ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра комп‘ютерної інженерії

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: **«РОЗРОБКА ТЕЛЕГРАМ-БОТУ МОВОЮ JAVA ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ»**

Виконав: студент 5 курсу, групи
ПП3–51 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
_____ Гончар В.В_____
(прізвище та ініціали)
Керівник _____Зінченко О.В._____
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормконтроль_____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Телекомунікацій

Кафедра Інженерії програмного забезпечення Ступінь вищої освіти - «Бакалавр» Напрям підготовки - 121 «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
О.В. Негоденко
«__»
_____2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Гончару Владиславу Віталійовичу

(прізвище, ім'я, по-батькові)

1. Тема роботи: «Розробка телеграм-боту мовою Java для вивчення англійської мови»
Керівник роботи д.т.н., доц., Зінченко О.В.,
Затверджені наказом вищого навчального закладу від «16» лютого 2022 року №22
1. Строк подання студентом роботи _____.
2. Вихідні дані до роботи:
3. Методи і методики вивчення англійської мови;
Науково-технічна література з питань пов'язаних з програмним забезпеченням щодо розробки чат-ботів;
Технічна література щодо гнучких та безпечних програмних архітектур;
Технічна література щодо реляційних баз дадих.
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Аналіз і виділення переваги розробленого ПЗ
4.2 Розробка ПЗ
4.3Тестування ПЗ
5. Перелік графічного матеріалу.
1. Титульний слайд
2. Мета, об'єкт та предмет дослідження
3. Актуальність роботи
4. Аналоги
5. Програмні засоби реалізації
6. Архітектура боту

7. Апробація результатів дослідження
8. Висновки

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№	Процедура	Термін виконання	
1	Підготовка Розділу 1	-	До 04.04
2	Підготовка Розділу 2	З 04.04	До 25.04
3	Підготовка Розділу 3	З 25.04	До 06.05
4	Висновки + Презентація	З 06.05	До 13.05
5	Перевірка роботи на плагіат + Передзахист	З 16.05	До 01.06
6	Захист роботи	З 02.06	До 21.06
7	Випуск	30.06	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 65 с., 3 табл., 33 рис., 13 джерел.

ТЕЛЕГРАМ БОТ, АНГЛІЙСЬКА МОВА, МЕТОДИКИ НАВЧАННЯ, TELEGRAM API, JAVA, MYSQL, DOMAIN DRIVEN DESIGN, ЕФЕКТИВНІСТЬ, ЗРУЧНІСТЬ.

Об’єкт дослідження – телеграм-бот для вивчення англійської мови.

Предмет дослідження – чат-боти, методи вивчення англійської мови.

Мета роботи – розробка телеграм-боту для вивчення англійської мови.

Методи дослідження – методи оптимального управління, обробляння, аналіз отриманих даних, порівняння аналогів, моделювання архітектури.

Визначено потреби користувачів вивчення англійської мови онлайн і в чат-ботах в тому числі.

На основі результатів виконаних досліджень зроблено телеграм-бот для вивчення англійської мови.

Упровадження розробленої системи дозволяє вивчати нові слова, не звертаючись до перекладачів та словників, так як бот має функцію показу медіа-матеріалів невідомих слів, а згодом можна подивитись статистику вивчених слів. Також бот має ігровий елемент, так як гра в рими, в якому користувачу потрібно придумати риму на слово, яке йому відправляє бот.

ЗМІСТ

ВСТУП

Вивчення англійської мови стало дуже актуальним з розвитком IT індустрії. Методи її вивчення переповнюють Інтернет ресурси, також разом з ним розвиваються і чат-боти.

Чат-бот – програма, яка створюється людиною для людей та заточена під певне коло цілей. Чат-бот імітує розмову з людиною в Інтернеті, саме тому даний сервіс найкраще зарекомендував себе саме в месенджерах.

Чат-боти використовують багато компаній, так як надають багато переваг:

- зручність (чат-бот завжди знаходиться під рукою, у вашому улюбленому месенджері);
- легкість (не потрібно скачувати окремий додаток для виконання базових цілей);
- безкоштовність (обумовлено тим що месенджери, як правило безкоштовні)

Компанії надають базовий функціонал, такий як: привітання клієнта, відправлення йому необхідної інформації у вигляді тексту, зображення тощо, планування повідомлень, встановлення автоматичної відповіді на ключові слова (текст, зображення, меню), автоматична трансляція у ваш RSS-канал та багато іншого.

Багато спеціалістів стверджують, що чат-боти краще використовувати на інформаційних ресурсах. Компаніям інших напрямів усе ж варто залишатися «живими», тобто використовувати для технічної підтримки працю людей. На це є, як мінімум, 3 причини:

- ви відповідаєте на повідомлення “живою” мовою, а не як робот;
- ви слідкуєте за оновленнями, а це значить, що ви відчуваєте свою аудиторію;

Цільовою аудиторією телеграм-ботів для вивчення іноземних може вважатися любий користувач, який зацікавлений вивченням певної мови і хоче регулярно тренуватися і покращувати свої навичк

1. АНАЛІЗ І ВИДІЛЕННЯ ПЕРЕВАГИ РОЗРОБЛЕНОГО ПЗ

1.1 Переваги вивчення англійської мови в Інтернеті над класичним методом

З появою Всесвітньої мережі Інтернет, наше життя ніколи не буде таким яким воно було до нього. Можливості Інтернету буквально безмежні. Його величина зростає кожний день, постійно з'являються нові сайти, локальні мережі і також величезна кількість інформації. На сьогоднішній день.

В даний час інтернет використовують більше 2,5 млрд осіб і ця кількість постійно зростає. До всесвітньої мережі вже підключено більшість розвинених держав, а можливості супутникового зв’язку показують, що у найближчому майбутньому інтернет буде проведений навіть у важкодоступні регіони. У світі, де глобалізація виходить на перші позиції в економічних, соціальних і політичних відносинах, неможливо уявити собі життя без цього універсального засобу зв’язку.

Очевидно, що, поява Інтернету не могла не задіти освітню систему і зокрема вивчення інших мов.

Найпопулярніші способи вивчення англійської сьогодні - це:

- робота в групах (відвідування курсів);
- індивідуальні зайняття з репетитором;
- самостійне вивчення мови.

Методика вивчення англійської мови - це система принципів, за якими будується план вивчення іноземної мови. Іншими словами - це сукупність методів, педагогічних прийомів, форм занять і завдань, які позитивно себе зарекомендували під час практики.

Метод є шляхом реалізації методики. Наприклад, обираючи певну

методику навчання, ви все одно будете використовувати певний метод - самостійний метод навчання, або метод навчання в групах.

Тобто, ви можете самостійно навчатися за певною методикою, і з групою, і з репетитором.

Методика:

- має головний принцип. Наприклад: «Вивчити англійську мову за 2 місяці»;
- має прописані способи реалізації. Наприклад: «Займайтеся по 20 хвилин в день»;
- передбачає ефективні комбіновані форми завдань, які спрямовані на одну мету. Наприклад, швидко вивчити англійську;
- обіцяє результат через певний проміжок часу, якщо учень буде виконувати всі завдання і рекомендації.

Метод:

- описує формат ваших занять (самостійний, з групою, з репетитором, в інтернеті);
- має на увазі, що учень або вчитель самостійно підберуть завдання для своїх уроків (немає ніяких критеріїв щодо того, якими будуть ці завдання);
- не структурує використання тих чи інших прийомів навчання;
- не відповідає за швидкість сприйняття інформації і не може гарантувати результатів сам по собі.

1.2 Огляд методик вивчення англійської мови

А тепер поговоримо про самі популярні методики, які добре показують себе під час вивчення англійської мови.

Методика Шехтера – добре підходить для тих, хто вивчає англійську мову для подорожей або неформальних поїздок, декілька його ознак:

- група повинна бути з однаковим рівнем знань;

- граматика вивчається в останню чергу;
- неможливо реалізувати при самостійному навчанні.

Методика Берліца – її основний принцип занурити учня в незнайоме для нього середовище, сьогодні така методика використовується при навчанні по обміну або роботи за кордоном, особливості:

- засвоєння іншої мови без допомоги;
- знаходження учня в природній бесіді з носієм мови;
- можна повторити з репетитором;
- неможливо реалізувати при самостійному навчанні.

Методика Іллі Франка – реалізується шляхом читання якомога більше іншомовних текстів, особливості:

- необхідність повторювати вивчені слова;
- наявність вже готових текстів та мобільних додатків, які оформленні по цій методиці;
- можна реалізувати при самостійному навчанні.

Самостійне навчання завжди було складніше, так як через недостатню мотивацію, люди просто припиняють вивчати іншу мову. В таблиці 1.1, наведено переваги та недоліки самостійного навчання англійської мови.

Переваги	**Недоліки**
Безкоштовність (Інтернет має всю необхідну базу знань)	Досягнення перших результатів потребує часу та мотивації
Немає прив'язки до графіку	Потребує самодисциплінованості
Можна поспілкуватися з вчителем	Немає можливості поспілкуватися з учителем чи іншими

	учнями
Вивчення англійської мови стає стилем життя	Складно практикувати розмовну мову

Таб. 1.1. – Переваги та недоліки самостійного навчання

В навчанні англійської мови в групах також є свої переваги, особливо навчання такого роду дуже сильно допомагає розвинути розмовні навички і це приближує учня до повноцінної розмови з носіями. Переваги і недоліки навчання в групах наведено в таблиці 1.2.

Переваги	**Недоліки**
Розумна ціна (заняття в групах зазвичай коштують дешевше, ніж індивідуальні	Частота занять в групах відбуваються по 1-2 рази на тиждень (щоб досягти результатів швидше треба додатково вчитися вдома)
Дозволяє говорити з іншими студентами, заводити нові знайомства, викладач може допомогти зі складними граматичними темами	Під час уроку, одному учню приділяють небагато індивідуального часу, як можна було б отримати при індивідуальному занятті
Учню надають всі необхідну літературу, в якій знання вже структуровані	Доводиться вивчати теми, які можуть бути не цікаві учню

Таб 1.2. – Переваги та недоліки навчання в групах

Зараз в епоху процвітання месенджерів та чат-ботів, дуже

актуальною стає тема вивчення англійської мови безпосередньо через месенджери. В цьому випадку потенційному учню не потрібно завантажувати окремий додаток і може навчатися в любий момент в улюбленому месенджері.

