МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут електронних та інформаційних технологій

Факультет електронних та інформаційних технологій

Кафедра інформаційних та комп'ютерних систем

Допущено до захисту

Завідувач кафедри

Базилевич Володимир Маркович

"____" _____ 2020 р.

ВИПУСКНА КВАЛІФІКЦІЙНА РОБОТА (ПРОЕКТ)

Telegram бот для пошуку дешевих авіаквитків

Спеціальність: 123 – Комп'ютерна інженерія

Виконавець:
студент гр. МКІп-191
Погуляйло М.С. _____
(підпис)

Керівник:
Зав.кафедри ІКС, к.е.н., доцент
Базилевич В.М. _____
(підпис)

Чернігів 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут електронних та інформаційних технологій

Факультет електронних та інформаційних технологій

Кафедра інформаційних та комп'ютерних систем

ЗАТВЕРДЖУЮ:

Завідувач кафедри

Базилевич Володимир Маркович

"____" _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ) ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Погуляйло Максим Сергійович

Тема роботи: Telegram бот для пошуку дешевих авіаквитків

Тему затверджено наказом ректора від "__"_____ 2020р. №_____

1. *Вхідні дані до роботи: матеріали по використовуваних технологіях, рекомендації керівника*
2. *Зміст розрахунково-пояснювальної записки: відповідно до Методичних рекомендацій.*
3. *Перелік графічного матеріалу (у разі необхідності)*_____

4. Календарний план

№	*Назва етапів роботи*	*Термін виконання*	*Примітки*
1	Вивчення та аналіз існуючих систем, технологій та інструментів	Вересень 2020	
2	Вибір технічних засобів та підходящої архітектури	Жовтень 2020	
3	Реалізація інформаційної системи	Листопад 2020	

Завдання підготував:

керівник_____ Базилевич В.М.

(підпис)

«___» _____2020__ р.

Завдання одержав:

студент_____ Погуляйло М.С.

(підпис)

«___» _____2020__ р.

АНОТАЦІЯ

Кваліфікаційна робота магістра Погуляйло Максима Сергійовича на тему: «Telegram бот для пошуку дешевих авіаквитків» складається з 60 сторінок , 38 рисунків, 6 таблиць та 16 джерел.

Основна ціль проекту - реалізація чат-боту для пошуку дешевих авіаквитків з використанням стороннього API, в якості джерела даних. Результатом розробки системи буде чат-бот, доступний всім користувачам додатку “Telegram”, та який буде надавати настпний функціонал:

- Пошук авіаквитків за вибраними напрямами
- Фільтрація квитків за датою вильоту
- Можливість підписатися на вибраний маршрут та слідкувати за змінами ціни

Проект розроблено з використанням мови програмування Java, фреймворку Spring та його модулів Spring Data JPA, Spring Web, Spring Boot. В якості джерела зберігання даних, використано PostgreSQL, та Hibernate в якості реалізації специфікації Spring Data JPA.

В майбутньому розширити проект можна за допомогою, наприклад використання стороннього API, для пошуку готелів в місті прильоту або популярних туристичних місць.

Ключові слова: чат-бот, Telegram, API, Java, Spring, авіаквитки.

ABSTRACT

Qualifying work of the master Pohuliailo Maksym Serhiyovych on the topic: "Telegram bot for finding cheap flights" contains 60 pages, 38 figures, 6 tables, 16 sources.

The main goal of the project is to implement a chatbot to search for cheap tickets using a third-party API as a data source. The result of the development of the system will be a chatbot, available to all users of the application "Telegram", and which will provide the following functionality:

- Search for tickets in selected destinations
- Filter tickets by departure date
- Ability to subscribe to the selected route and follow price changes

The project was developed using the Java programming language, the Spring framework and its modules Spring Data JPA, Spring Web, Spring Boot. As a storage source, used PostgreSQL, and Hibernate as an implementation of the Spring Data JPA specification. In the future, you can expand the project with, for example, the use of a third-party API, to search for hotels in the city of arrival or popular tourist destinations.
Keywords: chatbot, Telegram, API, Java, Spring, air tickets.

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс.

SQL (Structured Query Language) - мова структурованих запросів.

JSON (JavaScript Object Notation) – текстовий формат обміну даними між комп'ютерами.

ОС – операційна система.

ЗМІСТ

ЗМІСТ 7
ВСТУП 9
1. АНАЛІЗ ЗАДАЧІ РОЗРОБКИ ПРОГРАМНОГО СЕРВІСУ 10
1.1. Що таке Telegram і чим він відрізняється від інших програм обміну повідомленнями? 10
1.2. Що таке чат-бот 12
1.3. Які існують типи чат-ботів 12
Висновки до першого розділу 13
2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ 14
2.1. Проектування архітектури 14
2.1.1 Загальні вимоги до системи 15
2.1.2. Загальні вимоги до клієнта 15
2.2 Вибір мови та середовища програмування 16
2.2.1 Мова програмування Java 16
2.2.1.2 Особливості Java 16
2.2.1.3 Особливості Java 8 17
2.2.2 Spring Framework 19
2.2.2.1 Архітектура Spring Framework 20
2.2.2.2 Особливості Spring Framework 21
2.2.2.3 Життєвий цикл бінів в Spring Framework 22
2.2.2.4 Spring Boot 24
2.2.2 Hibernate 25
2.2.2.4 Особливості Hibernate 27
2.2.2.5 Бази данних які Hibernate підтримує 28
2.2.2.6 Spring Data JPA та Hibernate 28
2.2.3 Liquibase 29
2.2.4 SQL 31
..... 32

2.2.4.1 PostgreSQL 34
2.2.5 Apache Maven 36
2.2.5.1 Maven репозиторії 37
2.2.6 Intellij IDEA 38
2.2.6.1 Порівняння Ultimate та Community версій 41
Висновки до другого розділу 43
3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 44
3.1 Скачати Telegram 44
3.2 Реєстрація чат-боту 45
3.3 Реалізація чат-боту 49
3.3.1 Класс BotConfig 50
3.3.2 Класс UserMessageHandler 51
3.3.3 Пакет Model 53
3.3.4 Результат роботи програми 56
ВИСНОВКИ 58
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 59

ВСТУП

В наш час месенджери є, мабуть найпопулярнішими додатками на мобільні телефони. Це вже давно не просто програми за допомогою яких можна лише відправляти текстові повідомлення. Сьогодні їх можна використовувати для пересилання різних за типом, та розміром зображень, відео, файлів. Слідкувати за останніми новинами в світі, курсом валют, погодою, новинками кіно чи музики, можна навіть розпочати свою справу використовуючи один з популярних месенджерів.

Тему своєї кваліфікаційної роботи я обрав по декількох причинах. Перша — я ніколи не створював чат-ботів, хоча сам ними часто користуюся, наприклад чат-бот Державної міграційної служби, через який можна дізнатися статус виготовлення закордонного паспорта, чи чат-бот Чернігівводоканалу, через який зручно передавати показання лічильника.

Друга — чому саме чат-бот для пошуку дешевих авіаквитків, бо вперше полетів в іншу країну, знайшовши квитки за оптимальну ціну саме за допомогою чат-боту, який на жаль більше не працює.

1. АНАЛІЗ ЗАДАЧІ РОЗРОБКИ ПРОГРАМНОГО СЕРВІСУ

1.1. Що таке Telegram і чим він відрізняється від інших програм обміну повідомленнями?

Telegram - це програма для обміну повідомленнями в Інтернеті, яка працює так само, як і популярні програми обміну повідомленнями WhatsApp та Facebook Messenger. Це означає, що ви можете використовувати його для надсилання повідомлень друзям при підключенні до Wi-Fi або мобільного інтернету.

Telegram заснований на Cloud технології і стверджує, що він надає пріоритет безпеці та швидкості, що робить його хорошою альтернативою іншим популярним програмам обміну повідомленнями. Послуга запущена в 2013 році, і з тих пір вона охопила 200 мільйонів активних користувачів щомісяця.

Як і WhatsApp, Telegram також має можливість показувати статус друга в Інтернеті та додавати та обмінюватися фотографіями, відео, місцезнаходженням, контактами та документами.

Відмінною рисою Telegram є безпека. Він стверджує, що вся його діяльність, включаючи чати, групи та медіа, якими обмінюються учасники, зашифрована. Це означає, що їх не буде видно без попередньої розшифровки.

Додаток також дозволяє встановити таймери самознищення для повідомлень та носіїв, якими ви ділитесь, які можуть бути від двох секунд до одного тижня завдяки вбудованій функції «Секретний чат». Він також пропонує наскрізне шифрування, не залишаючи слідів на серверах Telegram.

Також є можливість перевірити безпеку ваших «таємних чатів», використовуючи зображення, яке служить ключем шифрування. Порівнюючи свій ключ шифрування з ключем друга, ви можете ефективно переконатись, що ваша розмова безпечна і менш вразлива до "man-in-the-middle" атак.

За данними сайту statista.com, Telegram посідає шосте місце в світі серед месенджерів за кількістю активних користувачів.

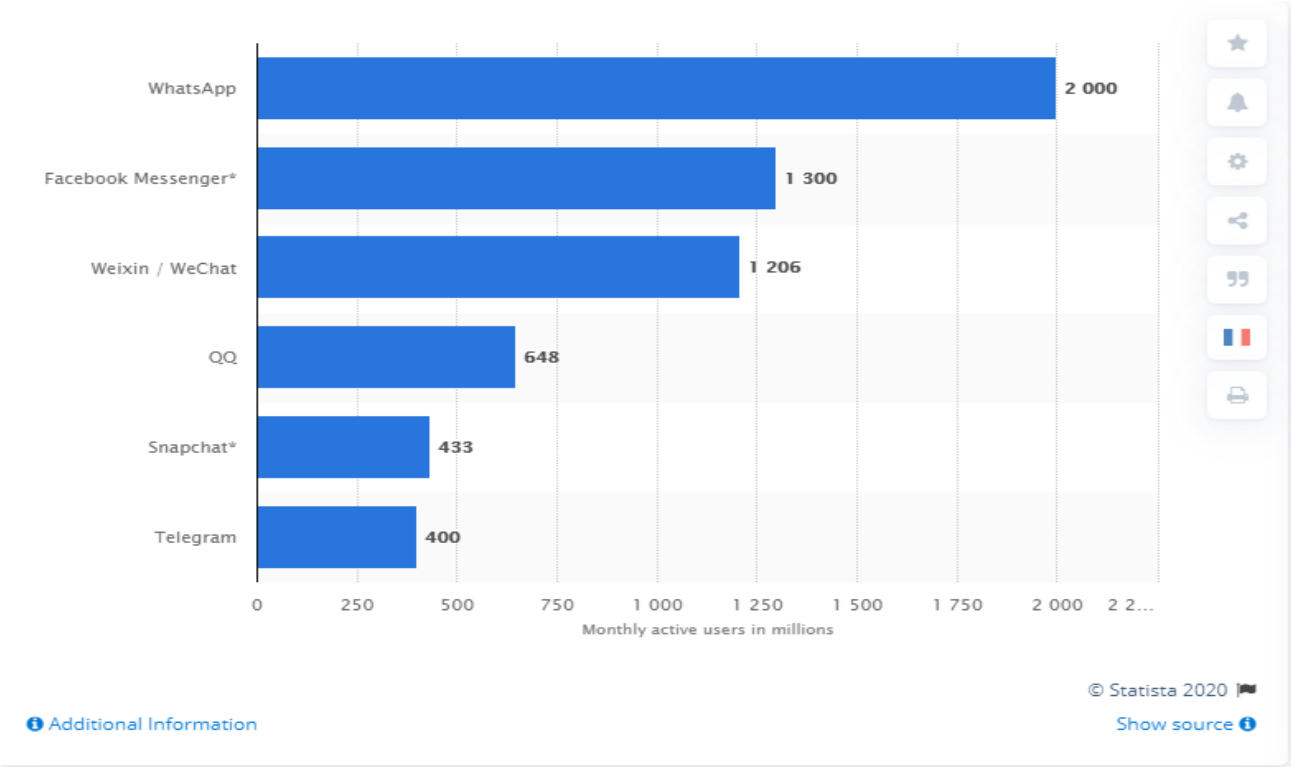


Рисунок 1.1 Світовий рейтинг популярності мессенджерів

Та третє місце в Україні за популярністю серед користувачів

Рисунок 1.2 Рейтинг популярності мессенджерів в Україні

Крім цього, Telegram надає дуже зручну API, та детальну документацію для розробки чат ботів.

1.2. Що таке чат-бот

Чат-бот - це програмне забезпечення, яке виконує автоматизоване завдання, наприклад, відповідає на поширені запитання або завершує робочий процес. Чат-боти живуть в Інтернеті на чат-платформах або в соціальних мережах і зазвичай покращуються з часом за допомогою штучного інтелекту та машинного навчання. Існує плутанина щодо того, що таке чат-бот, оскільки існують різні можливі визначення цього терміна, і існує безліч різних випадків використання ботів. Ці фактори впливають на те, як люди сприймають чат-бота. Простіше кажучи, чат-бот - це програмне забезпечення, яке може вести розмови з людьми. Наприклад, користувач може дати вказівки або задати питання боту, а натомість чат-бот може виконати певну дію або надати відповідь.

1.3. Які існують типи чат-ботів

Для того, щоб зрозуміти, що таке чат-бот, важливо знати, що існує два типи:

- На основі попередньо запрограмованого сценарію
  Розмови зі скриптовими чат-ботами йдуть лише заздалегідь визначеними шляхами. Щоб визначити наступний крок у розмові, користувач повинен вибрати між явними параметрами. Ці параметри можуть бути представлені по-різному (наприклад, текст, голос, відповідь на дотик тощо) залежно від особливостей та дизайну чат-бота.

- На основі використання штучного інтелекту

  Ці чат-боти приймають більше інформації від користувачів, оскільки вони використовують штучний інтелект (ШІ), що забезпечує більшу гнучкість. Вони здатні приймати вільну форму у вигляді голосових висловлювань або тексту. Завдяки ШІ ці чат-боти з часом можуть вдосконалюватися тим

більше, чим їх використовують (тому їх називають «розумними» чат-ботами). Однак важливо знати, що їх інтелект обмежений. Незважаючи на те, що ШІ чудово працює в обмежених доменах знань або для одноразових інструкцій, боти мають обмеження. Наприклад, цим ботам буде дуже важко зрозуміти контекст або використовувати свою «пам'ять», щоб підтримувати розмову.

