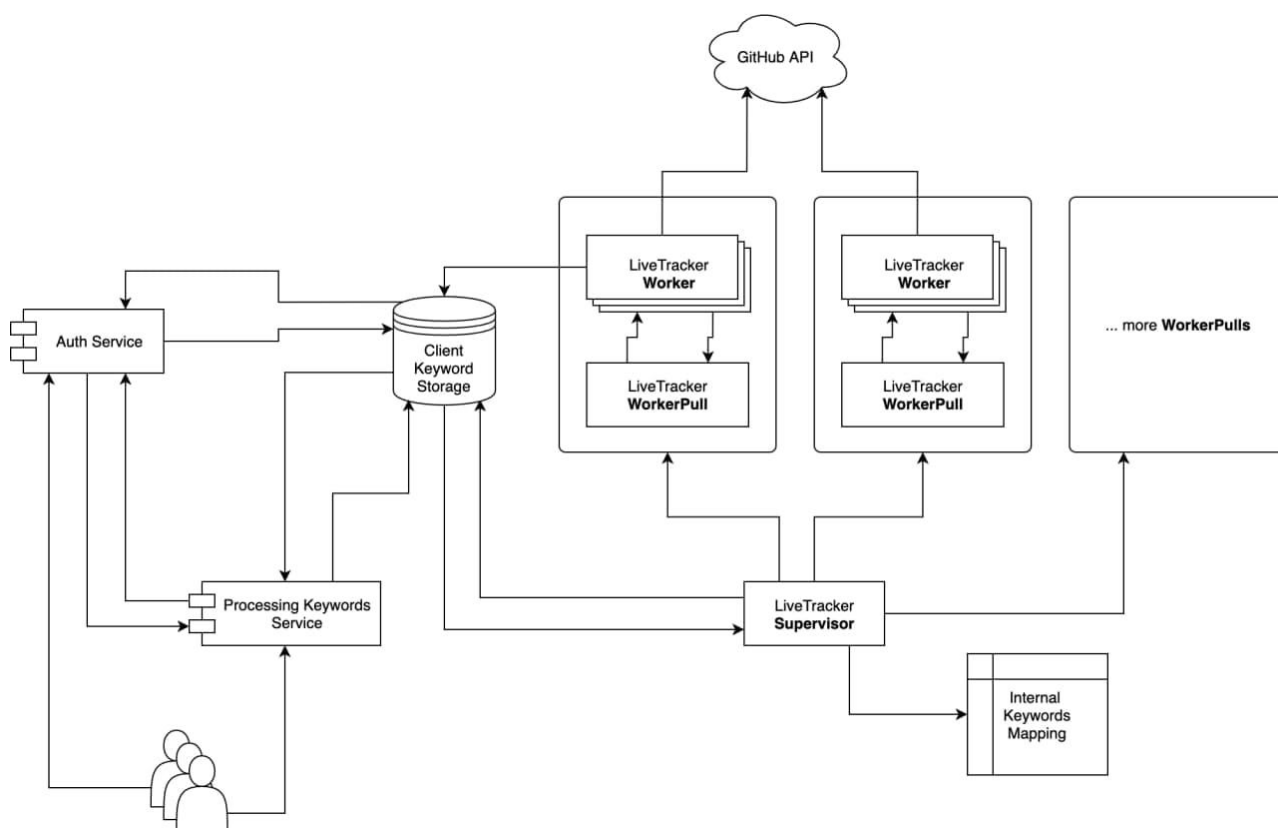


Рисунок 2.1 – Архітектура розробленої системи.

У даній курсовій роботі описана реалізація двох мікросервісів, з якими взаємодіє користувач, та бази даних, що їх об'єднує, а саме “Auth Service”, “Processing Keywords Service” та “Client Keyword Storage”.

Для того, щоб користуватися системою, користувач повинен зареєструватися. Реєстрація користувача відбувається через “Auth Service”.

Після реєстрації користувач отримує токен доступу, який він повинен використовувати для підключення до “Processing Keywords Service”.

“Processing Keywords Service” дозволяє отримувати сповіщення у режимі реального часу по ключовим словам, а також інформацію про найбільш активні проекти, що їх використовують.

Як видно з рисунку 2.1, база даних “Client Keyword Storage” є посередником між тією частиною системи, з якою безпосередньо взаємодіє користувач, та частиною системи, яка отримує інформацію по ключовим словам з Github API.