1.3 Порівняння аналогів учбових Telegram-ботів

В месенджері Telegram аналогів розроблюваного боту небагато, але можна виділити декілька програмних продуктів.

Наприклад, AndyRobot - даний бот має доволі невеликий функціонал, який може тільки допомогти у вивченні нових слів, але робить це у своєму цікавому стилі. В процесі вивчення слів, дуже часто зустрічаються такі, яких ви не знаєте і бот вам допомагає зрозуміти, що воно означає за допомогою медіа та аудіо-матеріалів, щоб зрозуміти вимову невідомого слова, приклад як виглядає вивчення слів за допомогою цього бота наведено на рисунку 1.3.

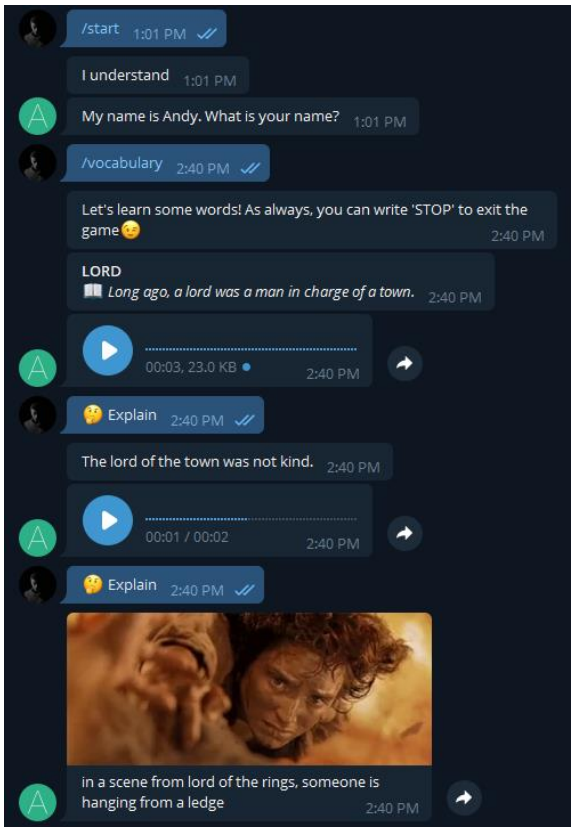


Рис 1.3 – Бот AndyRobot

Переваги:

- надає аудіо та медіа-матеріали для кращого розуміння слова;
- має заняття за допомогою емодзі;
- має функцію нагадування про себе;

Недоліки:

- має багато службових слів для виклику функціоналу бота, який засмічує історію розмови з ботом;
- не має нормальної ігрової функції вивчення мови;
- більшість слів призначенні для користувачем з невеликим рівнем знань англійської мови, користувачам зі середніми знаннями буде не цікаво займатися;
- має багато команд-заглушок, які по значенню повинні розпочинати якусь активність, а по факту являється рекламою свого додатку;

Наступний наш бот-аналог, це EnglishChatBot. Бот виконує функцію чат-кафе, де вам шукають співрозмовника, щоб практикувати розмовні навички. Сам по собі бот не дає ніякого навчального матеріалу або навчальних функцій, а лише шукає вам співрозмовника.

Переваги:

- дає можливість поспілкуватися з іншою людиною, яка також зацікавлена в вивченні англійської мови;
- простий інтерфейс;

Недоліки:

- бот не має функціоналу для допомоги вивчення англійської мови, а являється підбирачем співрозмовника;
- низька популярність бота, пошук нових людей для розмови може зайняти дуже багато часу.

І останній достойний бот під назвою WordBot. Бот має функціонал для покращення словникового запасу, вивчення слів антонімів, синонімів і також дає змогу дізнатись значення різних слів.

Переваги:

- дає можливість дізнатися слова синоніми, антоніми, які можуть цікавити користувача;
- має функцію «слово дня».

Недоліки:

- на дає можливості потренувати граматику;
- не має ігрових команд для більш ефективного та не нудного вивчення слів.

2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Завдання Telegram-боту для вивчення англійської мови

Вивчення англійської мови являється дуже довгим, вимагаючим уваги та наснаги процесом. Багато учнів, які намагаються вивчити англійську мову стикаються з недостатнім рівнем концентрації, або нудьгою під час занять. Все це може бути обумовлено тим, що:

- викладач не має достатньої кваліфікації;
- вивчення граматики переважає над розмовними заняттями.

Ці фактори являються причиною відмови від бажання вивчити англійську мову. Одним з самих дієвих способів являється вивчення англійської мови онлайн за допомогою занять в групах, але це веде за собою багато обмежень, наприклад:

- прив’язка до графіку;
- курси коштують значних грошей.

Саме в випадку коли в потенційного учня немає часу або коштів на курси в групах, на допомогу приходять боти в популярних месенджерах. Основні переваги чат-ботів для вивчення англійської мови в тому, що:

- немає потреби нагляду вчителя;
- завжди «під рукою»;
- безкоштовно;
- наявність медіа-матеріалів для будь-якого слова.

Проектований бот дозволить виконувати такі задачі, пов’язані з навчанням англійської мови:

- допомога вивчення нових слів;
- спілкування з ботом без зайвого засмічення історії чату;
- перегляд статистики учня;
- спеціальна команда, яка пояснить як працювати з ботом;

- дозволити учню проявити свою креативність в грі в рими;

Мета роботи – розробка Telegram-боту для допомоги у вивченні англійської мови.

2.2 Огляд месенджеру Telegram та Telegram API

Для чого потрібен Telegram:

- відправка повідомлень;
- прикріплення файлів до повідомлень (картинки, музика) об’ємом до 1.5 ГБ кожен;
- відправка емодзі, стікерів, збережених відео та анімацій;
- створення конференцій;
- наявність каналів та ботів.

Також месенджер має мобільний додаток, онлайн версію та повноцінну ПК версію.

Найбільше нас цікавить Telegram API, за допомогою якого можна створювати чат-ботів.

Чат-бот – програма, яка створюється людиною для людей та заточена під певне коло цілей. Чат-бот імітує розмову з людиною в Інтернеті, саме тому даний сервіс найкраще зарекомендував себе саме в месенджерах.

Чат-боти використовують багато компаній, так як надають багато переваг:

- зручність (чат-бот завжди знаходиться під рукою, у вашому улюбленому месенджері);
- легкість (не потрібно скачувати окремий додаток для виконання базових цілей);
- безкоштовність (обумовлено тим що месенджери, як правило безкоштовні)

Компанії надають базовий функціонал, такий як: привітання клієнта, відправлення йому необхідної інформації у вигляді тексту, зображення тощо, планування повідомлень, встановлення автоматичної відповіді на ключові слова (текст, зображення, меню), автоматична трансляція у ваш RSS-канал та багато іншого.

Багато спеціалістів стверджують, що чат-боти краще використовувати на інформаційних ресурсах. Компаніям інших напрямів усе ж варто залишатися «живими», тобто використовувати для технічної підтримки працю людей. На це є, як мінімум, 3 причини:

- ви відповідаєте на повідомлення “живою” мовою, а не як робот;
- ви слідкуєте за оновленнями, а це значить, що ви відчуваєте свою аудиторію;
- ви можете вчасно реагувати на неправильні коментарі.

API дозволяє підключати ботів до системи Telegram. Боти Telegram – це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Ці облікові записи служать інтерфейсом для коду, який виконується десь на Telegram сервері.

Щоб скористатися цим, не потрібно нічого знати про те, як працює протокол шифрування MTProto — клієнт-серверний набір протоколів, який служать для доступу до сервера з клієнтського додатку через незахищене з'єднання, схему роботи можна на рисунку 1.3. Цей набір можна розділити на 3 частини:

- API високого рівня (High-level API) та мова типів – відповідає, як за запити API і відповіді, так і за серійність даних;
- криптографічні компоненти та компоненти авторизації – визначає, як додаток (клієнт) авторизується на сервері, а також шифрування повідомляє перед відправкою на транспортний рівень;

- транспортний компонент – визначає, як клієнт і сервер обмінюються повідомленнями за допомогою таких транспортних протоколів, як UDP, TCP, HTTP(S), WebSocket та інші.

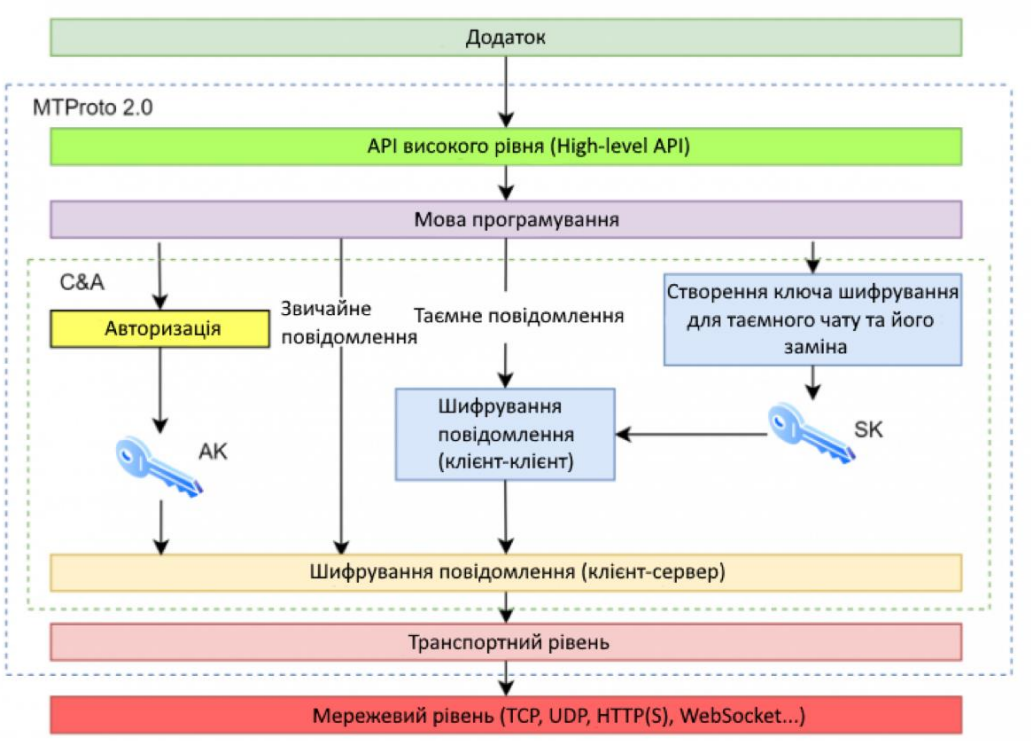


Рис. 2.1. – Схема роботи MTProto 2.0

Рівень який зв'язаний з криптографією і авторизацією (C&A), можна поділити на 3 модулі:

- Авторизація – цей модуль відповідає за оригінальну авторизацію клієнта. Він працює під час першого запуску програми, щоб отримати ключ авторизації.
- Створення ключа шифрування для таємного чату та його заміна – цей модуль відповідає за встановлення сесійного загального секретного ключа між клієнтами, що використовується у наскрізному шифруванні.
- Шифрування повідомлення – модуль, не покладаючи рук, шифрує наші повідомлення симетричним шифром AES.