Висновки до першого розділу

В цьому розділі була розглянута основна інформація про месенджери, рейтинг месенджерів в Україні та світі. Також представлена інформація про чат-ботів, їх види та області використання. На основі вище представлених досліджень, було прийнято рішення про використання додатку Telegram в якості месенджера для якого буде створюватись чат-бот.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1. Проектування архітектури

Систему створена на основі клієнт-серверної архітектури. Основні складові:

- Клієнт
- Сервер Telegram
- Сервер додатку
- База даних
- Зовнішнє API

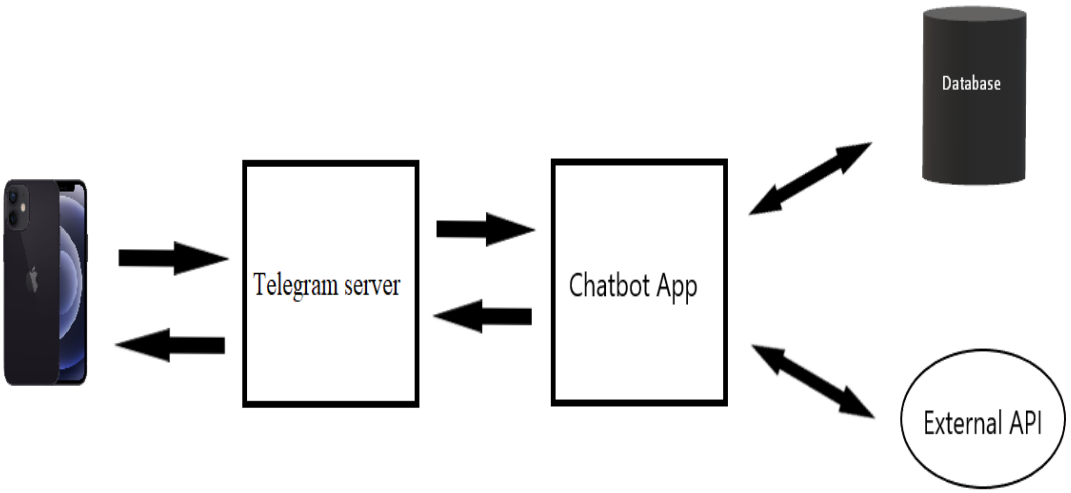


Рисунок 2.1 – Архітектура чат-боту.

Об’єкти архітектури взаємодіють за наступним сценарієм:

1) Клієнт через додаток Telegram на телефоні або комп’ютері відправляє повідомлення чат-боту.

2) Telegram реагує на повідомлення користувача, створює HTTP запит, та відправляє його на сервер додатку.
3) Сервер додатку, реагує на запит, обробляє його, якщо потрібно то посилає запит в базу даних або до зовнішнього API, формує відповідь та посилає її до серверу Telegram.
4) Telegram сервер перевіряє відповідь від серверу додатку, та відправляє її клієнту.

2.1.1 Загальні вимоги до системи

Архітектура розроблена в цій дипломній роботі повинна відповідати тій, що показана на рисунку 2.1, та дотримуватись наступних правил:

- Час відповіді від серверу до клієнта не повинен перевищувати 3с.
- Система повинна надавати інтуїтивно зрозумілий для клієнта інтерфейс.
- Система повинна бути реалізована відповідно до SOLID принципів.

2.1.2. Загальні вимоги до клієнта

Для використання чат-боту розробленного під час виконання дипломної роботи клієнт повенен:

- Бути зареєстрованим в системі Telegram.
- Мати встановлений додаток Telegram Desktop, або Telegram Mobile.
- Мати доступ до інтернету.

2.2 Вибір мови та середовища програмування

2.2.1 Мова програмування Java

Для розробки чат-боту було вирішено в якості мови програмування використовувати Java версії 1.8.

Nov 2020	Nov 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.21%	+0.17%
2	3	▲	Python	12.12%	+2.27%
3	1	▼	Java	11.68%	-4.57%
4	4		C++	7.60%	+1.99%
5	5		C#	4.67%	+0.36%
6	6		Visual Basic	4.01%	-0.22%

Рисунок 2.2 – Рейтинг мов програмування від TIOBE

Java - це мова комп’ютерного програмування загального призначення заснована на класах, об’єктно-орієнтована і спеціально розроблена, щоб мати якомога менше залежностей від реалізації. Вона призначений для того, щоб дозволити розробникам додатків " write once, run anywhere" (WORA), що означає, що скомпільований код Java може працювати на всіх платформах, що підтримують Java, без необхідності перекомпіляції.

2.2.1.2 Особливості Java

- **Об'єктно-орієнтована** - на Java все представлено як об'єкти. Об'єкт - це свого роду обгортка, яка інкапсулює дані та пов'язану з ними поведінку. Java забезпечує підтримку всіх основних об'єктно-орієнтованих принципів.

- **Незалежна від платформи** - програми, написані на Java, перетворюються спочатку в байт-код, компілятором Java. Цей байт-код можна запустити на

будь-якій машині, що має середовище виконання Java (JRE). Це робить додатки Java незалежними від платформи.
Це дуже відрізняється від програм, написаних наприклад, на C або C ++, де програми компілюються у двійкові файли, специфічні для ОС.

- **Безпечна** - програми Java працюють у середовищі виконання Java (JRE) майже без взаємодії з системною ОС. Це робить Java більш безпечним, ніж інші мови.

- **Багатопотоковість** - Java підтримує написання програм, які можуть виконувати кілька завдань в окремих потоках.
Наприклад, програма Java обслуговує форму входу користувача під час запуску фонових процесів.

- **Незалежність від архітектури ОС** - компілятор Java генерує нейтральні до архітектури ОС файли класу або байт-код.
Наприклад, у програмуванні на C тип даних int займає 2 байти пам'яті для 32-розрядної архітектури та 4 байти пам'яті для 64-розрядної архітектури. Однак він займає 4 байти пам'яті як для 32, так і для 64-розрядної архітектури в Java.

2.2.1.3 Особливості Java 8

Lambda Expression - У мові програмування Java вираз Лямбда (або функція) є просто анонімною функцією, тобто функцією без імені та без обмеження на ідентифікатор. Вони записуються саме там, де це потрібно, як правило, як параметр для якоїсь іншої функції.

```
either
(parameters) -> expression
or
(parameters) -> { statements; }
or
() -> expression
```

Рисунок 2.3 – Базовий синтаксис Labmbda Expression

Приклад використання Lamda Expression:
(x, y) -> x + y // Функція, що приймає два параметри, та повертає їхню суму.

Функціональний інтерфейс

Функціональні інтерфейси також називають інтерфейсами Single Abstract Method (інтерфейси SAM). Як випливає з назви, вони допускають рівно один абстрактний метод всередині них. Java 8 представляє анотацію, тобто @FunctionalInterface, яку можна використовувати для помилок рівня компілятора, коли інтерфейс, який ви анотували, порушує контракти функціонального інтерфейсу.

```
@FunctionalInterface
public interface MyFirstFunctionalInterface {
    public void firstWork();
}
```

Рисунок 2.4 – Приклад функціонального інтерфейсу

Stream API

Ще однією суттєвою зміною був введений Java 8 Streams API, який забезпечує механізм обробки набору даних різними способами, які можуть включати фільтрування, перетворення або будь-який інший спосіб, який може бути корисним для програми.

API потоків у Java 8 підтримує інший тип ітерацій, де ви просто визначаєте набір елементів, що підлягають обробці, операції, які потрібно виконати з кожним елементом, і де слід зберігати результати цих операцій.

Приклад API потоку. У цьому прикладі items є колекцією рядкових значень, і ви хочете видалити записи, які починаються з певного префіксу.

```
List<String> items;
String prefix;
List<String> filteredList = items.stream().filter(e -> (!e.startsWith(prefix))).collect(Collectors.toList());
```

Рисунок 2.5 – Приклад використання Stream API

2.2.2 Spring Framework

Spring - це найпопулярніший фреймворк для розробки додатків на Java. Мільйони розробників у всьому світі використовують Spring Framework для створення високопродуктивного, легко тестованого та багаторазового коду.

Spring доволі легкий, якщо говорити про розміри. Базова версія Spring framework становить близько 2 МБ.

За даними сервісу https://trends.google.com, Spring – найпопулярніший серед всіх web технологій для Java.

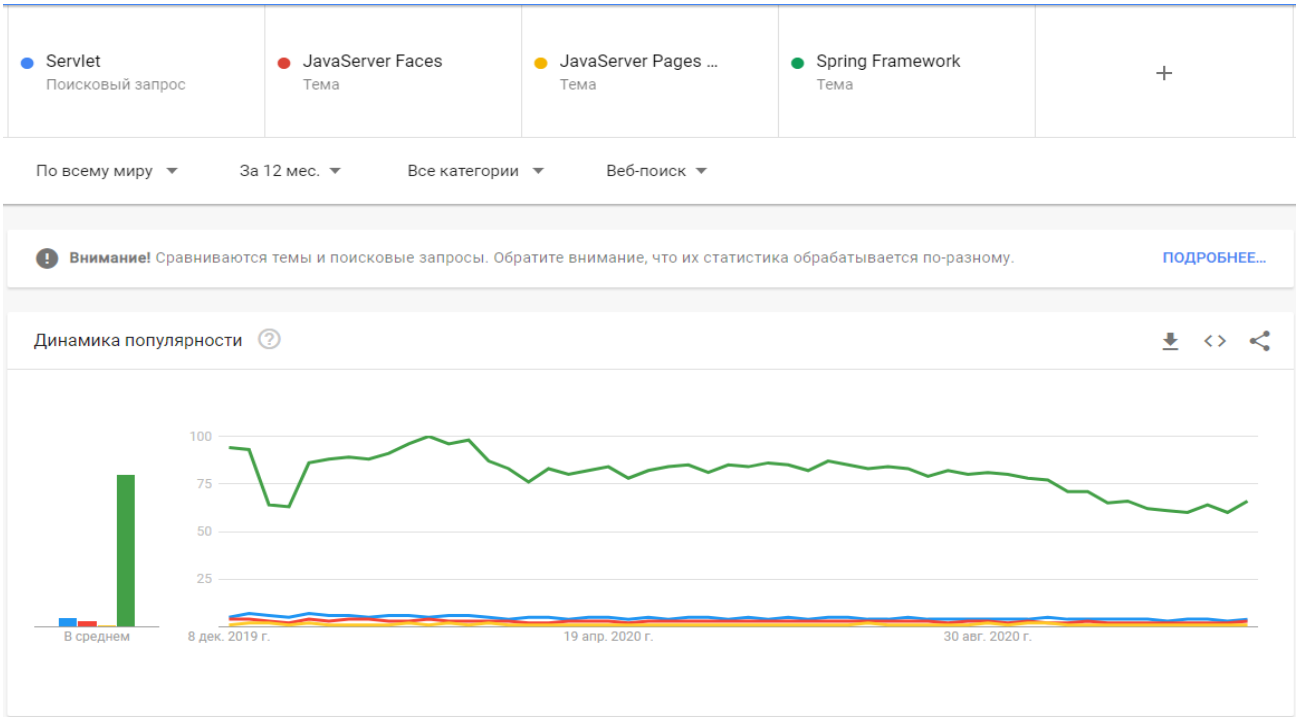


Рисунок 2.5 – Рейтинг java web технологій

2.2.2.1 Архітектура Spring Framework

Spring Framework розділений на декілька окремих модулів, що дозволяє вам вирішити, які саме використовувати у вашому додатку. Наведене нижче зображення ілюструє найважливіші модулі архітектури Spring Framework.

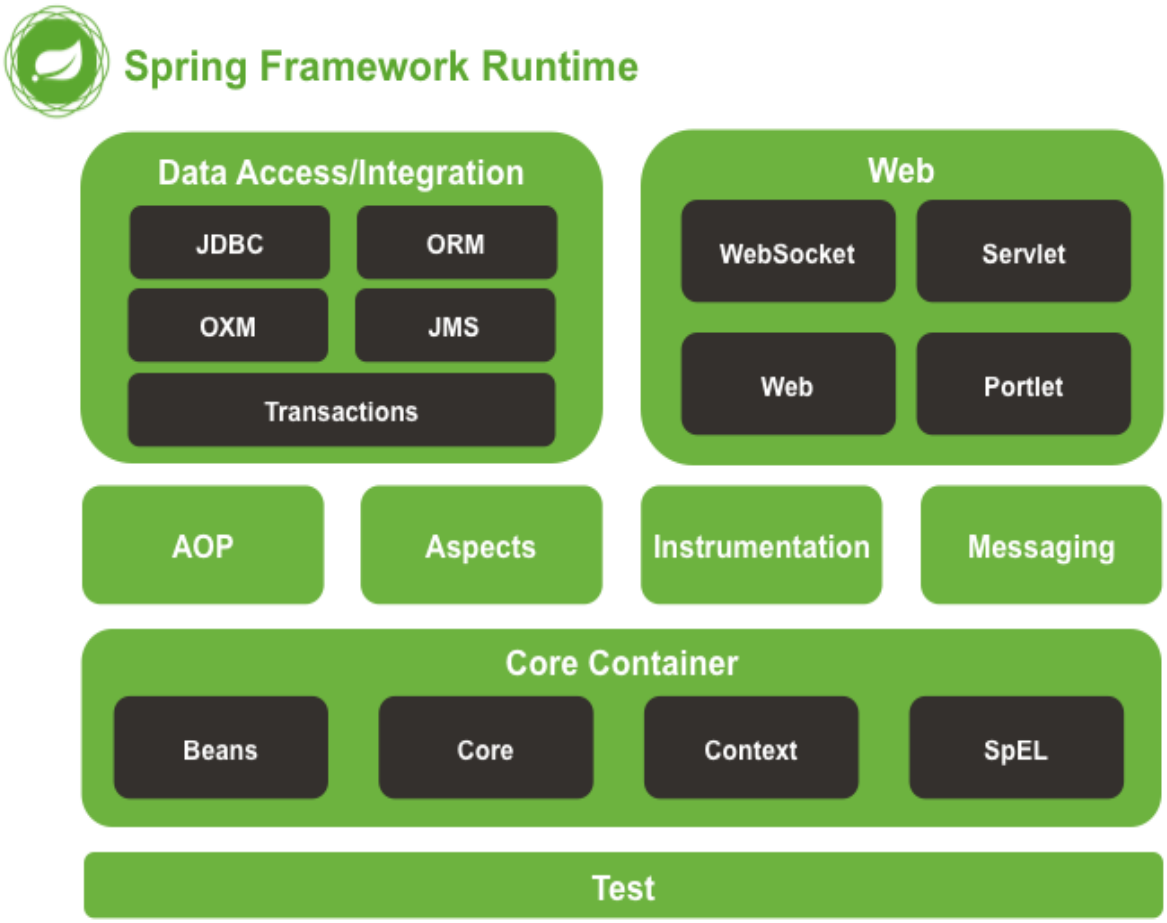


Рисунок 2.6 – Архітектура Spring Framework

2.2.2.2 Особливості Spring Framework

Особливості фреймворку Spring, такі як IoC, AOP та управління транзакціями, роблять його унікальним серед переліку фреймворків. Деякі найважливіші особливості фреймворку Spring:

- **IoC-контейнер:**

  Посилається на основний контейнер, який використовує шаблон DI або IoC для неявного посилання на об'єкт у класі під час виконання. Цей шаблон діє як альтернатива шаблону локатора послуг. Контейнер IoC містить код асемблера, який обробляє управління конфігурацією об'єктів програми.