База даних “Client Keyword Storage” має наступну структуру.

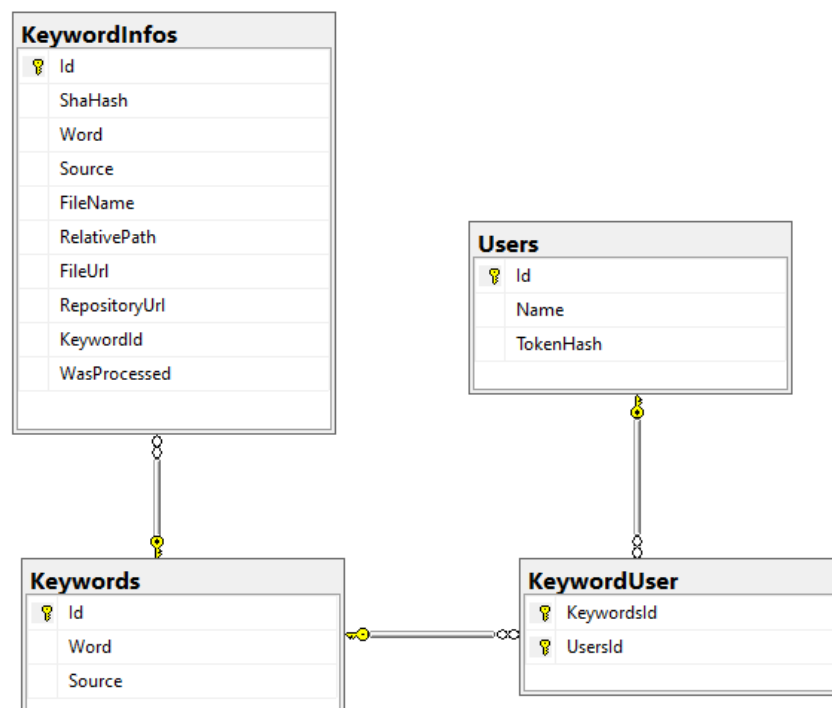


Рисунок 2.2 – ER діаграма бази даних “Client Keyword Storage”.

Коли користувач підписується на ключове слово, воно потрапляє у таблицю “Keywords”. Частина системи, яка взаємодіє з Github API, відслідковує зміни у цій таблиці і при появі там нового ключового слова

починає відстежувати зміни по ньому в Github. Отримані з Github API дані записуються у таблицю “KeywordInfos”.

“Processing Keywords Service” у свою чергу відслідковує додавання нових даних у таблицю “KeywordInfos”. Усі нові записи там зчитуються і розсилаються підписаним на відповідні ключові слова користувачам.

Отримання змін з таблиць бази даних у режимі реально часу відбувається за допомогою інструменту SqlTableDependency.

SqlTableDependency - це високорівневий компонент C#, який використовується для аудиту, моніторингу та отримання повідомлень про будь-які зміни в таблиці бази даних SQL Server, як от операції вставки, оновлення чи видалення. Повідомлення, що містить значення зміненого запису, доставляється в SqlTableDependency.

Цей інструмент для відстеження змін дозволяє уникнути виконання зайвих запитів для отримання оновлень з таблиці, оскільки всі зміни доставляються з повідомленнями через SqlTableDependency.

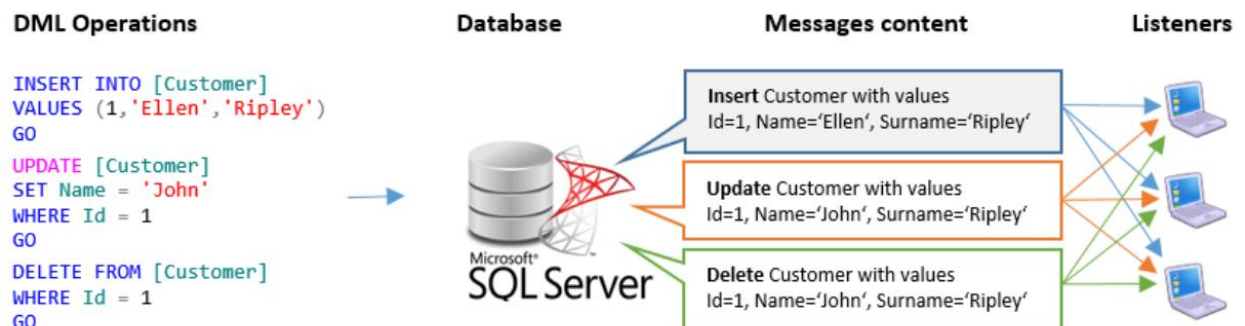


Рисунок 2.3 – Принцип роботи інструменту SqlTableDependency.