АК (Ключ авторизації) - 2048-бітний ключ, який створюється на етапі реєстрації користувача за допомогою алгоритму Diffie-Hellman, а доступ до нього має тільки клієнт і сервер. Більше того, стверджується, що ключ шифрується на сервері, а ключ до ключа відправляється на інший сервер. Таким чином, отримання доступу лише до одного серверу недостатньо для злому.

2.3 Архітектура протоколу HTTP та HTTPS

Це дозволяє спілкуватися сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram

HTTP (HyperText Transfer Protocol) — це протокол передачі даних у Інтернеті. ого використовують для отримання інформації з веб-сайтів. Протокол HTTP заснований на використанні технології «клієнт-сервер»: клієнт, що відправляє запит, є ініціатором з’єднання, а сервер, який отримує запит, виконує його і відправляє клієнту результат, схема роботи зображена на рисунку 1.4.

HTTPS (HyperText Transfer Protocol Secure) — це розширення протоколу HTTP, що підтримує шифрування за допомогою криптографічних протоколів SSL и TLS.

Далі проговоримо як це працює на практиці. Протокол не зберігає інформацію про користувачів. Перевага цього рішення, безумовно, полягає в меншому завантаженні даних на сервер, що призводить до набагато більш високої продуктивності веб-сайту.

Відмінність між HTTP і HTTPS - безпека. Безумовно варто вибрати останній варіант. Захист даних важливий не тільки там, де проводяться платежі. Цей тип шифрування також все частіше вибирають власники веб-сайтів, на яких користувачі створюють аккаунти, надаючи свої персональні дані.

Однак це також пов'язано з одним особливим недоліком - щоб

включити веб-сайт кілька разів, його необхідно знову завантажити з сервера. Є рішення цієї проблеми - це файли cookie. Вони додаються на сайт і постійно збирають дані про користувачів мережі. Зібрана інформація може використовуватися власником веб-сайту для багатьох маркетингових кампаній або під час позиціонування. За допомогою файлів cookie він отримує інформацію про інтереси та останніх відвіданих сторінках відвідувачів свого веб-сайту. Завдяки цьому він може рекламувати свої нові товари або спонукати їх купувати конкретний товар, якщо замовлення не був оформлений раніше. Це звичайна тактика, використовувана в інтернет-маркетингу.

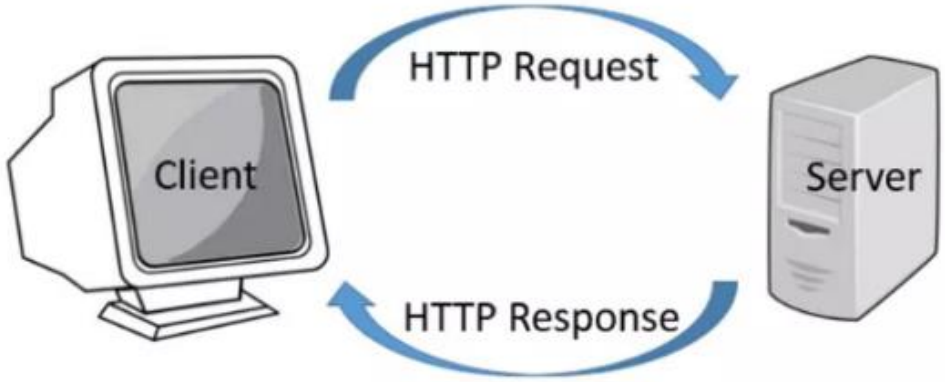


Рис.2.2 – Архітектура роботи протоколу HTTP

2.4 Мова програмування Java та її сучасні фреймворки

Java — це багатоплатформна, об'єктно-орієнтована мова, одна з найбільш використовуваних мов програмування. Java також використовується як обчислювальна платформа.

Вважається однією з швидких, безпечних і надійних мов програмування, якій надає перевагу більшість організацій для створення своїх проектів.

Деякі важлива функції Java:

- одна з простих у використанні мов програмування для вивчення;
- можна написати код один раз і запустити його практично на будь-

якій комп’ютерній платформі;

- не залежить від платформи. Програми, розроблені на одній машині, можна виконувати на іншій машині;
- призначений для створення об'єктно-орієнтованих додатків;
- це багатопотокова мова з автоматичним керуванням пам'яттю;

Java складається з трьох компонентів:

- JDK
- JVM
- JRE

JDK - містить інструменти, необхідні для написання програм на Java, і JRE для їх виконання, також має компілятор та засіб запуску програм Java. Компілятор перетворює код, написаний на Java, у байтовий код. Панель запуску програм Java відкриває JRE, завантажує необхідний клас і виконує його основний метод.

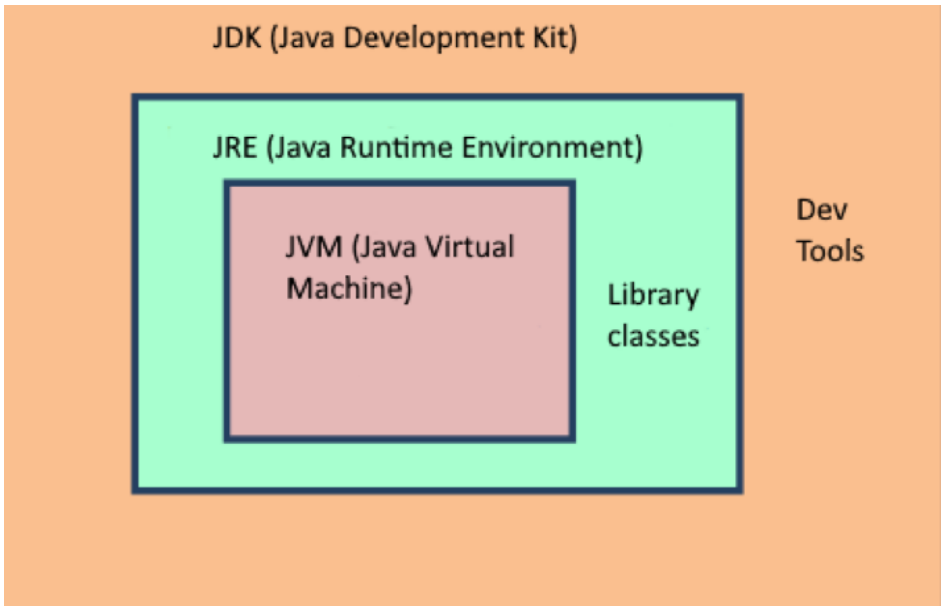


Рис 2.3 – Схема JDK

JVM - це віртуальна машина, яка дозволяє комп’ютеру запускати програми Java, а також програми, написані іншими мовами, які також скомпільовані в байт-код Java. JVM детально описується специфікацією,

яка формально описує те, що потрібно для реалізації. Наявність специфікації гарантує сумісність програм Java в різних реалізаціях, тому авторам програм, які використовують JDK, не потрібно турбуватися про особливості базової апаратної платформи.

Довідкова реалізація JVM розроблена проектом OpenJDK як відкритий вихідний код і включає JIT-компілятор під назвою HotSpot. Комерційно підтримувані випуски Java, доступні від Oracle Corporation, засновані на середовищі виконання OpenJDK. Eclipse OpenJ9 — ще одна JVM з відкритим кодом для OpenJDK.

Також JVM має свою модель пам'яті, кожен потік, що працює у віртуальній машині Java, має свій власний стек потоків. Стек потоків містить інформацію про те, які методи викликав потік, щоб досягти поточної точки виконання. Коли потік виконує свій код, стек викликів змінюється (рисунок 1.6).

Java Heap (куча) - динамічна розширювана область пам'яті, створювана при старті JVM. Використовується Java Runtime для виділення пам’яті під об’єкти та класи JRE. Створення нового об'єкта також відбувається в кучі. Тут працює збірник сміття (Garbage Collector, GC): звільняє пам'ять шляхом видалення об'єктів, на яких немає будь-яких посилань. Любий об’єкт, створений у кучі, має глобальний доступ і на нього може посилатися з будь-якої частини додатку.

Стекова пам'ять в Java працює за схемою LIFO (Останній-зайшов-Перший-вийшов). Всякий раз, коли викликається метод, у стека пам'яті створюється новий блок, який містить примітиви та посилання на інші об'єкти в методі розташування в RAM та досягнення процесора через стека вказівника.

Як тільки метод закінчує роботу, блок також перестає використовуватися, тим самим надається доступ для наступного методу. Розмір стекової пам'яті на багато менше обсягу пам'яті в кучі.

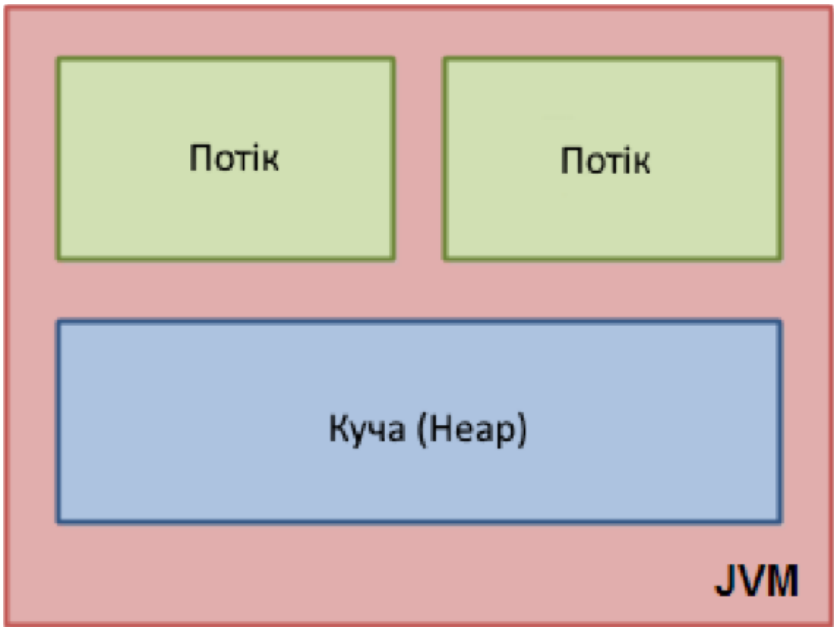


Рис 2.4. – Схема з точки зору логіки

Стек потоків також містить усі локальні змінні для кожного методу, що виконується (усі методи в стеку викликів). Потік може отримати доступ лише до власного стеку потоків. Локальні змінні, створені потоком, невидимі для всіх інших потоків, крім потоку, який його створив. Навіть якщо два потоки виконують однаковий код, ці два потоки все одно створюватимуть локальні змінні цього коду в кожному своєму стеку потоків. Таким чином, кожен потік має свою версію кожної локальної змінної.