  Структура Spring забезпечує два пакети, а саме: org.springframework.beans та org.springframework.context, що допомагає забезпечити функціональність контейнера IoC.

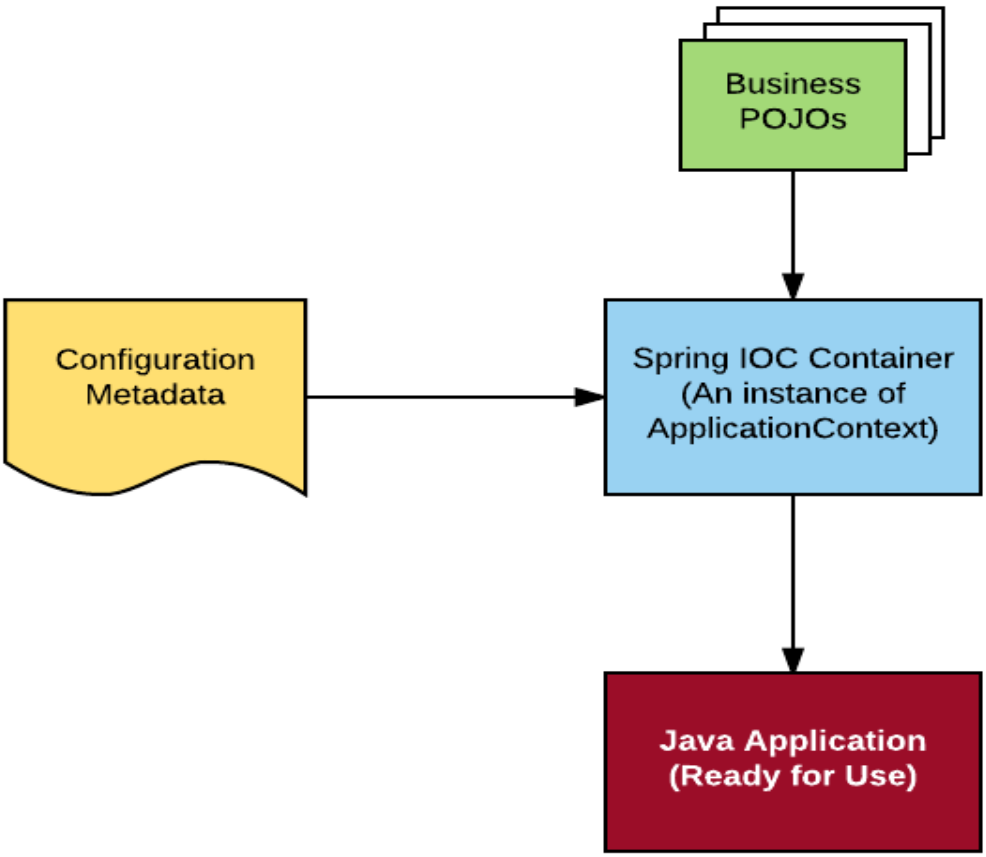


Рисунок 2.7 – Spring IoC контейнер

- **Система доступу до даних:**

  Дозволяє розробникам використовувати API, такі як JDBC та Hibernate, для зберігання даних у базі даних. Це допомагає у вирішенні різних проблем розробника, таких як взаємодія з підключенням до бази даних, перевірка що з'єднання закрито, боротьба з винятками та реалізація управління транзакціями. Також це дозволяє розробникам легко писати код для доступу до баз даних протягом усієї програми.

- **Spring MVC framework:**

  Дозволяє створювати веб-додатки на основі архітектури MVC. Всі запити, зроблені користувачем, спочатку проходять через контролер, отримують дані в моделі а потім відправляються на різні сторінки представлення. Функції обробки форм та перевірки форми в рамках Spring MVC можуть бути легко інтегровані з усіма популярними технологіями перегляду, такими як Jasper Report, FreeMarker чи Velocity.

2.2.2.3 Життєвий цикл бінів в Spring Framework

Життєвий цикл бінів в Spring Framework - одна з найважливіших особливостей, яку слід зрозуміти.

У багатьох додатках при розробці часто необхідно виконати деякі операції перед ініціалізацією компонента, і необхідно виконати деякі операції очищення, перш ніж компонент буде знищений контейнером.

У Java життєвий цикл об’єкта починається з нового ключового слова "new".

Коли ми створюємо об'єкт за допомогою "new" викликається ряд конструкторів ієрархічного класу (виклик йде знизу вгору і, отже, виконання зверху вниз)

І нарешті робить об’єкт доступним.

І коли цей об’єкт не матиме жодного посилання, він буде видалений об'єктом Garbage Collector. Це простий життєвий цикл об’єкта в Java.

Але в Spring Framework життєвомий цикл біна дещо відрізняється.

Схема життєвого циклу бінів в Spring Framework наведена нижче:

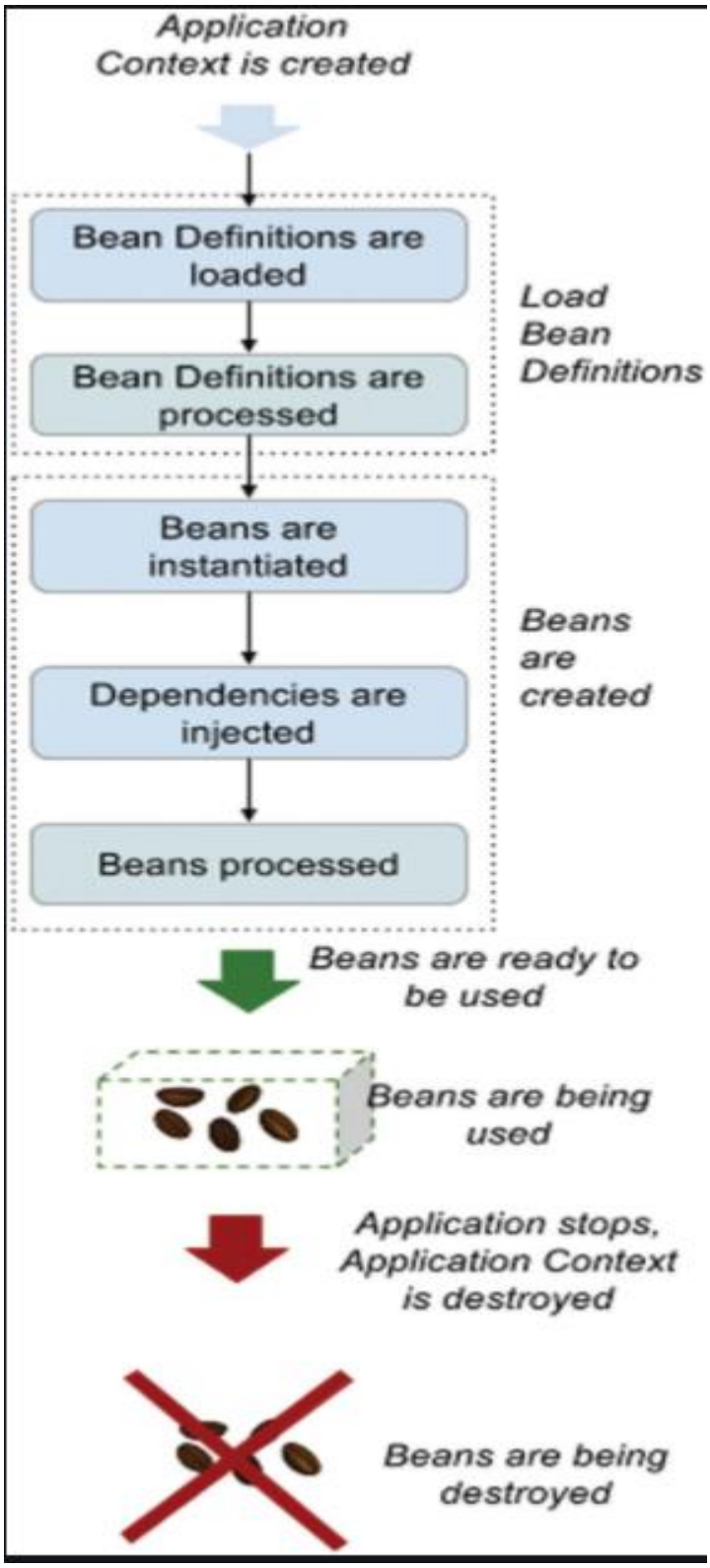


Рисунок 2.8 – Життєвий цикл бінів в Spring Framework

2.2.2.4 Spring Boot

Spring Boot забезпечує хорошу платформу для розробників Java для розробки автономного Spring додатку, який можна просто запустити. Ви можете розпочати роботу з мінімальними конфігураціями, не потребуючи цілого налаштування конфігурації Spring.

Переваги

Spring Boot пропонує своїм розробникам наступні переваги:

- Легко зрозуміти та розробляти Spring програми
- Підвищує продуктивність праці
- Скорочує час розробки
- Він надає гнучкий спосіб налаштування Java Beans, конфігурацій XML та транзакцій баз даних.
- Він забезпечує потужну пакетну обробку та керує кінцевими точками REST.
- У Spring Boot все налаштовано автоматично; ніякі ручні конфігурації не потрібні.
- Полегшує управління залежностями

Цілі

Spring Boot розроблений з наступними цілями:

- Щоб уникнути складної конфігурації XML в Spring
- Простіше розробити готові до використання програми Spring
- Скоротити час розробки та запустити програму самостійно

Spring Boot автоматично налаштовує вашу програму на основі залежностей, які ви додали до проекту за допомогою анотації @EnableAutoConfiguration. Наприклад, якщо база даних PostgreSQL знаходиться у вашому шляху до класу, але ви не налаштували жодне з’єднання з базою даних, тоді Spring Boot автоматично налаштовує базу даних у пам’яті.

Точкою входу програми є клас, що містить анотацію @SpringBootApplication та main() метод.

Spring Boot автоматично сканує всі компоненти, що входять до проекту, за допомогою анотації @ComponentScan. Управління залежністю є складним завданням для великих проектів. Spring Boot вирішує цю проблему, надаючи набір залежностей для зручності розробників. Наприклад, якщо ви хочете використовувати Spring та JPA для доступу до бази даних, достатньо, якщо ви включите у свій проект залежність spring-boot-starter-data-jpa. Для зручності в Spring Boot всі початкові завантажувачі мають однаковий шаблон імені spring-boot-starter- *, де * вказує, що це тип програми.

2.2.2 Hibernate

Hibernate - це ORM фреймворк для Java. Це потужний високоефективний об’єктно-реляційний сервіс персистентності та запитів для будь-якої програми Java.

Hibernate пов'язує класи Java із таблицями баз даних та типи даних Java з типами даних SQL і звільняє розробника від більшості загальних завдань програмування, пов'язаних із збереженням даних.

Hibernate знаходиться між традиційними об'єктами Java та сервером баз даних, щоб обробляти всі роботи щодо збереження цих об'єктів на основі відповідних механізмів та шаблоні.

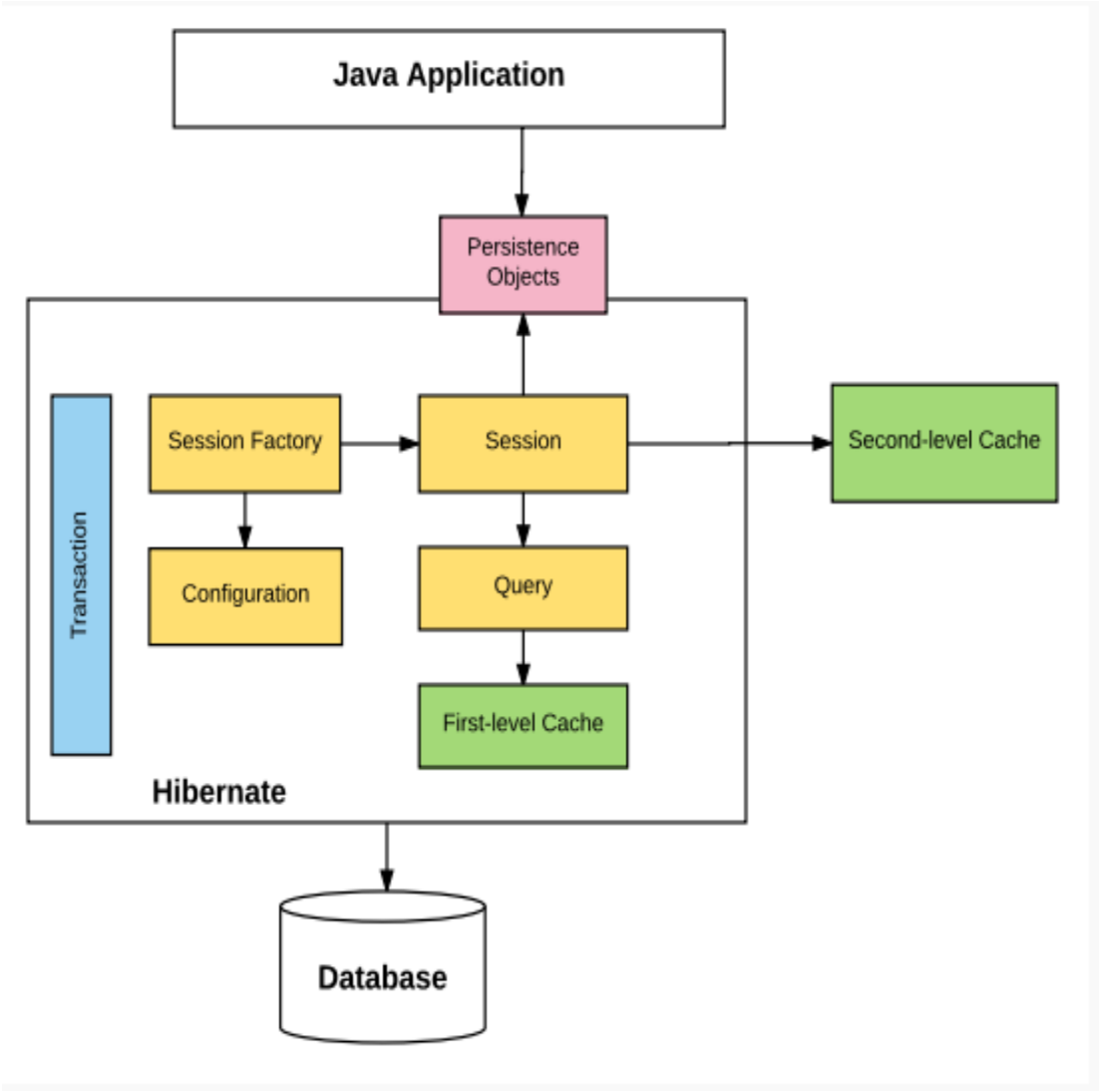


Рисунок 2.4 – Архітектура Hibernate Framework

Configuration - Зазвичай пишеться у файлах hibernate.properties або hibernate.cfg.xml. Для конфігурації Java ви можете знайти клас, анотований @Configuration. Він використовується Session Factory для роботи з Java-додатком та базою даних. Він представляє цілий набір зіставлення типів Java програми з базою даних SQL.