SqlTableDependency надає низькорівневу реалізацію для отримання сповіщень про зміни таблиці бази даних. Інструмент створює тригери, черги та службовий посередник SQL Server, який негайно повідомляє програму, коли відбувається зміна у записах.

2.1. ОПИС МІКРОСЕРВІСУ РЕЄСТРАЦІЇ “AUTH SERVICE”

“Auth Service” – сервіс реєстрації користувача. Реалізований як REST API.

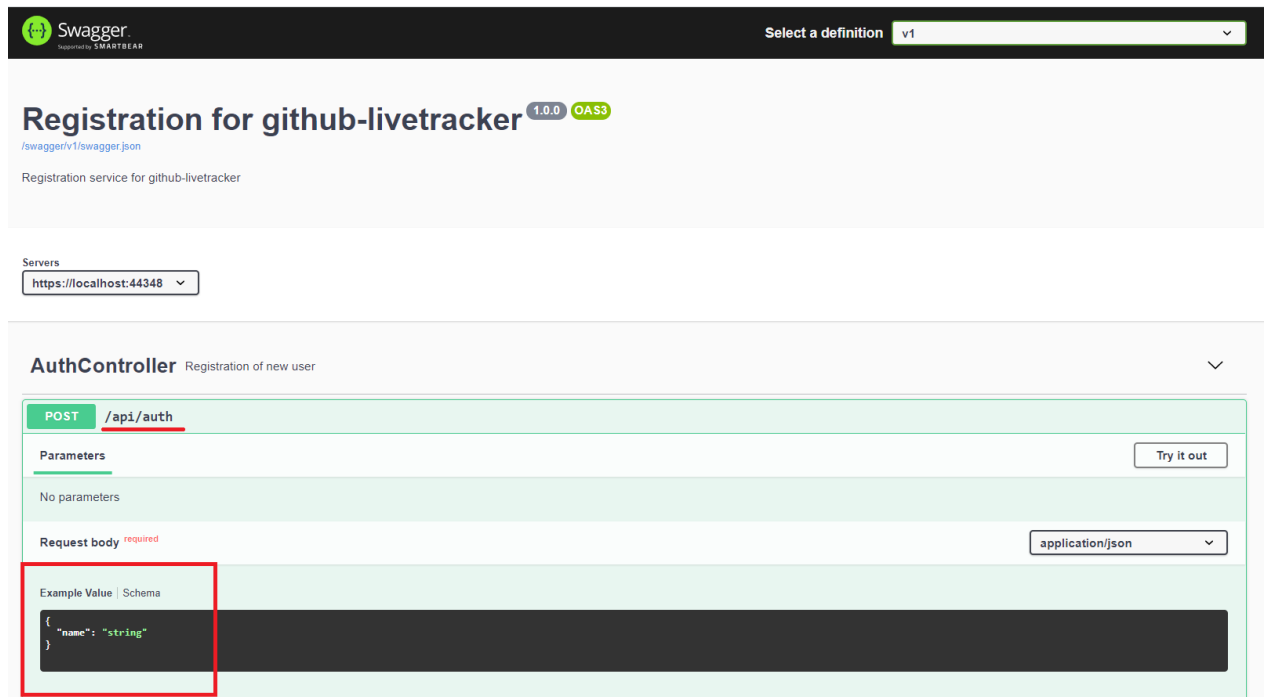


Рисунок 2.1.1 – API “Auth Service”.

Реєстрація відбувається через POST метод. Для реєстрації користувач у тілі запиту повинен передати своє ім'я за адресою `http://[сервер]:[порт]/api/auth`.

Під час реєстрації сервіс генерує унікальний токен. Інформація про користувача, а саме його ім'я та токен у хешованому вигляді, зберігаються у таблицю “Users” бази даних “Client Keyword Storage”. Токен хешується за допомогою алгоритму SHA 256.

Користувачу у відповідь повертається його токен у відкритому вигляді. Надалі цей токен використовується для підключення до “Processing Keywords Service”.



Рисунок 2.1.2 – Результат реєстрації.