Усі локальні змінні примітивних типів (boolean, byte, short, char, int, long, float, double) повністю зберігаються в стеку потоків і, таким чином, не видимі для інших потоків. Один потік може передати копію примітивної змінної в інший потік, але він не може ділитися самою примітивною локальною змінною.

Куча містить усі об’єкти, створені у вашій програмі Java, незалежно від того, який потік створив об’єкт. Це включає версії об’єктів примітивних типів (наприклад, Byte, Integer, Long тощо). Не має значення, чи був об’єкт створений і призначений локальній змінній, чи створений

як змінна-член іншого об’єкта, об’єкт все одно зберігається в купі.

Діаграма, що ілюструє стек викликів і локальні змінні, що зберігаються в стеках потоків, і об'єкти, що зберігаються в купі (рисунок 1.7):

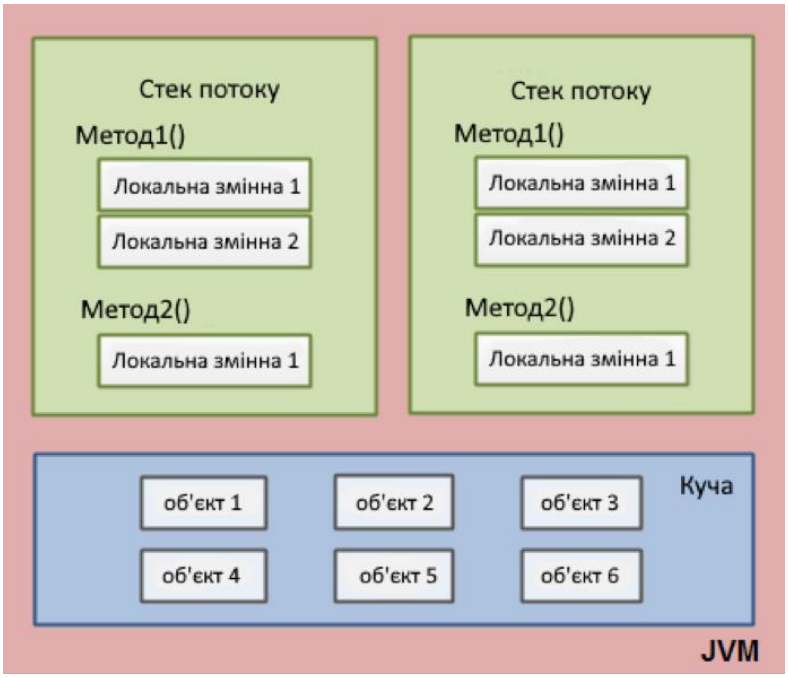


Рис 2.5. – Діаграма стеку викликів і локальних змінних

Локальна змінна може бути примітивного типу, і в цьому випадку вона повністю зберігається в стеку потоків. Вона також може бути посиланням на об'єкт. У цьому випадку посилання (локальна змінна) зберігається в стеку потоків, а сам об’єкт зберігається в купі.

Об’єкт може містити методи, а ці методи можуть містити локальні змінні. Ці локальні змінні також зберігаються в стеку потоків, навіть якщо об’єкт, якому належить метод, зберігається в купі.

Змінні об'єкта зберігаються в купі разом із самим об'єктом. Це працює як тоді, коли змінна має примітивний тип, так і якщо вона є посиланням на об’єкт. Статичні змінні класу також зберігаються в купі разом з визначенням класу.

До об’єктів у купі можуть отримати доступ усі потоки, які мають посилання на об’єкт. Коли потік має доступ до об’єкта, він також може отримати доступ до змінних цього об’єкта. Якщо два потоки викликають

метод для одного об’єкта одночасно, вони обидва матимуть доступ до змінних об’єкта, але кожен потік матиме власну копію локальних змінних, діаграма, що ілюструє пункти вище (рисунок 1.8).

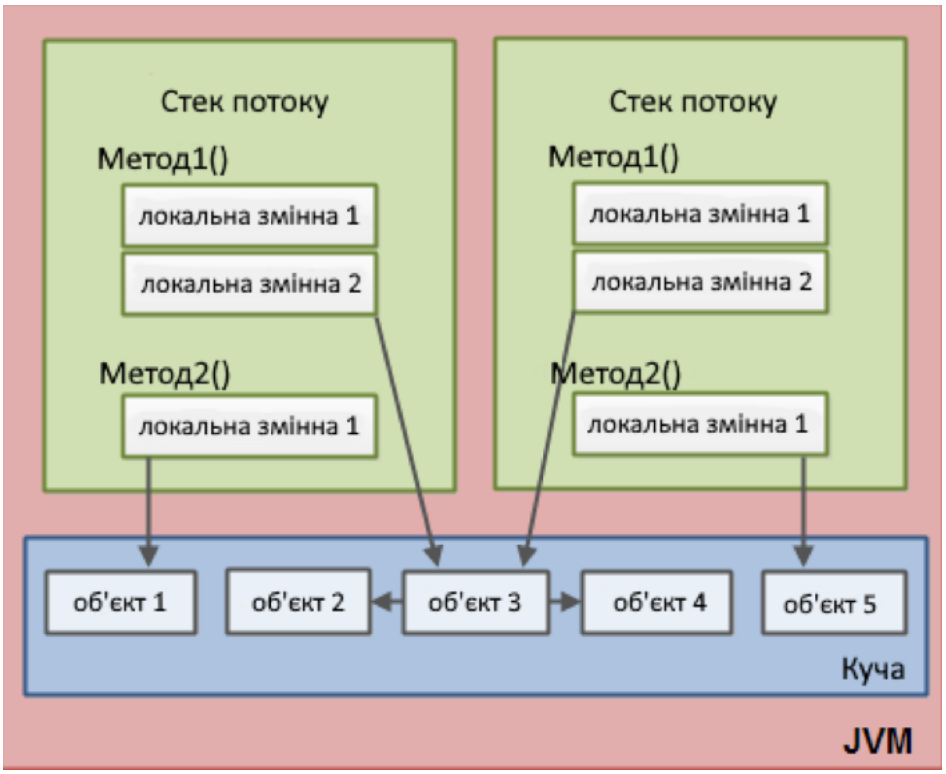


Рис 2.6. – Діаграма стеку викликів з локальними змінними та посиланнями на об’єкти

JRE - частина програмного забезпечення, призначена для запуску іншого програмного забезпечення. Містить бібліотеки класів, клас завантажувача та JVM. Простіше кажучи, якщо ви хочете запустити програму на Java, вам потрібен JRE. Якщо ви не програміст, вам не потрібно встановлювати JDK, а лише JRE для запуску програм Java.

Чому саме JRE:

- JRE містить бібліотеки класів, JVM та інші допоміжні файли. Він не містить жодних інструментів для розробки Java, таких як налагоджувач, компілятор тощо.
- він використовує важливі класи пакетів, такі як math, swing, util, lang, awt і бібліотеки часу виконання.
- якщо потрібно запускати Java-аплети, то JRE має бути

встановлений у системі.

Spring Framework - одна з найпотужніших і широко використовуваних платформ Java.

Завдяки своїй концепції впровадження залежностей та особливостям аспектно-орієнтованого програмування Spring підкорила світ розробки. Це платформа з відкритим вихідним кодом, яка використовується для корпоративних програм. Фреймворк зазвичай використовується для Enterprise Java. Середовище надзвичайно універсальне і може використовуватися для створення більшості типів програм Java.

З допомогою Spring розробники можуть створювати слабозв'язані модулі, у яких залежності обробляються фреймворком, а чи не залежать від бібліотек у коді. Spring найчастіше використовують у розробці веб-застосунків. Його застосовує низка технологічних гігантів, включаючи Netflix та Amazon. Головна перевага фреймворку полягає в тому, що він надзвичайно легкий і не вимагає виклику веб-сервера. З точки зору ефективності це одна з найпопулярніших платформ Java. Spring надзвичайно простий і, як правило, може використовуватися навіть новачками. Він також забезпечує зворотну сумісність та можливість тестування. Фреймворк Spring є вичерпним і охоплює безліч функцій, включаючи безпеку та налаштування підключення до БД, які легко вивчити. Крім того, оскільки це найпопулярніший веб-фреймворк, ви можете знайти безліч документації та активну спільноту.

Даний фреймворк активно використовує впровадження залежностей (Dependency Injection) — у цьому принципі замість послідовно брати управління потоком програма передає управління зовнішньому контролеру, який управляє потоком. Зовнішній контролер – це події. Коли відбувається якась подія, потік програми продовжується. Це надає гнучкість додатку. У Spring IoC виконується за допомогою DI,

які бувають трьох типів - використання установника, використання способу і використання конструктора, схема на рисунку 1.9.