Session Factory - Будь-яка програма користувача запитує фабрику сеансів для об’єкта сеансу. Session Factory використовує інформацію про конфігурацію з перерахованих вище файлів для належного створення об'єкта сеансу. Проте слід пям’ятати, що Session Factory досить важкий об’єкт, тому зазвичай створюється один раз на всю програму.

Session - представляє взаємодію між додатком та базою даних у будь-який момент часу. Представлено класом org.hibernate.Session. Екземпляр сесії можна отримати з компонента SessionFactory.

Query - дозволяє програмам запитувати базу даних щодо одного або декількох збережених об'єктів. Hibernate надає різні методи запитів до бази даних, включаючи NamedQuery та Criteria API.

First-level cache - представляє кеш за замовчуванням, який використовується об’єктом Hibernate Session під час взаємодії з базою даних. Він також називається кешем сесії та кешує об'єкти в межах поточного сесії. Усі запити від об'єкта Session до бази даних повинні проходити через кеш першого рівня.

Transaction - дозволяє досягти узгодженості даних, і відкат, в випадку непредбачуваної ситуації чи помилки.

Persistent objects - Це звичайні старі об'єкти Java (POJO), які за допомогою hibernate зберігаються як один із рядків у відповідній таблиці бази даних. Вони можуть бути налаштовані у файлах конфігурацій (hibernate.cfg.xml або hibernate.properties) або анотовані як @Enity.

2.2.2.4 Особливості Hibernate

- Hibernate дбає про відображення класів Java до таблиць баз даних за допомогою XML-файлів і без написання будь-якого рядка коду.
- Надає прості API для зберігання та отримання об’єктів Java безпосередньо в базу даних та з неї.
- Якщо в базі даних або в будь-якій таблиці є зміни, вам потрібно змінити лише властивості файлу XML.
- Hibernate не вимагає роботи сервера додатків.
- Маніпулює складними асоціаціями об’єктів вашої бази даних.

- Мінімізує доступ до бази даних за допомогою розумних стратегій отримання.

- Дозволяє писати незалежні запити до бази даних на мові HQL. HQL (Hibernate Query Language) - це об’єктно-орієнтована версія SQL. Вона генерує незалежні запити до бази даних. Отже, вам не потрібно писати запити до конкретної бази даних.

- Автоматичне створення таблиці. Hibernate Framework забезпечує можливість автоматичного створення таблиць бази даних. Отже, немає необхідності створювати таблиці в базі даних вручну.

- Підтримка кешування.

2.2.2.5 Бази данних які Hibernate підтримує

- HSQL Database Engine
- DB2/NT
- MySQL
- PostgreSQL
- FrontBase
- Oracle
- Microsoft SQL Server Database
- Sybase SQL Server
- Informix Dynamic Server

2.2.2.6 Spring Data JPA та Hibernate

Хоча Hibernate і можно викростовувати як самостійний фреймворк для роботи з базами данних із Java коду, на практиці, в реальних проектах, його частіше використовують як реалізація специфікації Spring Data JPA. Spring Data JPA, яка є частиною великого сімейства Spring Data, дозволяє легко реалізувати JPA репозиторії. Цей модуль стосується розширеної підтримки рівнів доступу до даних на основі JPA та спрощує створення програм на основі Spring, що пропонують технології доступу до даних.

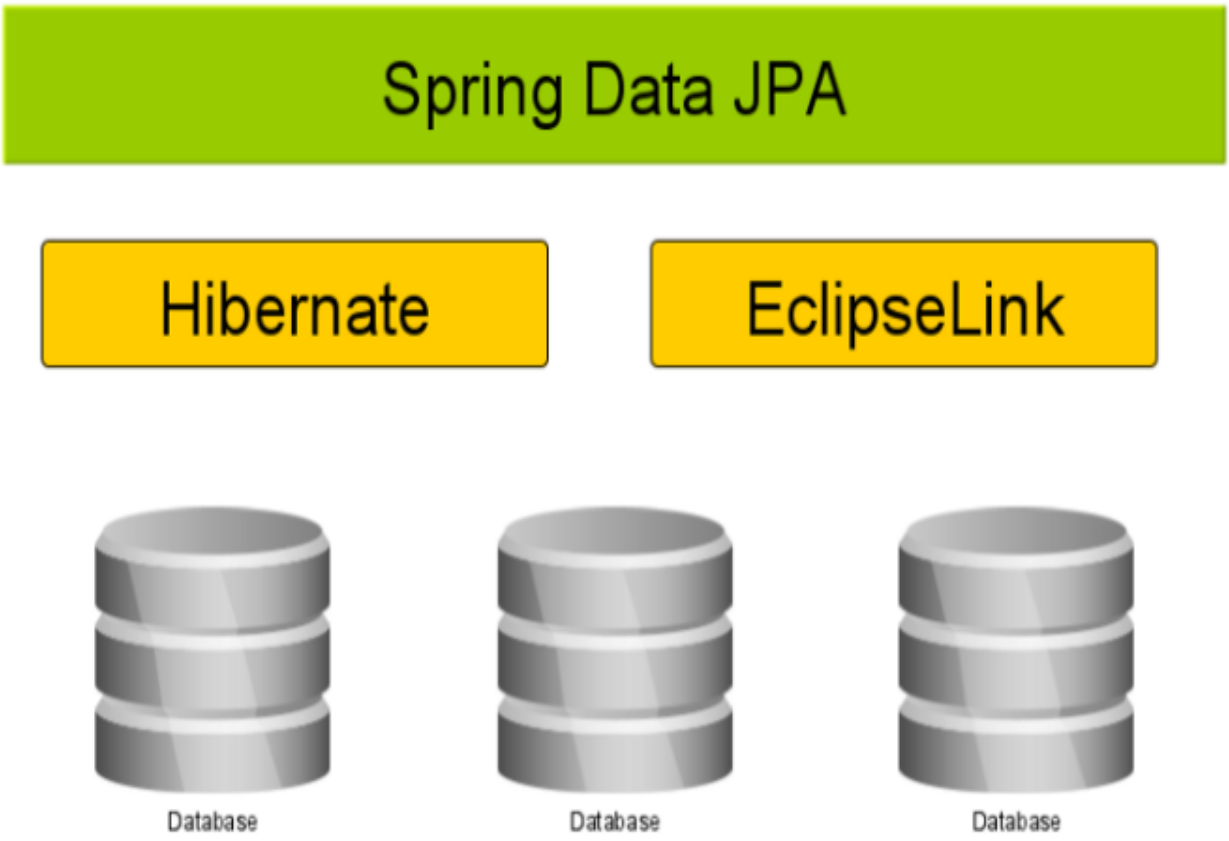


Рисунок 2.5 – Архітектура програми з використанням Spring Data JPA та Hibernate

2.2.3 Liquibase

Інструменти міграції баз даних допомагають нам відстежувати, контролювати версії та автоматизувати зміни схем баз даних. Вони допомагають нам мати узгоджену схему в різних середовищах.

Liquibase полегшує міграцію баз даних не тільки за допомогою простих старих сценаріїв SQL, але й за допомогою різних абстрактних форматів, наприклад

XML, YAML та JSON. Коли ми використовуємо не-SQL формати для міграції баз даних, Liquibase генерує для нас специфічний для конкретної бази даних SQL.

Основні складові liquibase:

- ChangeSet - це набір змін, які потрібно застосувати до бази даних. Liquibase відстежує виконання змін на рівні ChangeSet.

- Change - описує одну зміну, яку потрібно застосувати до бази даних. Liquibase пропонує кілька типів змін, таких як "створити таблицю" або "видалити колонку" з таблиці, кожна з яких є абстракцією над SQL скриптом.

- ChangeLog - файл, який містить список наборів змін бази даних, який потрібно застосувати, називається журналом змін. Ці файли журналу змін можуть бути у форматі SQL, YAML, XML або JSON.

- Preconditions - використовуються для керування changeLog або changeSets. Вони використовуються для визначення стану бази даних, для якої потрібно виконувати changeSets або changeLog.