2.2. ОПИС МІКРОСЕРВІСУ ОБРОБКИ КЛЮЧОВИХ СЛІВ “PROCESSING KEYWORDS SERVICE”

Сповіднення у режимі реального часу користувач отримує завдяки підключенню по WebSocket.

WebSocket — протокол, що призначений для обміну інформацією між клієнтом та сервером в режимі реального часу. Він забезпечує двонаправлений повнодуплексний канал зв'язку через один TCP-сокет.

При підключенні по WebSocket токен доступу передається як частина рядка запиту (query string).

Користувач може під'єднатися за адресою `ws://[сервер]:[порт]/keywords?access_token=[токен доступу]`.

Якщо у користувача вже є зареєстровані для прослуховування ключові слова, то з моменту підключення він зможе отримувати сповіщення по них. Також користувач може підписатися на нові або відписатися від вже відслідковуваних ключових слів.

Сервіс надає користувачу дані в режимі реального часу, що потрапляють до таблиці “KeywordInfos”, а також дані, що ще не були оброблені сервісом “Processing Keywords Service” по причині його можливої тимчасової непрацездатності. Дані зливаються у єдиний потік. Таким чином, система забезпечує, що всі записи у “KeywordInfos” будуть оброблені сервісом “Processing Keywords Service” та всі підключені користувачі отримають дані у повному обсязі.

Потік даних можна зобразити за допомогою наступної діаграми.