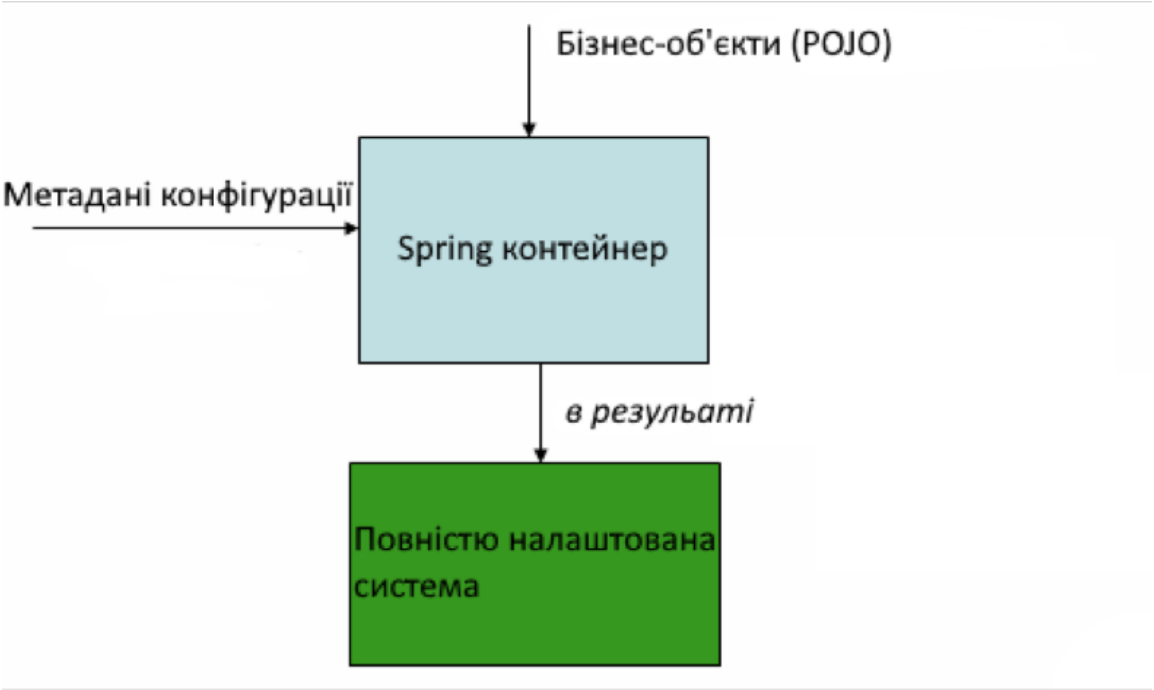


Рис 2.7. – Налаштування Spring контейнеру

У Spring об'єкти називаються bean-компонентами. Існує BeanFactory, яка управляє та налаштовує ці bean-компоненти. Ви можете думати про BeanFactory як контейнер, який створює екземпляри.

Більшість програм використовують анотації для конфігурації. ApplicationContext, який є надмножиною BeanFactory, використовується для більш складних додатків, які потребують поширення подій, декларативні механізми та інтеграція з аспектно-орієнтованими функціями Spring.

Сам Spring Framework, складається з багатьох модулів, які можна підключити до проекту, частина з них на рисунку 1.9.

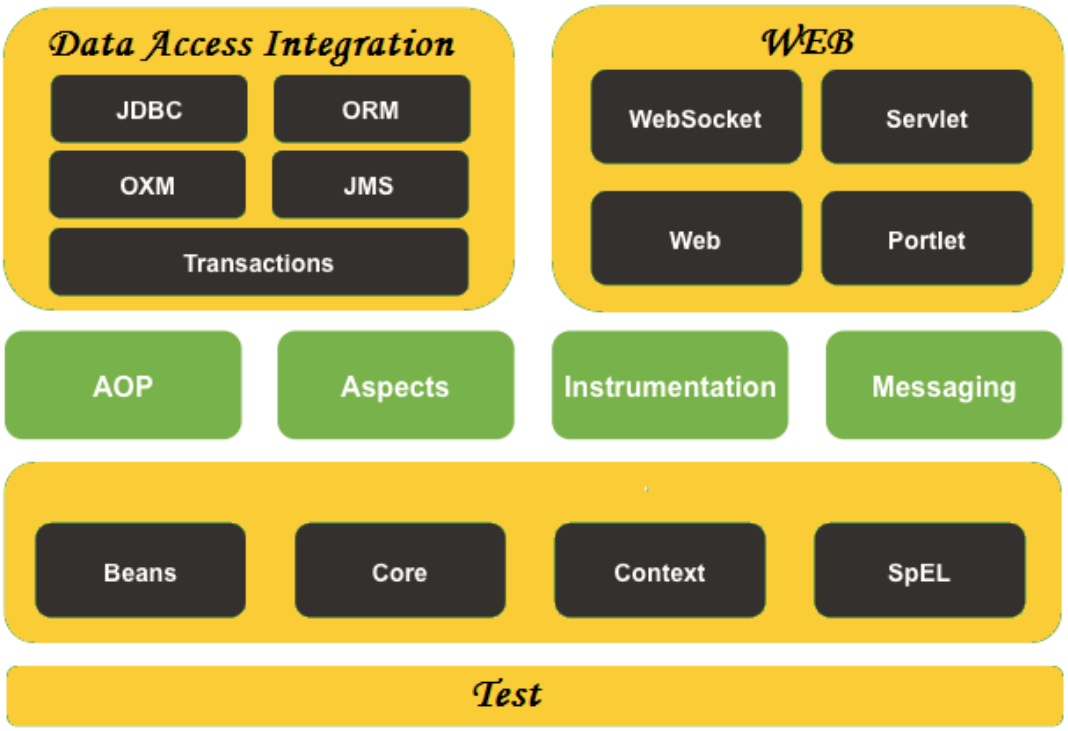


Рис 2.8. – Модулі Spring Framework

Spring Data можливість повністю видалити реалізації DAO. Інтерфейс DAO тепер є єдиним артефактом, який нам потрібно чітко визначити.

Щоб почати використовувати модель програмування Spring Data з JPA, інтерфейсу DAO необхідно розширити спеціальний інтерфейс JPA Repository, JpaRepository. Це дозволить Spring Data знайти цей інтерфейс і автоматично створити для нього реалізацію.

Розширюючи інтерфейс, ми отримуємо найбільш релевантні методи CRUD для стандартного доступу до даних, доступні в стандартному DAO.

Завдяки реалізації одного з інтерфейсів репозитарію DAO вже матиме деякі основні методи (і запити) CRUD, визначені та реалізовані.

Щоб зробити більш конкретні методи доступу, Spring JPA підтримує досить багато параметрів:

- просто задекларуйте новий метод в інтерфейсі;
- надайте фактичний запит JPQL за допомогою анотації @Query;
- є в наявності розширена специфікація та підтримка Querydsl;

- визначати користувацькі запити через іменовані запити JPA.

Третій варіант, підтримка специфікацій та Querydsl, схожий на критерії JPA, але використовує більш гнучкий і зручний API. Це робить всю операцію набагато більш читабельною та придатною для повторного використання. Переваги цього API стануть більш помітними при роботі з великою кількістю фіксованих запитів, оскільки ми потенційно могли б висловити їх більш стисло за допомогою меншої кількості повторно використовуваних блоків.

Spring Boot — це модуль фреймворку Spring на основі Java з відкритим кодом, який використовується для створення веб-додатків, також широко використовується у мікросервісній архітектурі. Він розроблений Pivotal Team і використовується для створення автономних додатків.

Мікросервіс - архітектура, яка дозволяє розробникам самостійно розробляти та розгортати додатки, які ізольовані один від одного. Кожен запущений мікросервіс має свій власний процес, і це досягає полегшеної моделі для підтримки бізнес-додатків. Мікросервісна архітектура дає наступні переваги:

- легке розгортання;
- проста масштабованість;
- сумісний з контейнерами;
- мінімальна конфігурація;
- менший час розробки.

Spring Boot надає хорошу платформу для розробників Java для розробки окремої та програми для розробки, яку можна просто запустити. Ви можете розпочати роботу з мінімальними конфігураціями без необхідності повного налаштування конфігурації Spring.

Spring MVC — використовується для створення веб-додатків. Він відповідає шаблону проектування Model-View-Controller. Він реалізує всі

основні функції основної пружинної структури, як-от інверсія керування, ін’єкція залежності.

Цей модуль Spring забезпечує елегантне рішення для використання за допомогою DispatcherServlet. Тут DispatcherServlet — це клас, який отримує вхідний запит і співставляє його з потрібним ресурсом, таким як контролери, моделі та представлення, детальніше на рисунку 1.11.

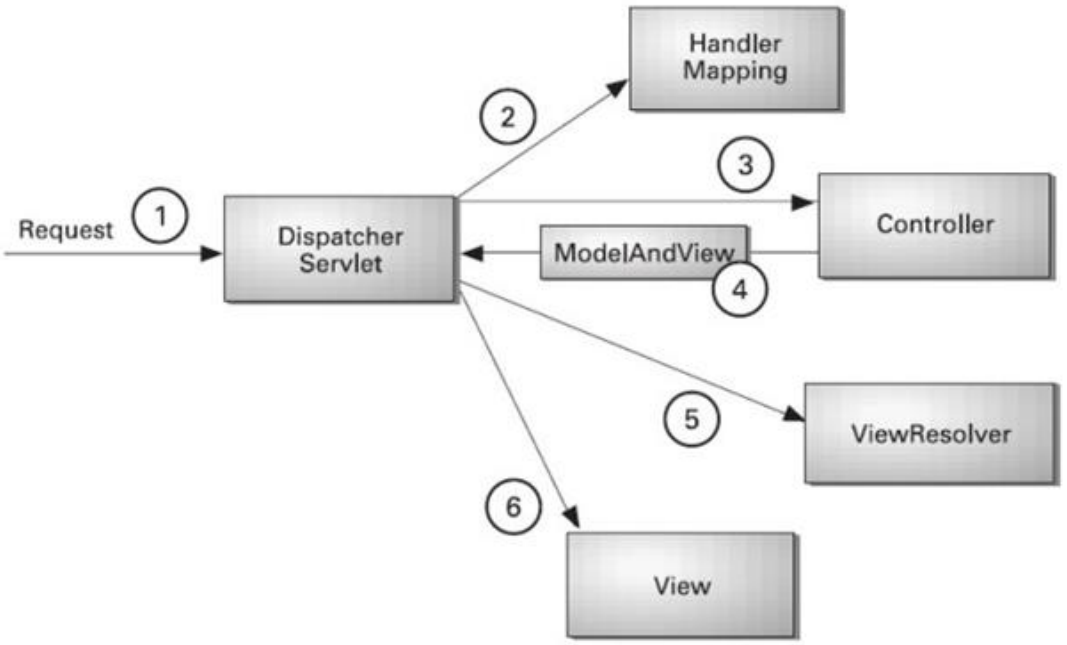


Рис 2.9. – Схема роботи Spring MVC

Переваги Spring MVC:

- розділення зон відповідальності –відокремлює кожну роль, де об’єкт моделі, контролер, об’єкт команди, роздільник перегляду, DispatcherServlet, валідатор тощо можуть виконуватися спеціалізованим об’єктом;
- легкий – він використовує легкий контейнер сервлетів для розробки та розгортання вашої програми;
- гнучка конфігурація – забезпечує надійну конфігурацію як для фреймворків, так і для класів додатків, що включає легке посилання в різних контекстах, наприклад від веб-контролерів до бізнес-об’єктів і валідаторів;

- швидка розробка - Spring MVC сприяє швидкому і паралельному розвитку додатку;
- багаторазовий бізнес-код – замість створення нових об’єктів він дозволяє нам перевикористовувати існуючі бізнес-об’єкти;
- легкий для тестування. У Spring зазвичай ми створюємо класи JavaBeans, які дозволяють вводити тестові дані за допомогою методів налаштування;
- гнучкий маппінг сторінок – надає конкретні анотації, які легко перенаправляють сторінку.