- Context - changeSet може бути позначений виразом контексту. Liquibase оцінить цей вираз, щоб визначити, чи слід виконувати changeSet під час виконання, враховуючи конкретний контекст.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <databaseChangeLog
3.    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
4.    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5.    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
6.
    http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.8.xsd">
7.    <changeSet  author="authorName"  id="changelog-1.0">
8.      <createTable  tableName="TablesAndTables">
9.        <column  name="COLUMN1"  type="TEXT">
10.         <constraints  nullable="true"  primaryKey="false"  unique="false"/>
11.       </column>
12.     </createTable>
13.   </changeSet>
14. </databaseChangeLog>
```

Рисунок 2.5 – Прилад liquibase changeLog’у на XML

2.2.4 SQL

SQL - це структурована мова запитів, яка є комп'ютерною мовою для зберігання, обробки та отримання даних, що зберігаються в реляційній базі даних.

SQL є стандартною мовою для релаційних баз даних. Усі реляційні системи управління базами даних, такі як MySQL, MS Access, Oracle, Sybase, Informix, postgres та SQL Server, використовують SQL як стандартну мову баз даних.

Особливості SQL:

- Дозволяє користувачам доступ до даних у реляційних системах управління базами даних.
- Дозволяє користувачам описувати дані.
- Дозволяє користувачам визначати дані в базі даних та маніпулювати ними.
- Дозволяє вбудовуватись в інші мови за допомогою модулів SQL, бібліотек та пре-компіляторів.
- Дозволяє користувачам створювати та видаляти бази даних і таблиці.
- Дозволяє користувачам створювати збережені процедуру чи функції в базі даних.
- Дозволяє користувачам встановлювати дозволи на таблиці чи процедури.

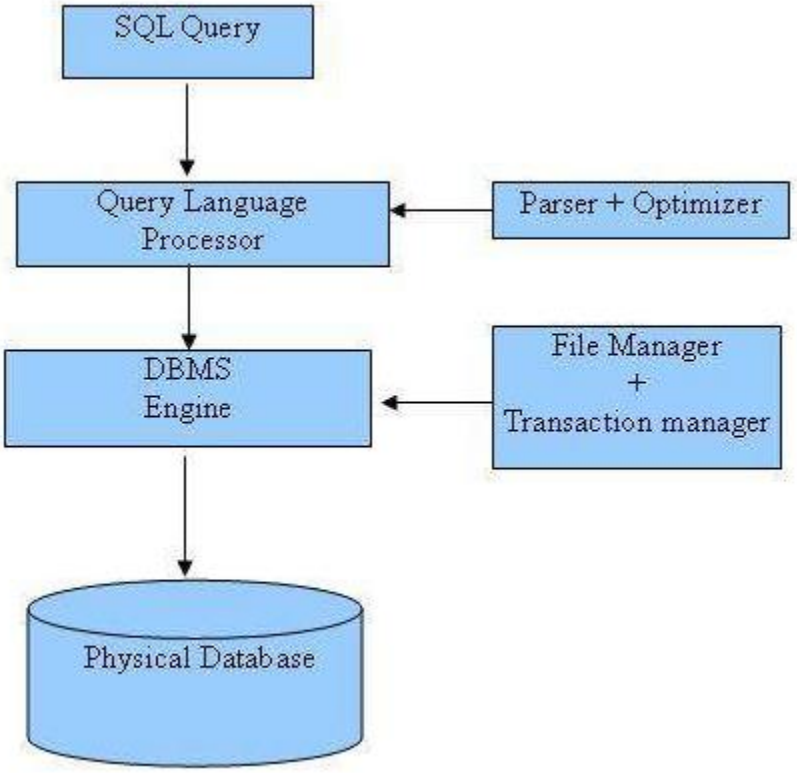


Рисунок 2.6 – SQL Архітектура

Стандартними командами SQL для взаємодії з реляційними базами даних є CREATE, SELECT, INSERT, UPDATE, DELETE і DROP. Ці команди можна класифікувати на групи залежно від їх природи:

DDL - Data Definition Language:

Команда	**Опис**
CREATE	Створює нову таблицю або інший об’єкт в базі даних
ALTER	Змінює існуючий об’єкт бази даних, наприклад таблицю.
DROP	Видаляє цілу таблицю або інший об’єкт в базі даних.

Таблиця 2.1 – DDL команди

DML - Data Manipulation Language

Команда	**Опис**
INSERT	Створює запис в таблиці.
UPDATE	Модифікує запис в таблиці.
DELETE	Видаляє запис в таблиці.

Таблиця 2.2 – DML команди

DCL - Data Control Language

Команда	**Опис**
GRANT	Видає дозволи користувачеві.
REVOKE	Забирає видані користувачеві дозволи.

Таблиця 2.3 – DCL команди

DQL - Data Query Language

Команда	**Опис**
SELECT	Отримує певні записи з однієї або декількох таблиць

Таблиця 2.4 – DQL команди

2.2.4.1 PostgreSQL

PostgreSQL - це система управління базами даних з відкритим кодом корпоративного класу. Він підтримує як SQL для реляційних, так і JSON для нереляційних запитів. Він підтримується досвідченим співтовариством розробників, які внесли величезний внесок у створення високонадійної системи СУБД.

PostgreSQL підтримує розширені типи даних та попередню оптимізацію продуктивності, функції доступні лише у дорогих комерційних базах даних, таких як Oracle та SQL Server.

Особливості PostgreSQL:

- Відповідає стандарту ANSI SQL
- Повна підтримка архітектури мережі клієнт-сервер
- Реплікація SSL на основі журналу та тригера
- Резервний сервер і висока доступність
- Об'єктно-орієнтована та сумісна з ANSI-SQL2008
- Підтримка JSON дозволяє зв'язуватися з іншими сховищами даних, такими як NoSQL, які виступають як об'єднаний хаб для баз даних поліглотів.

Рисунок 2.7 – лого PostgreSQL

Порівняння PostgreSQL та MySQL

MYSQL	POSTGRESQL
Проект MySQL зробив доступним свій вихідний код відповідно до умов GNU License та інших патентних угод.	PostgreSQL випускається під ліцензією Postgresql.
Зараз він належить корпорації Oracle і пропонує кілька платних видань.	Це безкоштовне програмне забезпечення з відкритим кодом. Це означає, що вам ніколи не доведеться нічого платити за цю послугу.
MySQL відповідає ACID лише при використанні з механізмами кластерного зберігання NDB та InnoDB	PostgreSQL - повністю сумісний з ACID.
MySQL добре працює в системах OLAP та OLTP, де важлива лише швидкість читання.	Виконання PostgreSQL найкраще працює в системах, які вимагають виконання складних запитів.
MySQL надійний і добре працює з програмами BI (Business Intelligence), які важко читати	PostgreSQL добре працює з програмами BI. Однак він більше підходить для зберігання даних та додатків для аналізу даних, яким потрібна швидка швидкість читання-запису.

Таблиця 2.5 – Порівняння MySQL та PostgreSQL

2.2.5 Apache Maven

Maven - це інструмент автоматизації збірки, який в основному використовується для проектів Java, але, маючи невелику конфігурацію, Maven також можна використовувати для створення та управління проектами, написаними на Kotlin, Groovy, Scala та інших мовах програмування. Maven настільки універсальний, що ви можете використовувати його для автоматизації таких завдань, як управління та завантаження залежностей, розміщення додаткових jar в classpath, компіляція вихідного коду в байт-код, запуск тестів, упаковка скомпільованого коду в такі артефакти, як JAR, WAR, та розгортання цих артефакти до сервера додатків або репозиторія.

Особливості Maven:

- Усі популярні IDE для платформи Java підтримують Apache Maven з безліччю готових функцій
- Дуже поширений для використання у великих компаніях та проектах з відкритим кодом (тобто String Framework, Spring Boot тощо)
- Можливість швидкого налаштування проекту
- Управління старими залежностями та життєвий цикл побудови проекту
- Надійна спільнота плагінів; Окрім стандартних плагінів Maven, ряд проектів надають власні плагіни для Maven. Наприклад, Spring-Boot має плагін Maven, який надає певні особливості та функціональність у процесі збірки
- Усі залежності, які вимагає проект, можна автоматично завантажити, лише прочитавши файл pom.xml.
- Безперервні збірки, інтеграція та тестування, це все можна використувати зручніше за допомогою maven.

2.2.5.1 Maven репозиторії

У сховищі в Maven зберігаються артефакти побудови та залежності різних типів. Існує два типи сховищ:

- Local - локальні сховища на вашому комп'ютері. Зберігається в папці / .m2 /
- Remote - місцеположення можуть бути загальнодоступними (спільнота Maven, JBoss, Oracle, Atlassian, Google Android,…) або приватними (розміщуються компаніями для внутрішніх артефактів)

Процес отримання залежностей які вимагає проект проходить за наступним сценарієм: спочатку maven загляне всередину локального сховища (каталог /.m2/ на комп'ютері користувача), і якщо його там не знайдеться, Maven почне пошук в зовнішніх репозиторіях. За замовчуванням це буде залежності будуть шукатитись https://mvnrepository.com/repos/central, але це можна змінити в конфігурації. Ця діаграма ілюструє цей процес:

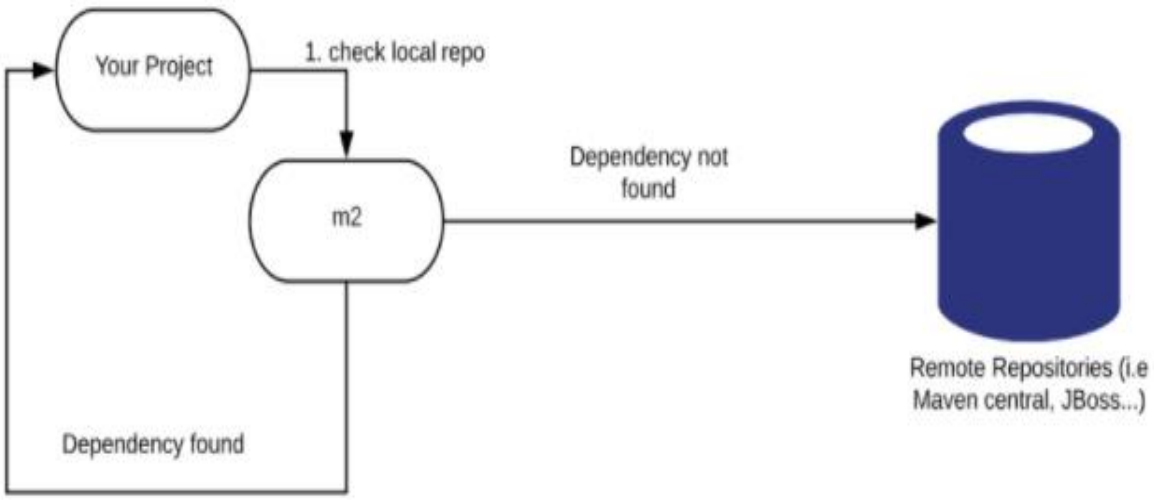


Рисунок 2.8 – Процес пошуку залежностей в Maven

2.2.6 Intellij IDEA

IntelliJ IDEA - одна з найпотужніших та найпопулярніших інтегрованих середовищ розробки (IDE) для Java. Вона розроблена і підтримується JetBrains і доступна Community(безкоштовна) та Ultimate(платна) версії. Ця IDE забезпечує швидкий розвиток та допомагає поліпшити якість коду.

IDE розшифровується як Інтегроване середовище розробки. Це комбінація безлічі інструментів, які роблять процес розробки програмного забезпечення простішим, надійнішим та менш схильним до помилок. Він має наступні переваги перед редактором простого тексту:

- Інтеграція з такими корисними інструментами, як компілятор, дебагер, система контролю версій, інструменти збірок, різні фреймворки, профайлери додатків тощо.

- Підтримує навігацію кодом, завершення коду, рефакторинг коду та функції генерації коду, що пришвидшує процес розробки.

- Підтримує модульне тестування, інтеграційне тестування та перевірку якості коду за допомогою різнів плагінів.

- Надає багатий набір плагінів для подальшого покращення функціональності IDE.

IntelliJ IDEA має деякі найпродуктивніші функції заповнення коду Java. Його алгоритм прогнозування може точно припустити, що програміст намагається ввести, і виконає це для нього, навіть якщо він не знає точної назви певного класу, методу чи будь-якого іншого ресурсу.

Перелік особливостей Intellij IDEA:

- ***Smart code completion*** - підтримує доповнення коду на основі контексту. Зберігає перелік найбільш відповідних символів, що застосовуються в поточному контексті.

- ***Chain code completion*** - це вдосконалена функція заповнення коду, яка перелічує застосовні символи, доступні за допомогою методів або геттерів у поточному контексті.

- ***Static member's completion*** - дозволяє використовувати статичні методи або константи та автоматично додає необхідні оператори імпорту, щоб уникнути помилки компіляції.

- ***Detecting duplicates*** - знаходить фрагменти коду, що повторюється, на льоту і дає сповіщення або пропозицію щодо цього користувачеві.

- ***Inspections and quick-fixes*** - щоразу, коли IntelliJ виявляє, що ви допустили синтаксичну помилку, на цьому ж рядку з'являється невелике сповіщення та іконки лампочки. Клацнувши на яку, відображається список пропозицій як виправити помилку.

Щоб допомогти розробникам організувати робочий процес, IntelliJ IDEA пропонує їм широкий набір інструментів, який складається з декомпілятора, підтримки Docker, засобу перегляду байт-кодів, FTP та багатьох інших інструментів:

- Контроль версій - IntelliJ підтримує більшість популярних систем контролю версій, таких як Git, Subversion, Mercurial, CVS, Perforce та TFS.

- Інструменти збірок - IntelliJ підтримує Java та інші інструменти збірок, такі як Maven, Gradle, Ant, Gant, SBT, NPM, Webpack, Grunt та Gulp.

- Різні інструменти тестування та аналізатори коду - IntelliJ IDEA дозволяє легко виконувати модульне тестування. IDE включає інструменти покриття для основних тестових платформ, включаючи JUnit, TestNG, Spock, Cucumber, ScalaTest, spec2 та Karma.

- Декомпілятор - IntelliJ постачається із вбудованим декомпілятором для класів Java. Якщо ви хочете заглянути всередину бібліотеки, для якої у вас немає вихідного коду, ви можете зробити це, не використовуючи сторонні плагіни.

- Термінал - IntelliJ забезпечує вбудований термінал. Залежно від вашої платформи, ви можете працювати з командним рядком, наприклад PowerShell або Bash.

- Інструменти баз даних - IntelliJ надає інструменти баз даних, які дозволяють підключатися до активних баз даних; виконувати запити; переглядати та оновлювати дані; і навіть керувати своїми схемами у візуальному інтерфейсі від самої IDE.

- Сервер додатків - IntelliJ підтримує основні сервери додатків: Tomcat, JBoss, WebSphere, WebLogic, Glassfish та багато інших. Ви можете розгорнути свої артефакти на серверах програм і налагодити розгорнуті програми в самій IDE.

- Підтримка Docker - За допомогою окремого плагіна IntelliJ надає спеціальне вікно інструментів, яке дозволяє підключатися до локально працюючих контейнерів Docker.

2.2.6.1 Порівняння Ultimate та Community версій

Функція	**Ultimate версія**	**Community версія**
Ліцензія	Платна	З відкритим кодом, Apache 2.0. для комерційного розвитку.
Java, Kotlin, Groovy, Scala	Підтримується	Не підтримується
Android development	Підтримується	Підтримується
Maven, Gradle, SBT	Підтримується	Підтримується
Git, SVN, Mercurial, CVS	Підтримується	Підтримується
Detecting Duplicates	Підтримується	Не підтримується
Perforce, TFS	Підтримується	Не підтримується
JavaScript, TypeScript	Підтримується	Не підтримується
Java EE, Spring, GWT, Vaadin, Play, Grails, Other Frameworks	Підтримується	Не підтримується
Database Tools, SQL	Підтримується	Не підтримується

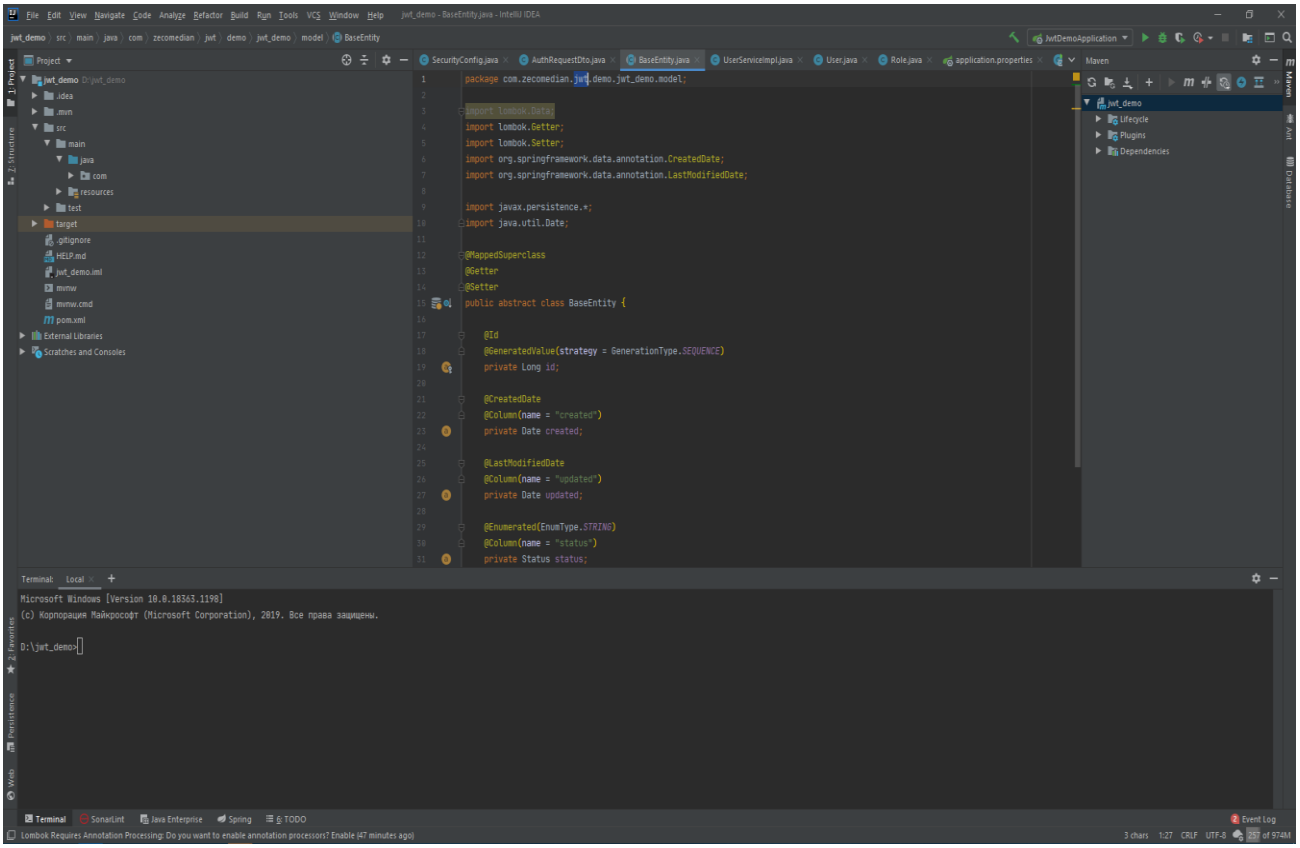


Рисунок 2.9 – Інтерфейс Intellij IDEA Ultimate

Висновки до другого розділу

В цьому розділі було розглянуто архітектуру систему розробленої при виконанні дипломної. Показана основна інформація про мову програмування Java, її особливості, та причини популярності.

Досліджені такі технології як Spring Framewok, найбільш попопулярний фреймворк на сьогодні, якій використовується майже на всіх комерційних проектах написаних на мові програмування Java. Hibernate, фреймворк якій суттєво полегшує роботу з реляційними базами даних.

Крім того, так як при розробці проектів бази даних часто модифікуються(додаються або видаляються таблиці, додаються або змінюються колонки в таблиці, тощо), було також розглянуто технологію liquibase, одну із небагатьох технологій за допомогою якої можна зручно мігрувати дані в реляційних базах даних.

Також в цьому розділі я розглянув середовище розробки Intellij IDEA, оскільки це найзручніша на сьогодні IDE, для java розробника, з якою мені доводиться працювати майже 100% часу на роботі.

3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Скачати Telegram

Першим кроком при розробці Telegram чат-боту, є завантаження та інсталяція самої програми “Telegram”.

Для цього переходимо на сайт https://desktop.telegram.org та завантажуємо файл в відповідному форматі. В моєму випадку .exe файл.

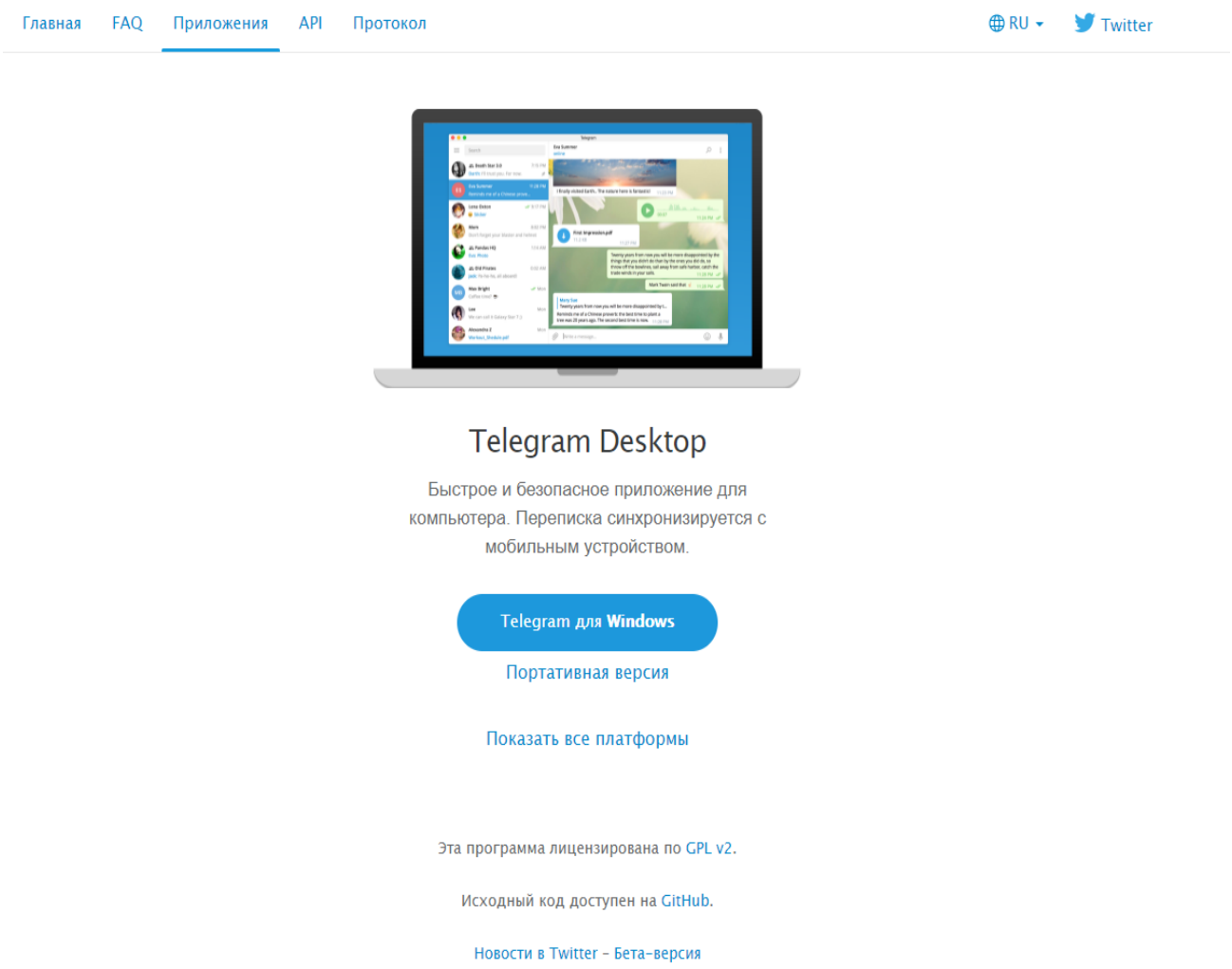


Рисунок 3.1 – Офіційний сайт Telegram

Після завантаження, запускаємо файл та слідуємо інструкції.

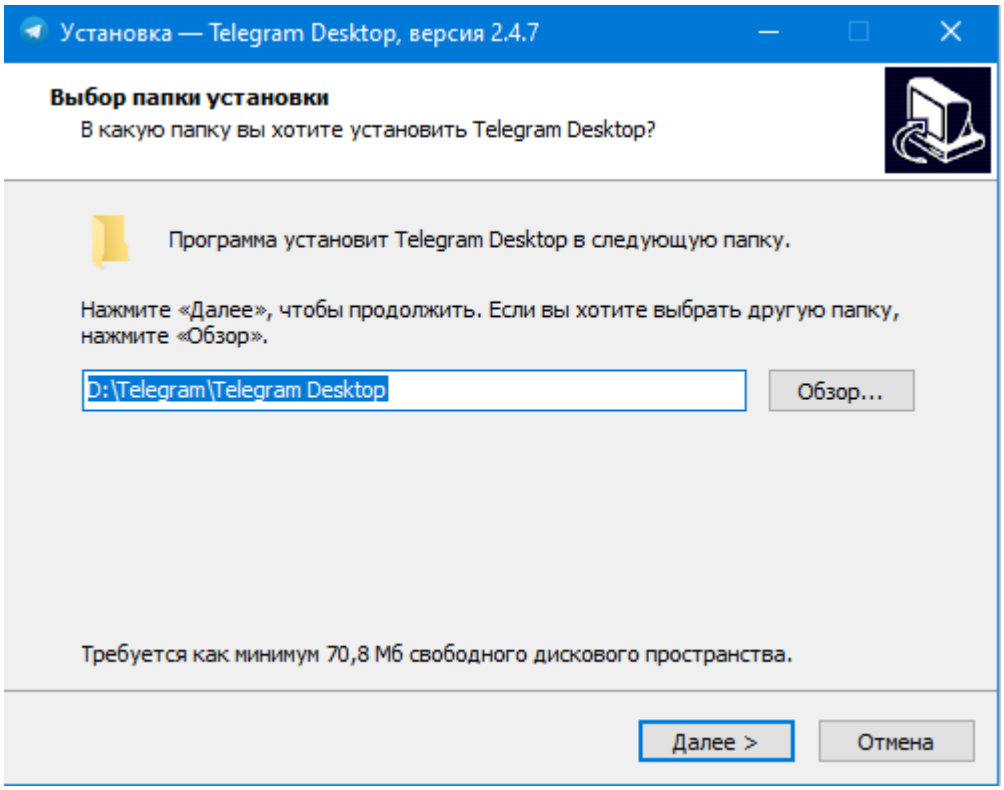


Рисунок 3.2 – Процес інсталяції Telegram на ОС Windows 10

3.2 Реєстрація чат-боту

Перед початком розробки чат-боту, його потрібно створити та налаштувати. За створення чат-ботів для в програмі Telegram, відповідає спеціальний бот з ім’ям BotFather, щоб його знайти достатньо ввести в пошук @BotFather

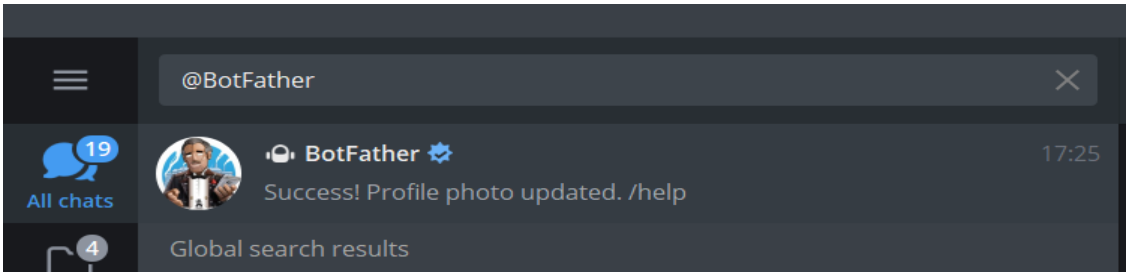


Рисунок 3.2 – Пошук боту @BotFather

Для початку роботи з ботом @BotFather, необхідно ввести команду “/start”, після відправки якої, бот поверне вам інструкції по створенню власних чат ботів.

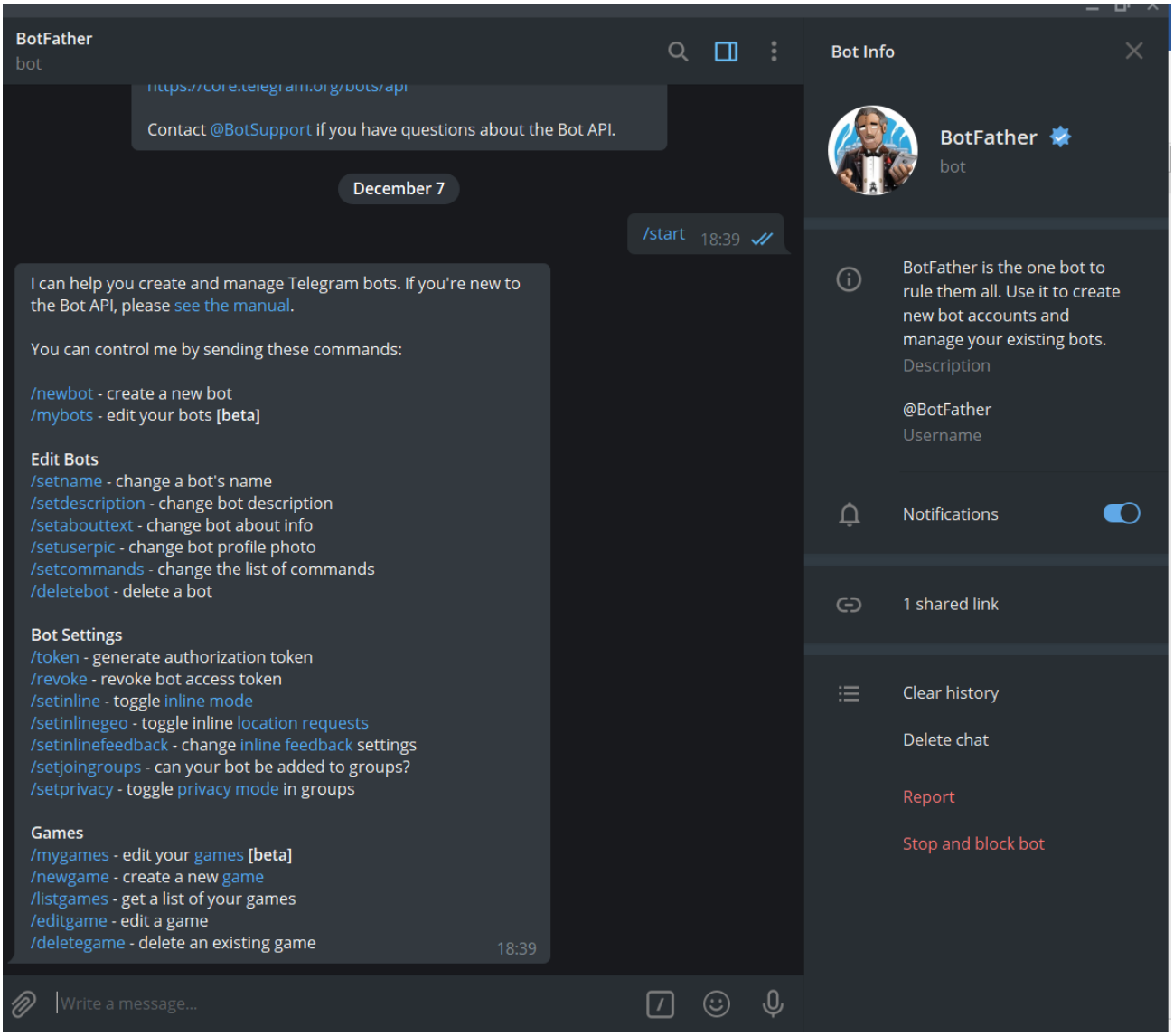


Рисунок 3.2 – Початок роботи з @BotFather

Оскільки ми хочемо створити нового чат-бота, наступна команда яку посилаємо *@BotFather* – “/newbot”, після чого задаємо нашому боту ім’я, нікнейм, та по бажанню аватар.

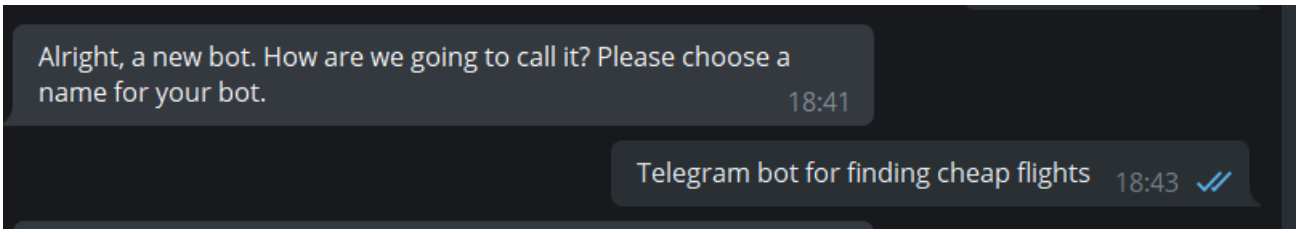


Рисунок 3.3 – Створення ім’я для бота

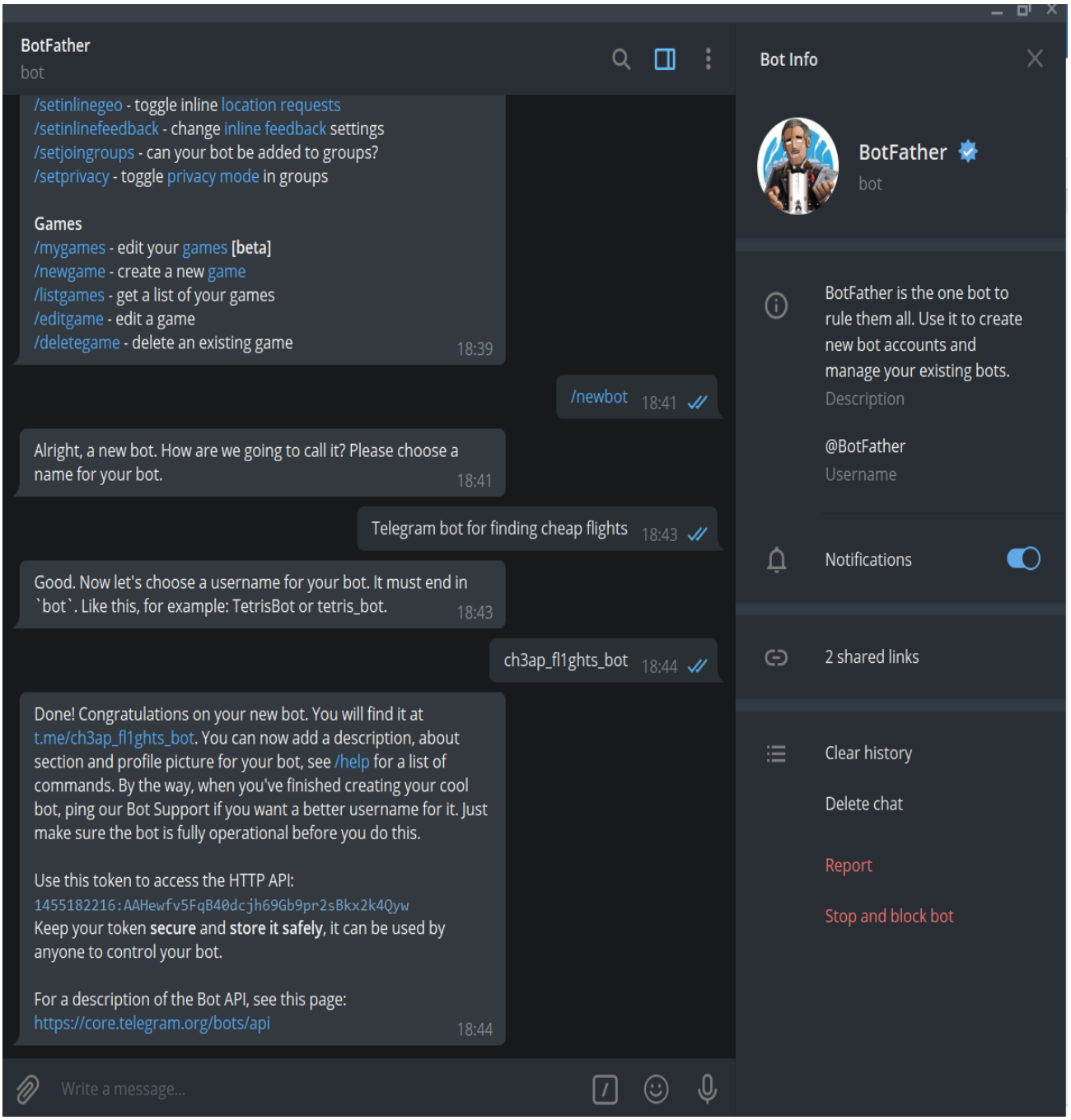


Рисунок 3.3 – Створення username боту, та завершення його реєстрації

За допомогою команд “/setuserpic’ та “setabouttext”, додав аватар та опис боту, в результаті він виглядає так:

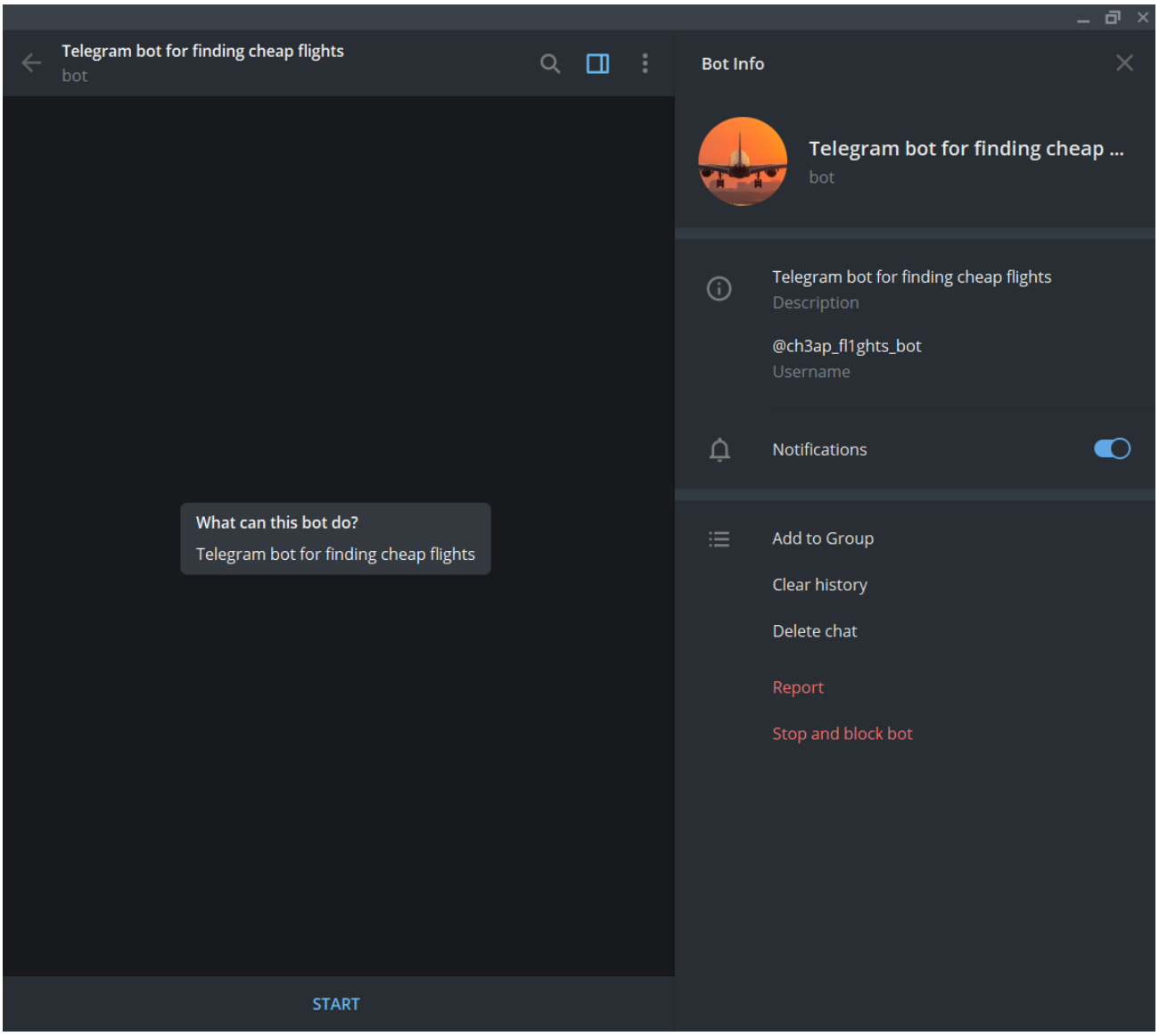


Рисунок 3.4 – Інтерфейс створеного боту

3.3 Реалізація чат-боту

Під час розробки проекту була використана багаторівнева структура проекту. Виділено наступні рівні:

Config – для конфігурації (Spring, Data Source, Telegram API…)

Controller – для створення методів що перехоплюють HTTP запити, оброблюють їх та повертаються в якості відповіді.

Service – рівень який відповідає за бізнес-логіку.

Model – рівень на якому зберігаються об’єкти, наприклад User.

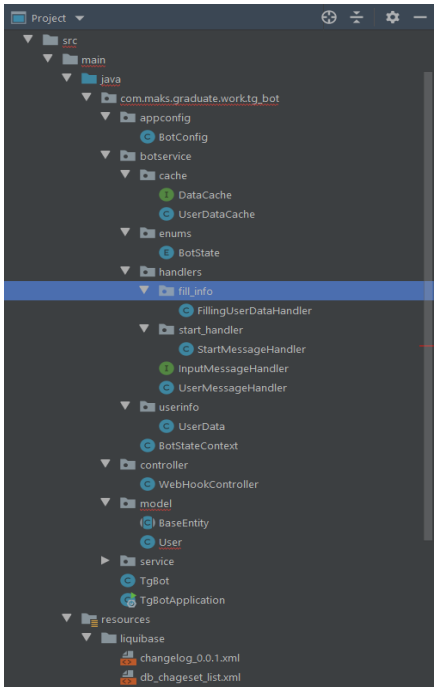


Рисунок 3.5 – Структура проекту

3.3.1 Класс BotConfig

Даний клас за допомогою аннотацій @Configuration та @Bean конфігурує біни які потім під час ініціалізації будуть поклдені в Spring контекст, та будуть доступні в будь якому місці коду.