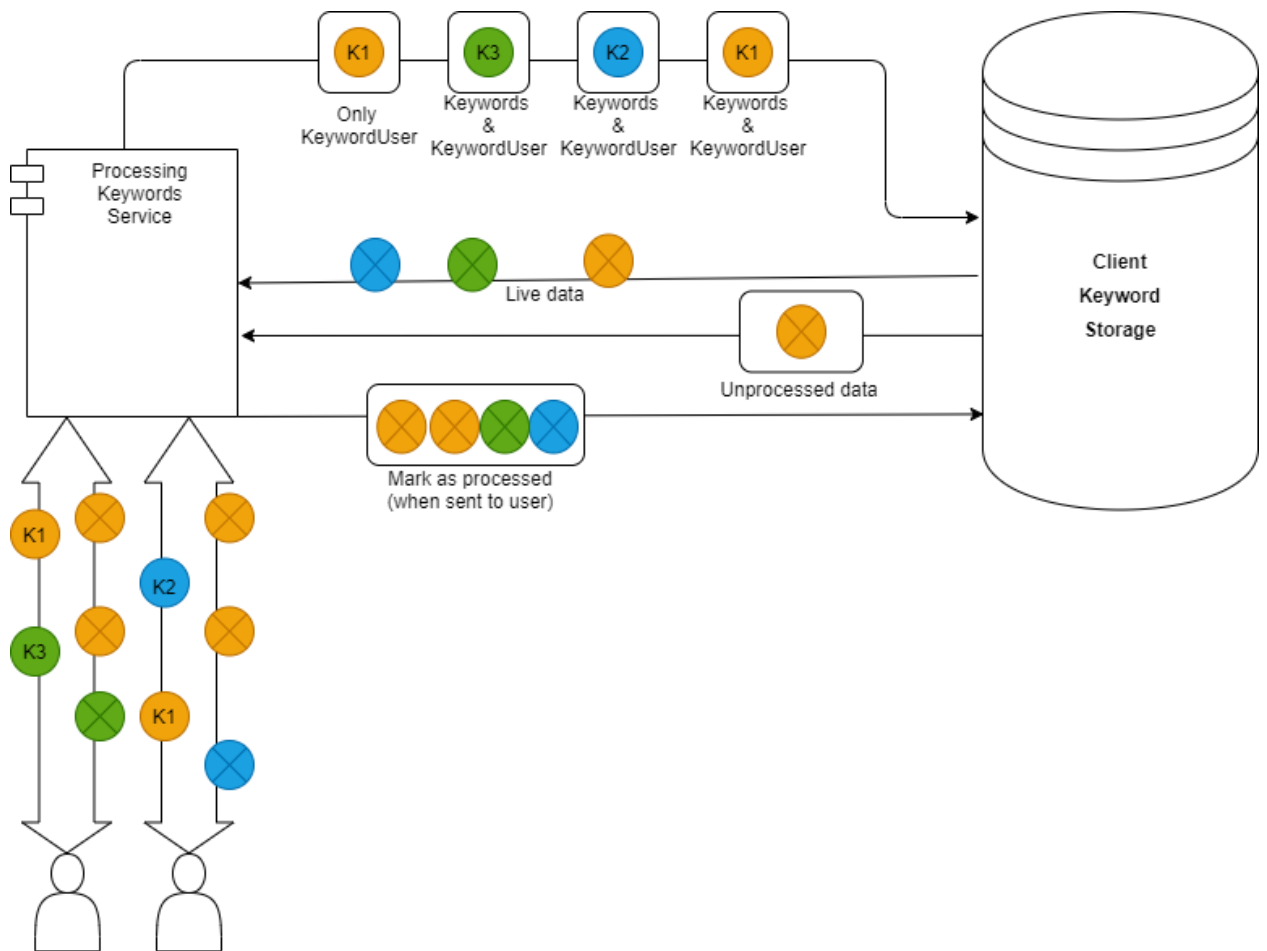


Рисунок 2.2.1 – Діаграма потоку даних.

На діаграмі зображений приклад підписки двох користувачів на ключові слова, які умовно позначені як K1, K2 і K3. Сервіс “Processing Keywords Service” обробляє заявки на підписки. Ключові слова поміщаються у таблицю “Keywords”, якщо їх ще там нема. Між користувачами та ключовими словами існує зв’язок багато до багатьох, що реалізовується через таблицю “KeywordUser”. При оформленні підписки в “KeywordUser” додається відповідний запис.

Після того, як ключове слово потрапило у таблицю “Keywords”, воно підхоплюється частиною системи, яка взаємодіє з Github API. Оновлення з Github API записуються у таблицю “KeywordInfos” та автоматично підхоплюються сервісом “Processing Keywords Service” за допомогою

інструменту SqlTableDependency. Надалі ці оновлення пересилаються користувачам у режим реального часу.

“Processing Keywords Service” також надає можливість отримати інформацію про найбільш активні проекти, що використовують ключове слово за допомогою REST GET методу за запитом [http://\[сервер\]:\[порт\]/api/keyword-statistics/for-keyword?keyword=\[ключове слово\]](http://[сервер]:[порт]/api/keyword-statistics/for-keyword?keyword=[ключове слово]).

При підключенні по HTTP токен доступу передається у хедері “Authentication”.



Рисунок 2.2.2 – Приклад запиту на отримання статистики по ключовому слову.

3. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для класів “Processing Keywords Service”, що для реалізації бізнес-логіки використовують реактивний підхід, були написані модульні (unit) тести за допомогою середовища NUnit.

NUnit - відкрите середовище юніт-тестування додатків для .NET. Воно було перенесено з мови Java (бібліотека JUnit). Перші версії NUnit були написані на J#, але потім весь код був переписаний на C# з використанням атрибутів .NET.

Усі автотести винесені в окремий проєкт KeywordAPI.Test.