Spring AOP дозволяє аспектно-орієнтоване програмування в додатках Spring. У AOP аспекти дають змогу модулювати такі проблеми, як керування транзакціями, журналювання або безпека, які охоплюють декілька типів та об’єктів (часто називаються наскрізними проблемами).

AOP забезпечує спосіб динамічного додавання наскрізних проблем до, після або навколо фактичної логіки, використовуючи прості підключаються конфігурації. Це полегшує підтримку коду як у сьогоденні, так і в майбутньому. Ви можете додати/усунути проблеми без перекомпіляції повного вихідного коду, просто змінивши файли конфігурації. Також все можна конфігурувати за допомогою анотацій, але в цьому випадку ми втрачаємо гнучкість додатку.

AOP глибоко використовує поняття advice, joinpoint та pointcut, розглянемо, що це таке (наглядна схема на рисунку 1.12):

- advice - це дія, яку виконує аспект у певній точці з’єднання;
- joinpoint — це точка виконання програми, наприклад, виконання методу або обробка винятку. У Spring AOP joinpoint завжди представляє виконання методу.
- pointcut — це предикат або вираз, який відповідає точкам з’єднання (joinpoint).
- Advice пов'язаний з виразом pointcut і виконується в будь-якій

точці з'єднання, що відповідає pointcut.

- Spring за замовчуванням використовує мову виразів AspectJ pointcut.

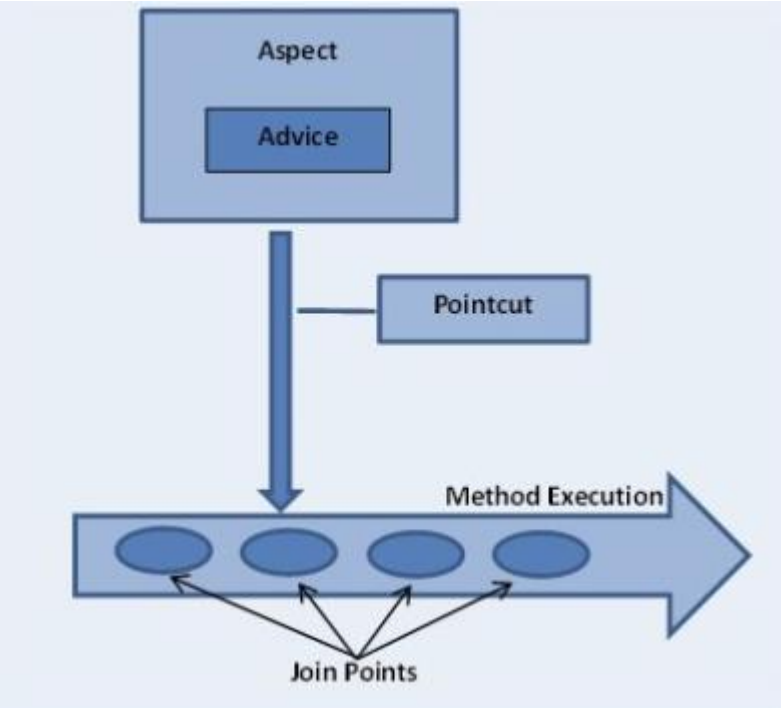


Рис. 2.10 – Схема роботи AOP

Існує п’ять видів advice в Spring AOP:

- before advice: advice, який виконується перед точкою з’єднання, але не має можливості запобігти потоку виконання, що переходить до точки приєднання (якщо це не викликає виняток).
- after returning advice: advice, який виконується після нормального завершення точки з’єднання: наприклад, якщо метод повертається без виключення.
- after throwing advice: advice, який буде виконаний, якщо метод завершує роботу, викликаючи виняток.
- after advice: advice повинна виконуватися незалежно від способу виходу точки з’єднання (звичайне чи виняткове повернення).
- around advice: advice, який оточує точку з’єднання, наприклад, виклик методу. Це найсильніший інструмент Spring AOP. Around

advice може виконувати користувацьку поведінку до і після виклику методу. Він також відповідає за вибір, чи перейти до точки з'єднання, чи скоротити виконання рекомендованого методу, повертаючи його власне значення, що повертається, або викликаючи виняток.

Spring AOP широко використовується в випадках, коли розробнику треба перехватити виконання коду в певній точці. Наприклад, за допомогою цього інструменту можна легко реалізувати логування всіх методів програми, код на рисунку 1.13.

```
@Aspect
@Component
@Slf4j
public class MethodLoggingAspect {

    @Around(value = "within(com.example.controller..*)")
    public Object methodLogging(ProceedingJoinPoint joinPoint) throws Throwable {
        double start = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        double executionTime = (System.currentTimeMillis() - start) / 1000;
        String baseUrl = ((RequestMapping) joinPoint.getSignature().getDeclaringType().getAnnotation(RequestMapping.class)).value()[0];
        log.info("{}/{} executed in {}s, passed with params: {}, and returned: {}",
                baseUrl,
                joinPoint.getSignature().getName(),
                executionTime,
                Arrays.toString(joinPoint.getArgs()),
                proceed.toString());
        return proceed;
    }

}
```

Рис 2.11 – Приклад реалізації логування за допомогою Spring AOP

Spring Security - Spring Security забезпечує аутентифікацію та авторизацію для нашої програми за допомогою простих фільтрів сервлетів. Веб-додатки чутливі до загроз та атак, оскільки вони доступні кожному, хто користується Інтернетом. Можливо, деякі кінцеві точки REST мають обмежений доступ для певних користувачів, наприклад, оновлення записів або операції, пов'язані з адміністратором.

Spring Security забезпечує структуру безпеки, яка захищає корпоративні програми на основі J2EE, надаючи потужні, настроювані функції, такі як аутентифікація та авторизація. Це де-факто стандарт захисту додатків на основі Spring.

Щоб впровадити максимально безпечний API, Spring Security має

влаштований шифратор паролів, або будь-якого тексту. Стандарті варіанти шифрування, які є в Spring Security:

- BCryptPasswordEncoder;
- Argon2PasswordEncoder;
- Pbkdf2PasswordEncoder;
- SCryptPasswordEncoder.

Також Spring Security забезпечує:

- інтеграція API сервлетів;
- розширювана підтримка як аутентифікації, так і авторизації;
- захист від атак, таких як фіксація сесії, підключення кліків;
- Spring MVC інтеграція;
- можливість захисту програми від bruteforce атак;
- портативність;
- захист від атак CSRF;
- підтримка конфігурації Java за допомогою анотацій.

Hibernate ORM - дозволяє розробникам легше писати програми, дані яких перебувають у процесі застосування. Фреймворк об’єктного/реляційного відображення (ORM), Hibernate займається збереженням даних, оскільки він застосовується до реляційних баз даних (через JDBC).

На додаток до власного «рідного» API, Hibernate також є реалізацією специфікації Java Persistence API (JPA). Таким чином, його можна легко використовувати в будь-якому середовищі, що підтримує JPA, включаючи програми Java SE, сервери додатків Java EE, контейнери Enterprise OSGi тощо.

Hibernate дає змогу розробляти персистентні класи відповідно до природних об’єктно-орієнтованих ідіом, включаючи успадкування,

поліморфізм, асоціацію, композицію та структуру колекцій Java. Hibernate не вимагає інтерфейсів або базових класів для постійних класів і дозволяє будь-якому класу або структурі даних бути постійними.

Також підтримує ліниву ініціалізацію, численні стратегії отримання та оптимістичне блокування з автоматичним встановленням версій і штампуванням часу,не вимагає спеціальних таблиць або полів бази даних і генерує більшу частину SQL під час ініціалізації системи, а не під час виконання.

Hibernate постійно пропонує вищу продуктивність порівняно з прямим кодом JDBC, як з точки зору продуктивності розробника, так і продуктивності під час виконання.

Даний фреймворк було розроблено для роботи в кластері серверів додатків і забезпечення високомасштабованої архітектури. Саме через це, він добре масштабується в будь-якому середовищі: використовується для керування внутрішньою мережею, яка обслуговує сотні користувачів, або для критично важливих програм, які обслуговують сотні тисяч клієнтів.

Переваги Hibernate:

- повністю відкритий;
- зменшує надмірність через JDBC API;
- підвищує продуктивність та зручність обслуговування;
- підтримує API-інтерфейси Persistence;
- забезпечує зв'язок між додатком та будь-якою базою даних.

2.5. Реляційна база даних MySQL

MySQL — це реляційна база даних з відкритим кодом. Одна з найбільш широко використовуваних баз даних SQL, MySQL надає пріоритет швидкості, надійності та зручності використання.

Зазвичай він відповідає стандарту ANSI SQL, хоча є кілька випадків, коли MySQL виконує операції інакше, ніж визнаний стандарт.

База даних написана на C і C++. Його синтаксичний аналізатор SQL написаний на yacc, але він використовує саморобний лексичний аналізатор.

MySQL був створений шведською компанією MySQL AB, заснованою шведами Девідом Аксмарком, Алланом Ларссоном і фінським шведом Майклом «Монті» Віденіусом. Початкова розробка MySQL Widenius і Axmark почалася в 1994 році.[22] Перша версія MySQL з'явилася 23 травня 1995 року. Спочатку вона була створена для особистого використання з mSQL на основі низькорівневої мови ISAM, яку розробники вважали занадто повільною та негнучкою. Вони створили новий інтерфейс SQL, зберігаючи той самий API, що й mSQL. Підтримуючи API узгодженим із системою mSQL, багато розробників змогли використовувати MySQL замість (за власною ліцензією) попередника mSQL.