```
@Getter
@Setter
@Configuration
@ConfigurationProperties(prefix = "telegrambot")
public class BotConfig {

    private String webHookPath;
    private String botUserName;
    private String botToken;

    private DefaultBotOptions.ProxyType proxyType;
    private String proxyHost;
    private int proxyPort;

    @Bean
    public TgBot tgBot(UserMessageHandler userMessageHandler){
        DefaultBotOptions options = ApiContext.getInstance(DefaultBotOptio

        options.setProxyHost(proxyHost);
        options.setProxyPort(proxyPort);
        options.setProxyType(proxyType);

        TgBot tgBot = new TgBot(options, userMessageHandler);
        tgBot.setBotUserName(botUserName);
        tgBot.setBotToken(botToken);
        tgBot.setWebHookPath(webHookPath);

        return tgBot;
    }

    @Bean
    public MessageSource messageSource(){
        ReloadableResourceBundleMessageSource messageSource
                = new ReloadableResourceBundleMessageSource();

        messageSource.setBasename("classpath:messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }
}
```

Рисунок 3.6 – BotConfig клас

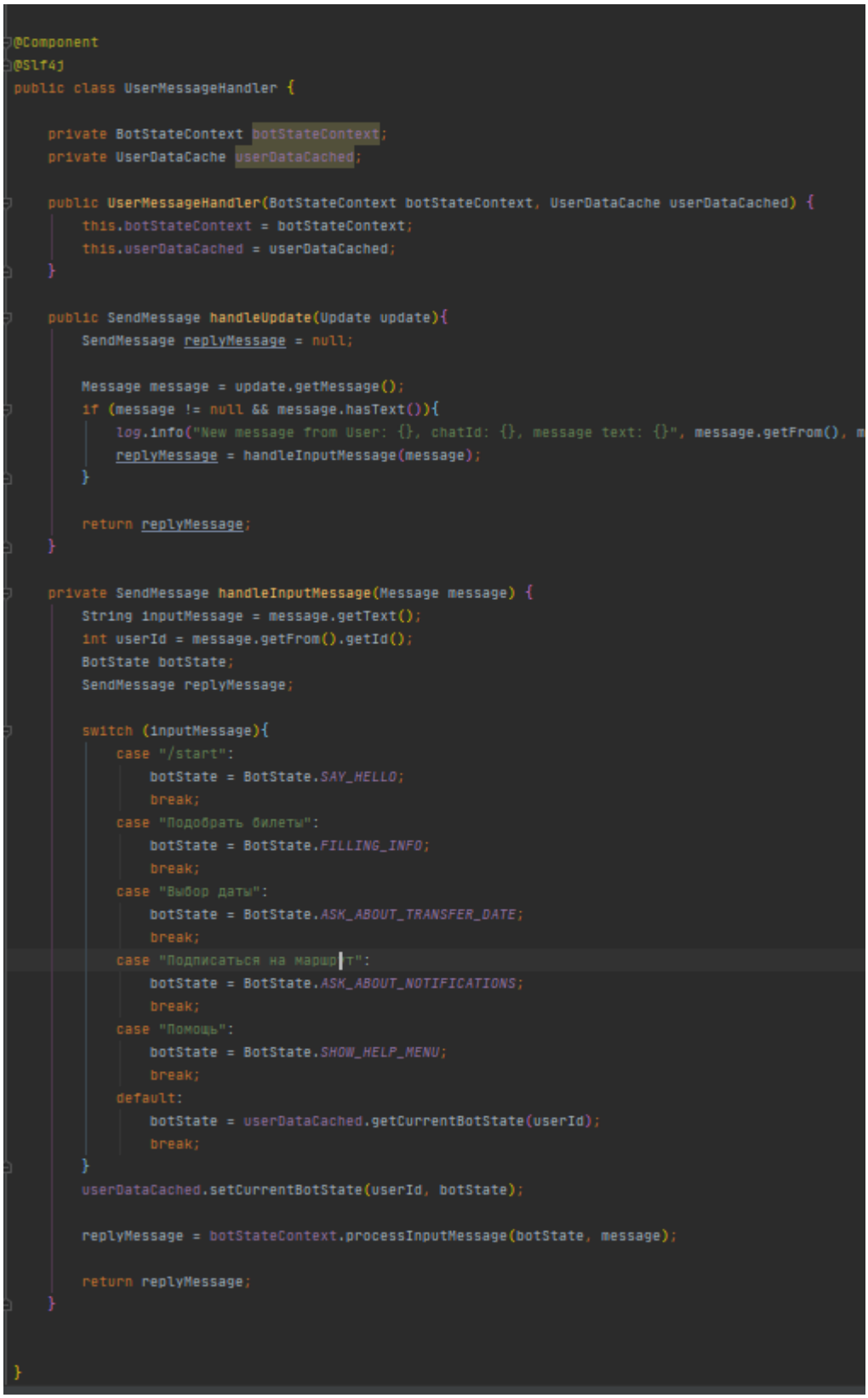
3.3.2 Класс UserMessageHandler

Цей класс відповідає за обробку повідомлень від користувача, виклиється в контроллері WebHookController:

```
@RestController
public class WebHookController {

    private final TgBot tgBot;

    @Autowired
    public WebHookController(TgBot tgBot){
        this.tgBot = tgBot;
    }

    @RequestMapping(value = "/", method = RequestMethod.POST)
    public BotApiMethod<?> onUpdateReceived(@RequestBody Update update){
        return tgBot.onWebhookUpdateReceived(update);
    }
}
```

Рисунок 3.7 – WebHookController клас

UserMessageHandler в свою чергу ствою replyMessage (повідомлення яке буде відправлене користувачеві), та встановлює новий статус BotStateContext.



```java
@Component
@Slf4j
public class UserMessageHandler {

    private BotStateContext botStateContext;
    private UserDataCache userDataCached;

    public UserMessageHandler(BotStateContext botStateContext, UserDataCache userDataCached) {
        this.botStateContext = botStateContext;
        this.userDataCached = userDataCached;
    }

    public SendMessage handleUpdate(Update update){
        SendMessage replyMessage = null;

        Message message = update.getMessage();
        if (message != null && message.hasText()){
            log.info("New message from User: {}, chatId: {}, message text: {}", message.getFrom(), m
            replyMessage = handleInputMessage(message);
        }

        return replyMessage;
    }

    private SendMessage handleInputMessage(Message message) {
        String inputMessage = message.getText();
        int userId = message.getFrom().getId();
        BotState botState;
        SendMessage replyMessage;

        switch (inputMessage){
            case "/start":
                botState = BotState.SAY_HELLO;
                break;
            case "Подобрать билеты":
                botState = BotState.FILLING_INFO;
                break;
            case "Выбор даты":
                botState = BotState.ASK_ABOUT_TRANSFER_DATE;
                break;
            case "Подписаться на марш т":
                botState = BotState.ASK_ABOUT_NOTIFICATIONS;
                break;
            case "Помощь":
                botState = BotState.SHOW_HELP_MENU;
                break;
            default:
                botState = userDataCached.getCurrentBotState(userId);
                break;
        }
        userDataCached.setCurrentBotState(userId, botState);

        replyMessage = botStateContext.processInputMessage(botState, message);

        return replyMessage;
    }


}
```

Рисунок 3.7 – UserMessageHandler клас

3.3.3 Пакет Model

Даний пакет містить класси User та BaseEnity. Перший був створений для зберігання данних про користувача, результати його вибору. Містить наступні поля:

username – ім’я користувача

routId – айді вибраного маршруту

isNotificationEnabled – чи підписався користувач на отримання новин про зміни ціни на маршрут.

BaseEnity є батьківським класом для класу User, та містить id, та так звані “аудит поля”:

created – коли було створено запис.

updated – коли було модифіковано запис.