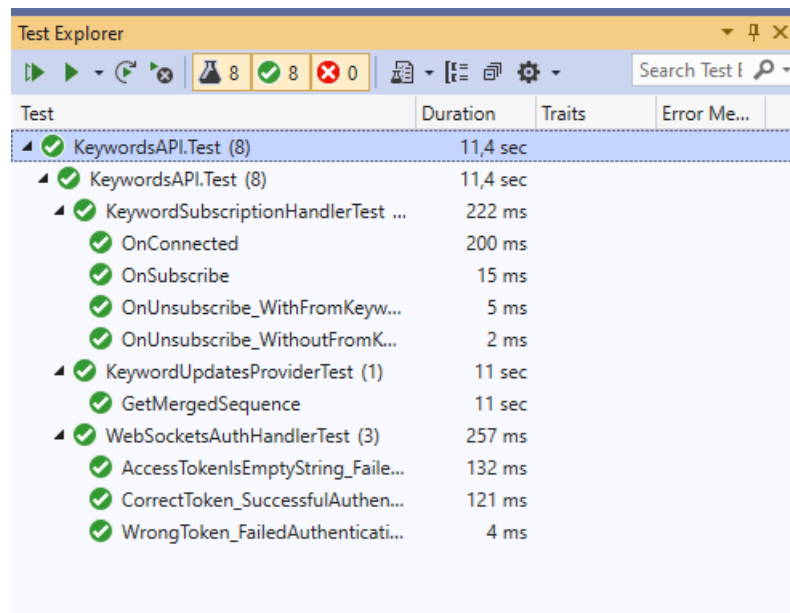


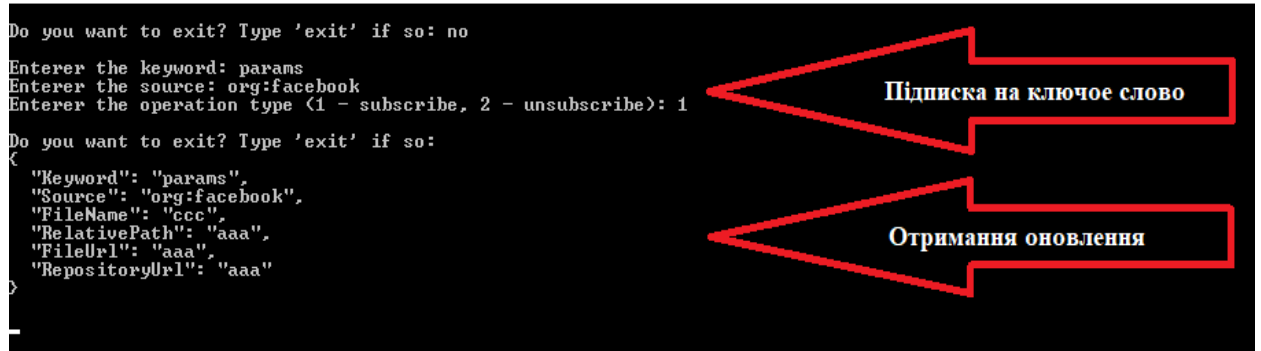
Рисунок 3.1 – Тестування у середовищі Visual Studio.

Значна увага була приділена мануальному тестуванню. Сервіси, що реалізовували методи REST API, такі як “Auth Service” (реєстрація користувача) та “Processing Keywords Service” (отримання статистики по ключовому слову) були протестовані за допомогою Swagger.

Swagger – технологія, яка дозволяє документувати REST-сервіси.

Для тестування підключення по WebSocket був написаний окремий консольний клієнт. За допомогою нього можна підписуватися на ключові

слова та відписуватися від них, а також отримувати оновлення у режимі реального часу.



```
Do you want to exit? Type 'exit' if so: no
Enterer the keyword: params
Enterer the source: org:facebook
Enterer the operation type (1 - subscribe, 2 - unsubscribe): 1
Do you want to exit? Type 'exit' if so:
{
  "Keyword": "params",
  "Source": "org:facebook",
  "FileName": "ccc",
  "RelativePath": "aaa",
  "FileUrl": "aaa",
  "RepositoryUrl": "aaa"
}
```

Підписка на ключове слово

Отримання оновлення

Рисунок 3.2 – Робота з консольним клієтом.

ВИСНОВОК

Першим кроком при розробці системи Github-Livetracker було проектування архітектури. Це дозволило виділити основні сутності, сервіси та визначитися зі стеком технологій.

Наступним кроком була розробка сервісу реєстрації “Auth Service”, що забезпечує успішну реєстрацію у системі та отримання токена доступу.

Подальшим етапом стала реалізація сервісу “Processing Keywords Service”. Він дозволяє користувачу отримувати дані у режимі реального часу для певного ключового слова чи набору ключових слів та інформацію про найбільш активні проекти, що використовують ключове слово.

У рамках даної курсової роботи вдалося доречно використати реактивну парадигму програмування для досягнення поставлених цілей, а також поглиблення навичок і знань у цій області.

В результаті виконання курсової роботи ми отримали програмний продукт, який повністю відповідає заданим вимогам та очікуваним результатам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Tamir Dresher Rx.NET in Action : посібник. Shelter Island, NY : Manning, 2017.
2. Введение в ASP.NET Core [Електронний ресурс] – Режим доступу <https://metanit.com/sharp/aspnet5/1.1.php>
3. Керівництво ASP.NET Core 3 [Електронний ресурс] – Режим доступу <https://metanit.com/sharp/aspnet5/>
4. Мова програмування C# та платформи .Net та .NET Core [Електронний ресурс] – Режим доступу https://codernet.ru/books/c_sharp
5. Енциклопедія C# та платформи .Net [Електронний ресурс] – Режим доступу <https://metanit.com/sharp/tutorial/>
6. MS SQL Server 2017 [Електронний ресурс] – Режим доступу <https://metanit.com/sql/sqlserver/>
7. Руководство по Entity Framework Core [Електронний ресурс] – Режим доступу <https://metanit.com/sharp/entityframeworkcore/>