2.6. Архітектура Telegram-боту

Почнемо з портів і адаптерів. Обидва вони чітко розділяють, який код є внутрішнім для програми, що є зовнішнім і що використовується для з’єднання внутрішнього та зовнішнього коду.

Крім того, ця архітектура чітко визначає три фундаментальні блоки коду в системі:

- робить можливим запуск користувальницького інтерфейсу, яким би типом інтерфейсу він не був;
- системна бізнес-логіка, або ядро програми, яке використовується користувацьким інтерфейсом для того, щоб справді відбувалися речі;
- код інфраструктури, який з’єднує ядро нашої програми з

такими інструментами, як база даних, пошукова система або сторонні API.

Далі йде ядро програми.

Ядро програми – це код, який дозволяє нашому коду робити те, що він повинен робити, це по суті і є наша програма. Він використовує кілька користувацьких інтерфейсів (CLI, API,…), але код, який фактично виконує роботу, той самий і розташований у ядрі програми, насправді не має значення, який інтерфейс користувача його запускає.

Типовий потік програми йде від коду в інтерфейсі користувача, через ядро програми до коду інфраструктури, назад до ядра програми і, нарешті, передає відповідь інтерфейсу користувача.

Інструменти - подалі від найважливішого коду в нашій системі, ми маємо інструменти, які використовує наша програма, наприклад, базу даних, пошукову систему, веб-сервер або консоль CLI, детально на рисунку 1.14.

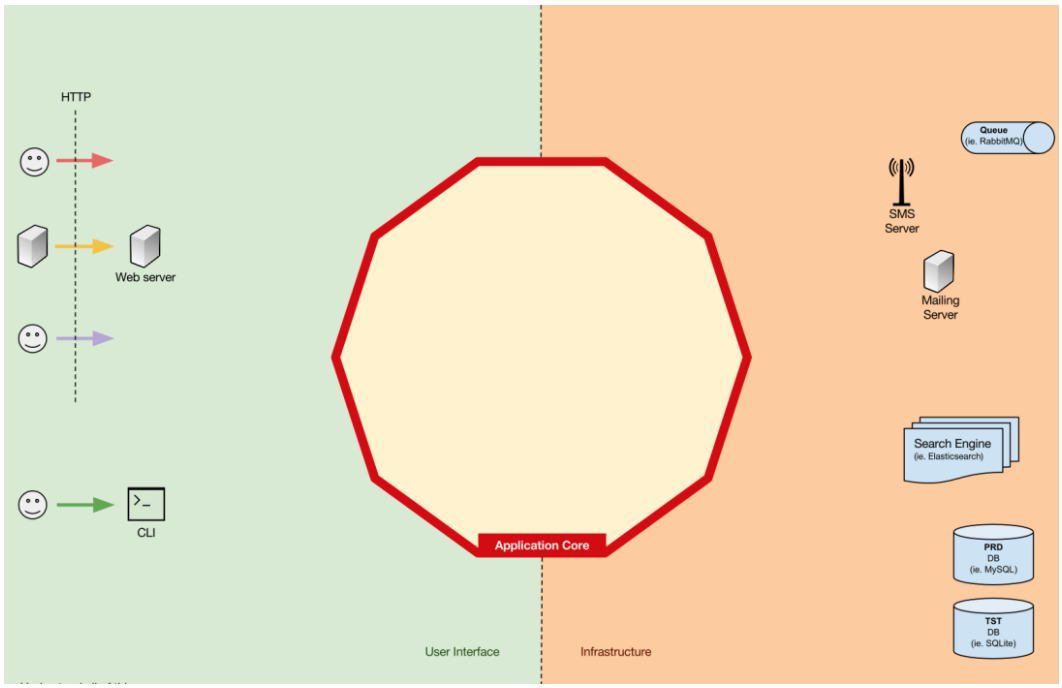


Рис. 2.12 – Ядро програми та інструменти, які воно використовує

Може здатися дивним помістити консоль CLI в те саме ж саме місце, що й механізм баз даних, і хоча вони мають різні типи цілей, насправді вони є інструментами, які використовує програма. Ключова відмінність полягає в тому, що, в той час як консоль CLI і веб-сервер використовуються для того, щоб сказати нашій програмі щось робити, наша програма повідомляє базі даних, щоб щось зробити. Це дуже актуальна відмінність, оскільки вона має серйозні наслідки для того, як ми пишемо код, який з'єднує ці інструменти з ядром програми.

Але, щоб не ламати всю архітектуру, потрібно підключити інструменти до ядра програми так, щоб ядро нічого не знало про конкретні реалізації інструментів з якими воно працює.

Блоки коду, які з'єднують інструменти з ядром програми, називають адаптерами (архітектура портів і адаптерів). Адаптери – це ті, які ефективно реалізують код, який дозволить бізнес-логіці взаємодіяти з певним інструментом і навпаки.

Адаптери, які наказують нашій програмі щось робити, називаються первинними або керованими адаптерами, а адаптери, яким наша програма каже щось робити, називаються вторинними або керованими адаптерами.

Порти – вони створені, щоб відповідати специфічній точці входу в ядро програми, порту. Порт, це не що інше, як специфікація того, як інструмент може використовувати ядро програми або як він використовується ядром програми. У більшості мов і в найпростішій формі ця специфікація, буде інтерфейсом, але насправді вона може складатися з кількох інтерфейсів і DTO (Data Transfer Object).

Важливо відзначити, що порти (інтерфейси) належать до бізнес-логіки, тоді як адаптери належать зовні. Щоб цей шаблон працював належним чином, надзвичайно важливо, щоб порти були створені відповідно до потреб ядра програми, а не просто мімікрували API інструменти.

Основні та вторинні адаптери - основні адаптери або адаптери

драйвера знаходяться навколо порту і використовують його, щоб повідомити ядро програми, що робити. Вони проводять все, що надходить від механізму замовлення, у виклик методу в ядрі програми.

Іншими словами, наші драйверні адаптери — це контролери або консольні команди, які впроваджуються в їхній конструктор із деяким об'єктом, клас якого реалізує інтерфейс (порт), який вимагає контролер або консольна команда.

У більш конкретному прикладі порт може бути інтерфейсом сервісу або інтерфейсом репозиторію, який вимагає контролер. Конкретна реалізація служби, репозиторію або запиту потім впроваджується та використовується в домені.

Крім того, порт може бути інтерфейсом командної шини або шини запитів. У цьому випадку конкретна реалізація команди або шини запитів вводиться в контролер, який потім створює команду або запит і передає їх відповідній шині (рисунок 1.15).

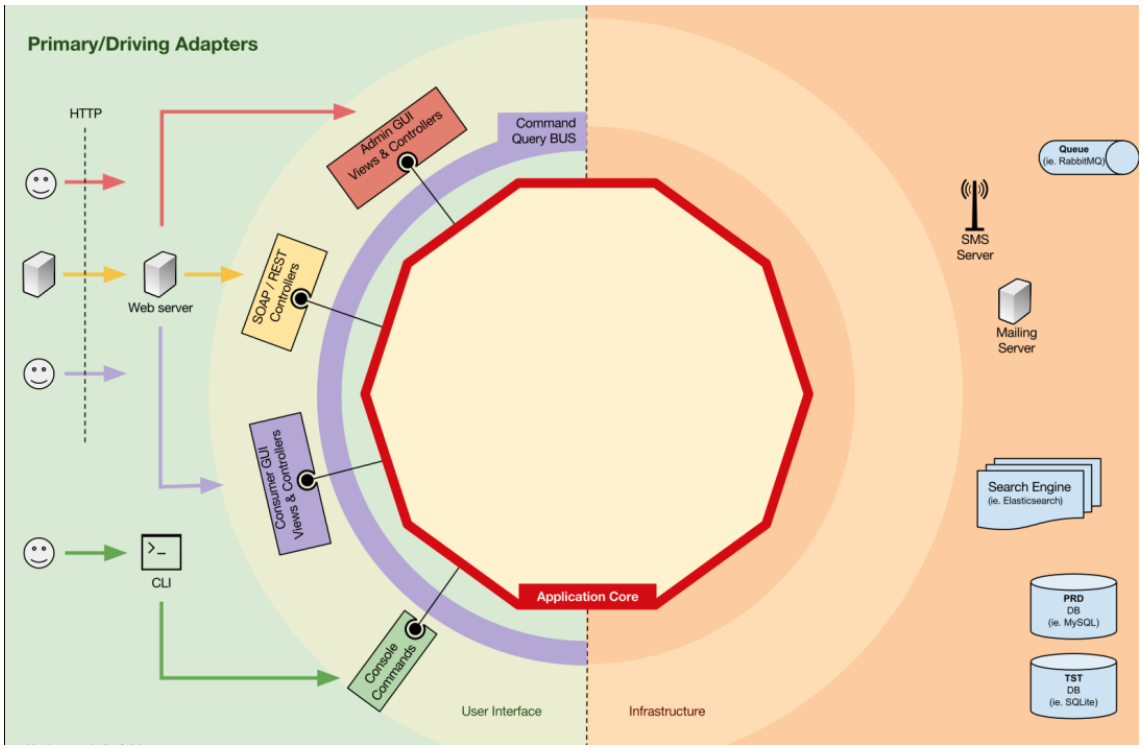


Рис. 2.13 – Архітектура з інтерфейсами (портами) та їх реалізаціями

На відміну від адаптерів драйверів, які знаходяться біля порту, керовані адаптери реалізують порт, інтерфейс, а потім вводяться в ядро програми, де потрібен порт.

Припустимо, що у нас є проста програма, в якій потрібно зберігати дані. Тому ми створюємо інтерфейс збереження, який відповідає його потребам, із методом збереження масиву даних і методом видалення рядка в таблиці за його ідентифікатором. Де б нашій програмі потрібно зберегти або видалити дані, ми потребуємо в її конструкторі об’єкт, який реалізує інтерфейс збереження, який ми визначили.

Тепер ми створюємо адаптер, специфічний для будь-якої бази даних, наприклад MySQL, який буде реалізовувати цей інтерфейс. Він матиме методи для збереження масиву та видалення рядка в таблиці, і ми введемо його всюди, де потрібен інтерфейс збереження.

Якщо в якийсь момент ми вирішимо змінити базу даних, скажімо, на PostgreSQL або MongoDB, нам просто потрібно створити новий адаптер, який реалізує інтерфейс збереження і є специфічним для PostgreSQL, і ввести новий адаптер замість старого.