```
@Entity
@Table(name = "user")
@Getter
@Setter
public class User extends BaseEntity{

    @Column(name = "username")
    private String username;

    @Column(name = "route_id")
    private UUID routId;

    @Column(name = "notification_enabled")
    private Boolean isNotificationEnabled = false;

}
```

Рисунок 3.8 – User клас

```java
@MappedSuperclass
@Getter
@Setter
public abstract class BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @CreatedDate
    @Column(name = "created")
    private Date created;

    @LastModifiedDate
    @Column(name = "updated")
    private Date updated;
}
```

Рисунок 3.9 – BaseEnity клас

Всі перераховні вище поля будуть зберігатись в базі даних в таблиці “user”. В якості бази даних було вибрано PostgreSQL, для підключення до якої в pom.xml додана наступна залежність:

```xml
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.16</version>
</dependency>
```

Рисунок 3.10 – PostgreSQL Maven підключення

Деталі для підключення до PostgreSQL, винесені в файл application.properties, та мають наступний вигляд:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/jwt_demo?serverTimezone=UTC
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.username=postgres
spring.datasource.password=0959
```

Рисунок 3.11 – PostgreSQL деталі підключення

Для управління міграціями баз даних, було підключено liquibase, деталі підключення якого також винесені в в файл application.properties:

```
spring.liquibase.change-log=classpath:liquibase/db_chageset_list.xml
spring.liquibase.url=jdbc:postgresql://localhost:5432/jwt_demo?serverTimezone=UTC
spring.liquibase.user=postgres
spring.liquibase.password=0959
```

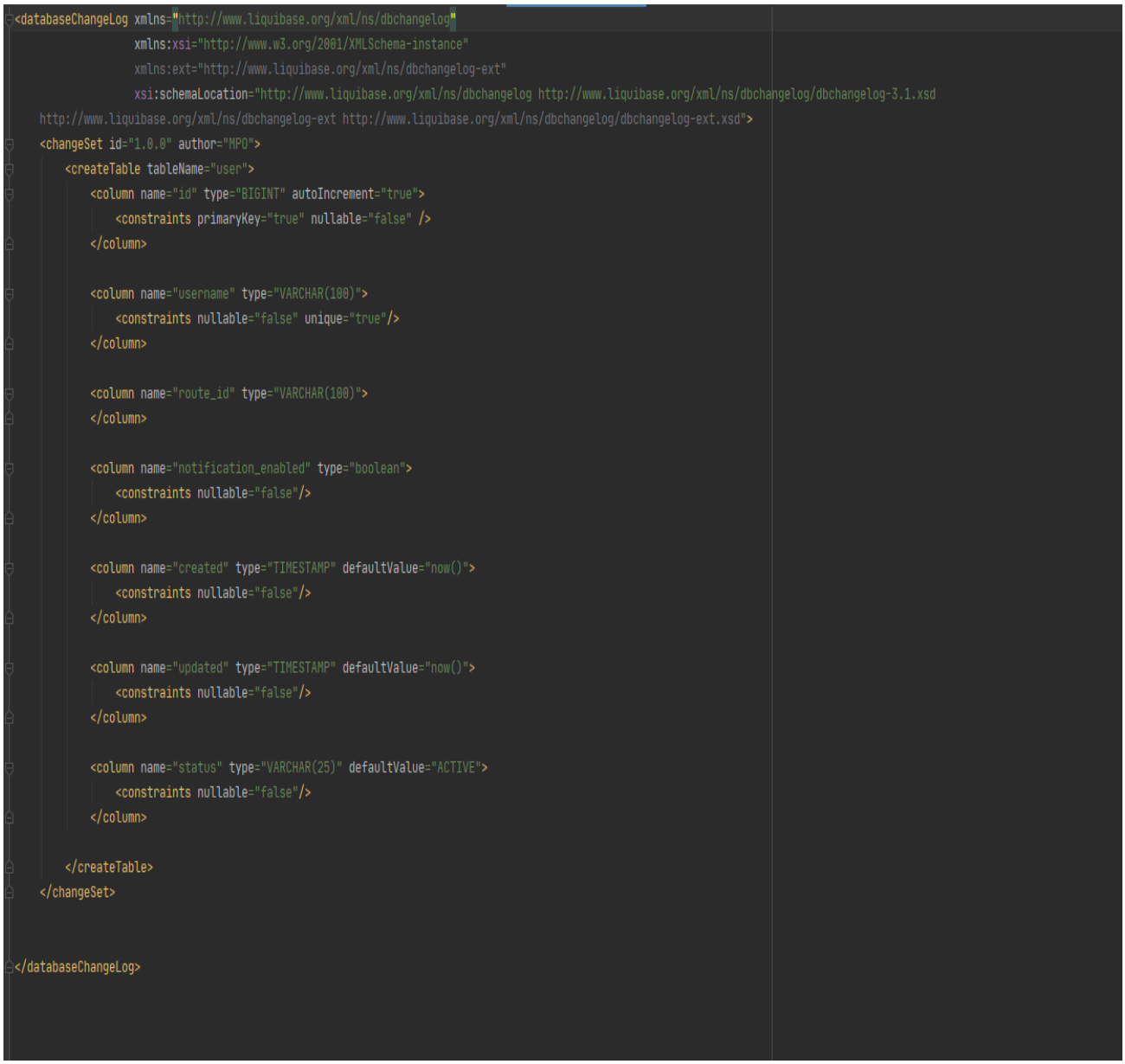
Рисунок 3.11 – liquibase деталі підключення

Для створення таблиці user в базі даних був створений файл db_changeset_list.xml, який зберігає список всіх changelog скриптів

```
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
                   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd
    http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">

    <include file="classpath:liquibase/changelog_0.0.1.xml"/>

</databaseChangeLog>
```

Рисунок 3.11 – db_changeset_list.xml

В файлі changelog_0.0.1.xml,описані зміни які повинні відбутися з таблицею “user” в базі даних:



```xml
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
                   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd
    http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">
    <changeSet id="1.0.0" author="MPO">
        <createTable tableName="user">
            <column name="id" type="BIGINT" autoIncrement="true">
                <constraints primaryKey="true" nullable="false" />
            </column>

            <column name="username" type="VARCHAR(100)">
                <constraints nullable="false" unique="true"/>
            </column>

            <column name="route_id" type="VARCHAR(100)">
            </column>

            <column name="notification_enabled" type="boolean">
                <constraints nullable="false"/>
            </column>

            <column name="created" type="TIMESTAMP" defaultValue="now()">
                <constraints nullable="false"/>
            </column>

            <column name="updated" type="TIMESTAMP" defaultValue="now()">
                <constraints nullable="false"/>
            </column>

            <column name="status" type="VARCHAR(25)" defaultValue="ACTIVE">
                <constraints nullable="false"/>
            </column>

        </createTable>
    </changeSet>


</databaseChangeLog>
```

Рисунок 3.12 – changelog_0.0.1.xml

3.3.4 Результат роботи програми

Для початку роботи з телеграм ботом потрібно ввести команду “/start”, після якої потрібно буде ввести пункт вильоту, а потім пункт призачення. Результат роботи показано на рисунках 3.13-3.14.

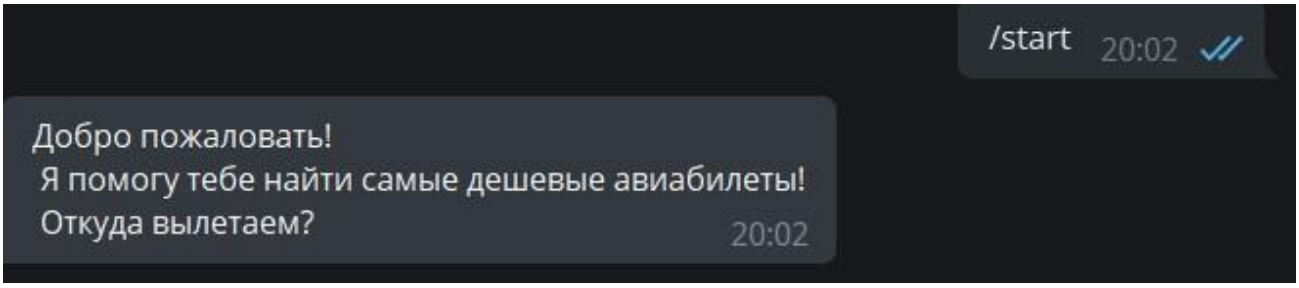


Рисунок 3.13 - Результат роботи команды /start

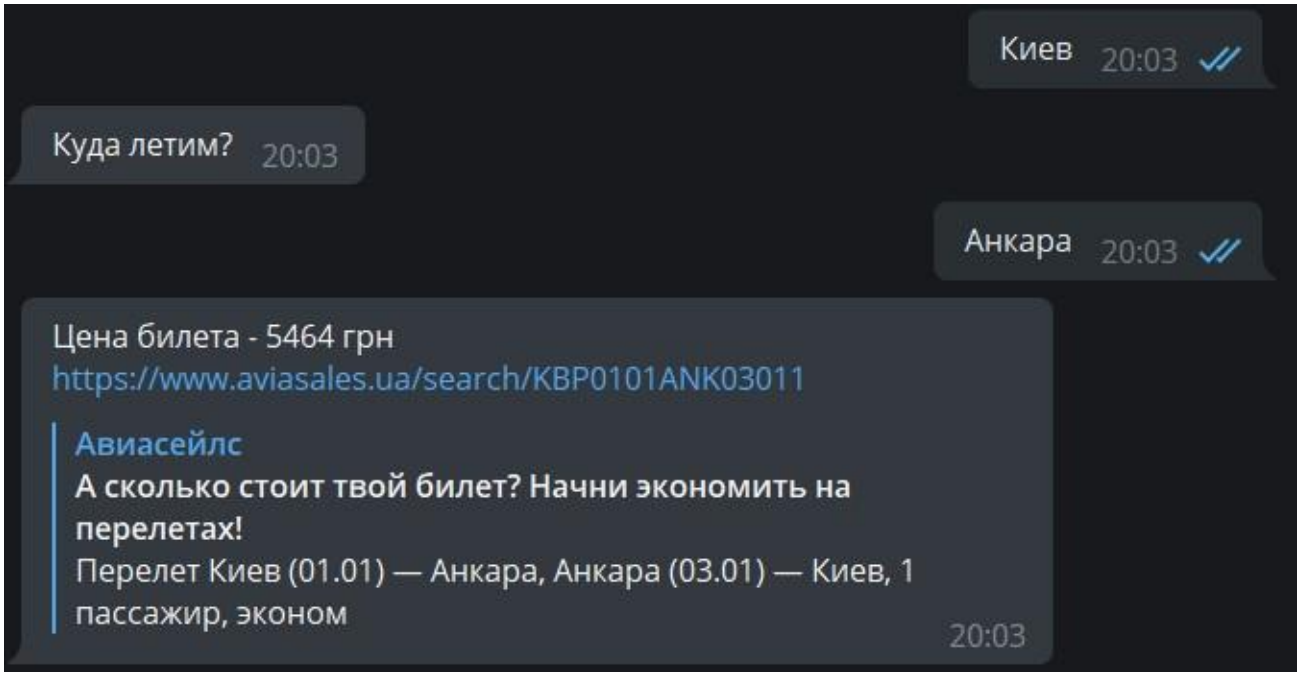


Рисунок 3.14 - Результат роботи чат-боту.

ВИСНОВКИ

В даний час популярність месенджерів як засобів спілкування незмінно зростає. Компанії, сім'ї, друзі щодня користуються можливостями обміну повідомленнями та медіаконтентом на відстані. Також варто відзначити зростання популярності такого виду програмних продуктів як чат-боти, які працюють на платформах мессенджерів.

Цілодобова служба підтримки користувачів, конвертація документів та медіафайлів, замовлення таксі, пошук необхідних даних та багато іншого інше в даний час може бути реалізовано в рамках лише одного мессенджера. Користувачі не доводиться скачувати безліч додатків для рішень вузьконаправлених завдань, тому що досить мати лише месенджер і необхідний набір чат-ботів, які не займають місце в пам'яті смартфона.

В рамках кваліфікаційної роботи були виконані поставлені завдання. По-перше, були вивчені мессенджери. Було проведено порівняння та аналіз переваг та недоліків, після чого було вибрано мессенджер Telegram як найзручніший та доступний у плані документації.

По-друге, були вивчені існуючі аналоги чат-бота на платформі Telegram, а також виявлені їх переваги, недоліки та цікаві рішення.

На основі цього були виявлені вимоги для розробки власного чат-бота.

В рамках виконаної задачі були вибрані технології та середа для розробки чат-бота серед яких Java 8, Spring Data Jpa, Hibernate, PostgreSQL, Liquibase, Maven та Intellij IDEA.

Таким чином, результатом випускної кваліфікаційної роботи є повністю реалізований чат-бото для пошуку дешевих авіаквитків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Telegram [Електронний ресур]. – Режим доступу: https://uk.wikipedia.org/wiki/Telegram
2) Statista [Електронний ресур]. – Режим доступу: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/
3) Месенджери в Україні [Електронний ресурс]. – Режим доступу: https://comments.ua/article/it/technology/641679-messendzhery-v-ukraine-osnovnye-igroki-problemy-i-perspektivy.
4) Самый популярный мессенджер в Украине [Електронний ресурс]. – Режим доступу: https://itc.ua/news/inmind-samyj-populyarnyj-messendzher-v-ukraine-viber-im-polzuyutsya-99-oproshennyh-v-vozraste-25-34-let-dalee-idut-facebook-messenger-telegram-whatsapp-i-skype/\
5) Рейтинг web технологій для java [Електронний ресур]. - Режим доступу: https://trends.google.com/trends/explore?q=Servlet,%2Fm%2F026mhl,%2Fm%2F07sz_x,%2Fm%2F0dhx5b
6) Spring Framework документація [Електронний ресур]. - Режим доступу: https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/overview.html
7) TIOBE Index for November 2020 [Електронний ресур]. – Режим доступу: https://www.tiobe.com/tiobe-index/
8) PostgreSQL [Електронний ресур]. – Режим доступу: https://www.guru99.com/introduction-postgresql.html
9) Apache Maven [Електронний ресур]. – Режим доступу: https://dev.to/hamzajvm/apache-maven-an-overview-40j0
10) Intellij IDEA [Електронний ресур]. – Режим доступу: https://www.jetbrains.com/help/idea/discover-intellij-idea.html

11) Intellij IDEA introduction [Електронний ресур]. – Режим доступу: https://www.tutorialspoint.com/intellij_idea/intellij_idea_introduction.htm
12) Инструкция по созданию телеграм ботов на java [Електронний ресур]. – Режим доступу: https://habr.com/ru/post/476306/
13) Hibernate (библиотека) [Електронний ресур]. – Режим доступу: https://ru.wikipedia.org/wiki/Hibernate_(библиотека)
14) Hibernate [Електронний ресур]. – Режим https://hibernate.org
15) Liquibase Документація [Електронний ресур]. – Режим доступу: https://www.liquibase.org
16) Maven [Електронний ресур]. – Режим доступу: https://maven.apache.org/issue-management.html