Інверсія контролю - характерним для цього шаблону є те, що адаптери залежать від певного інструменту та певного порту (за допомогою реалізації інтерфейсу). Але наша бізнес-логіка залежить лише від порту (інтерфейсу), який розроблено відповідно до потреб бізнес-логіки, тому вона не залежить від конкретного адаптера чи інструменту.

Це означає, що напрям залежностей – до центру, це інверсія принципу управління на архітектурному рівні.

Важливо, щоб порти були створені відповідно до потреб ядра програми, а не просто імітували API інструменти.

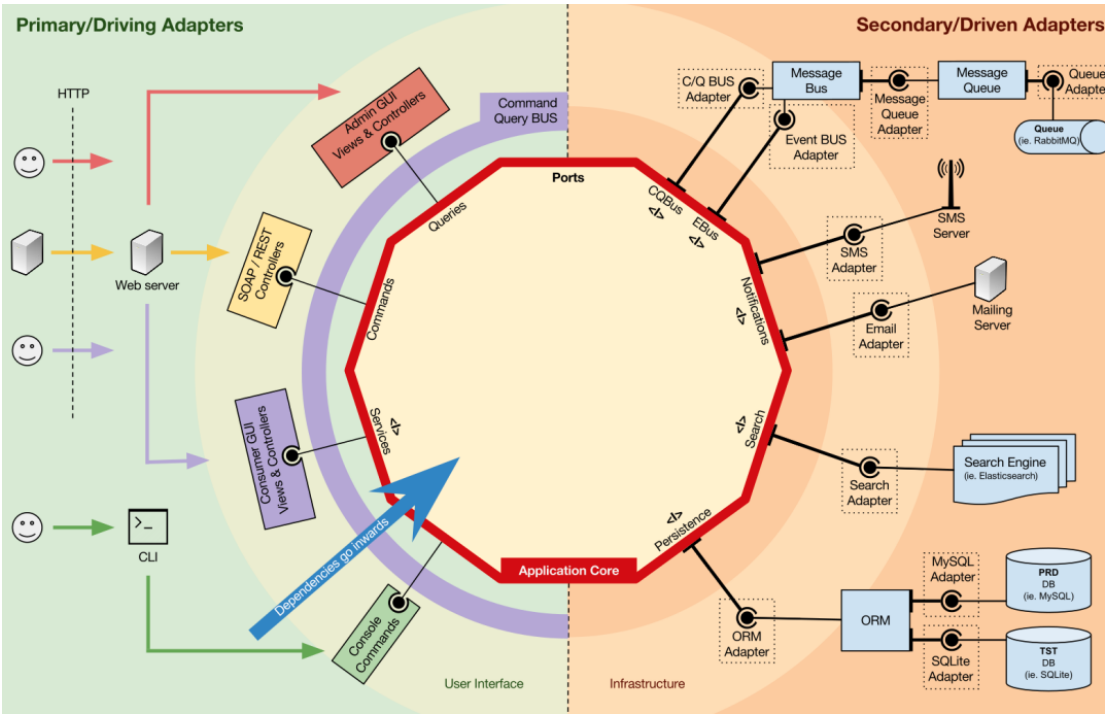


Рис 1.16 – Схема інверсії залежностей

Архітектура збирає шари DDD і включає їх в архітектуру портів і адаптерів. Ці шари призначені для того, щоб привнести певну організацію в бізнес-логіку, внутрішній «шестикутник» портів і адаптерів, і, як і в портах і адаптерах, напрямок залежностей спрямований до центру.

Тепер детальніше обговоримо роль ядра програми (Application Core).

Випадки використання — це процеси, які можуть бути запущені в ядрі програми одним або кількома інтерфейсами користувача в нашій програмі. У CMS ми можемо мати інтерфейс певної програми, який використовується звичайними користувачами, інший незалежний інтерфейс користувача для адміністраторів CMS, інший інтерфейс командної команди та веб-API. Ці інтерфейси можуть ініціювати випадки використання, які можуть бути специфічними для одного з них або повторно використовуватися кількома з них.

Цей рівень містить служби додатків (та їх інтерфейси), але він також містить інтерфейси (порти) портів і адаптерів, які включають ORM,

інтерфейси пошукових систем, інтерфейси обміну повідомленнями тощо.

Коли ми використовуємо шину з команд та/або шину запитів, на цьому рівні знаходяться відповідні обробники команд і запитів.

Служби додатків та/або обробники команд містять логіку розгортання випадку використання, бізнес-процесу. Їх роль полягає в тому, щоб:

- використовувати репозиторій для пошуку сутностей;
- об’єктам можна просто сказати виконати певну логіку домену;
- використовувати репозиторій, щоб зберігати сутності;

Обробники команд можна використовувати двома різними способами:

- вони можуть містити фактичну логіку для виконання якогось сценарію;
- можна використовувати як елементи прогону в нашій архітектурі, отримуючи команду і запускаючи логіку, яка існує в службі додатків.

Цей рівень також містить ініціювання подій програми, які представляють певний результат варіанту використання. Ці події запускають логіку, яка є побічним ефектом варіанту використання, наприклад надсилання електронних листів, сповіщення стороннього API, надсилання push-повідомлення або навіть запуск іншого варіанту використання, який належить до іншого компонента програми.

Область домену - об’єкти в цьому шарі містять дані та логіку для маніпулювання цими даними, які є специфічними для самого домену, і вони незалежні від бізнес-процесів, які запускають цю логіку, вони незалежні та повністю не знають про рівень додатків.

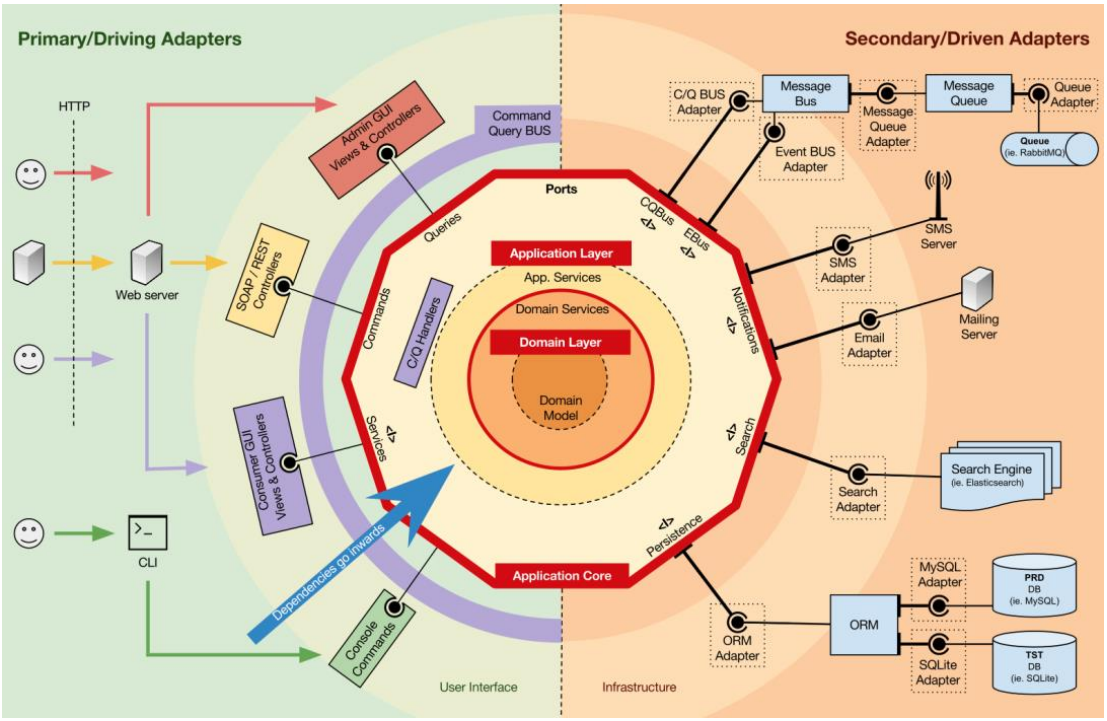


Рис 2.14 – Опис шару домену та прилеглих шарів

Іноді ми стикаємося з певною логікою домену, яка включає різні сутності, одного типу чи ні, і ми знаємо, що ця логіка домену не належить до самих сутностей, ми відчуваємо, що ця логіка не являється їх прямою відповідальністю.

Нашим першим ділом може бути розміщення цієї логіки за межами сутностей, у службі додатків. Однак це означає, що цю логіку домену не можна буде використовувати повторно в інших випадках: логіка домену повинна залишатися поза межами прикладного рівня!

Суть полягає в тому, щоб створити частину домену, роль якої є отримання набору сутностей і виконання певної бізнес-логіки для них. Служба домену належить до рівня домену, і тому вона нічого не знає про класи на рівні додатків. З іншого боку, він може використовувати інші служби домену і, звичайно, об’єкти моделі домену.

У самому центрі, знаходиться домен, який містить бізнес-об’єкти. Вони представляють щось конкретне у домені. Прикладами цих об'єктів є, сутності, але також об'єкти-значення, перерахування та інші різні

об'єкти, що використовуються в домені.

У моделі домену також знаходяться події домену. Вони запускаються, коли певний набір даних змінює свою структуру, і вони несуть ці зміни з собою. Коли сутність змінюється, ініціюється подія домену, яка несе нові значення змінених властивостей. Ці події підходять, наприклад, для використання в пошуку подій.

2.7 Проектування сценаріїв роботи Telegram-боту

Для того, щоб токен боту був в безпеці було написано метод для зчитування файлів. Файл в якому знаходиться токен, не потрапляє ні в систему контролю версій (GIT) ні в інші відкриті джерела.

Коли користувач користується даним ботом, всі його дії логуються в спеціальний файл-журнал логів (api.log). Щоб задати записам в журналі певної форми, можна її описати в файлі logback.xml. Цей метод логування використовується для відлову різних помилок, які виникають при роботі бота.

Сценарієм, називається люба доступна активність, яку обирає користувач. Щоб обрати якусь активність, потрібно написати або обрати з меню любу доступну команду. Щоб зрозуміти, що робить та чи інша команда, в меню є короткий опис функції.

Якщо користувач введе невірну команду, або якусь інше не дозволене слово на будь-якому етапі, бот може підказати, що користувач робить не так, схема доступних команд бота на рисунку 2.15.

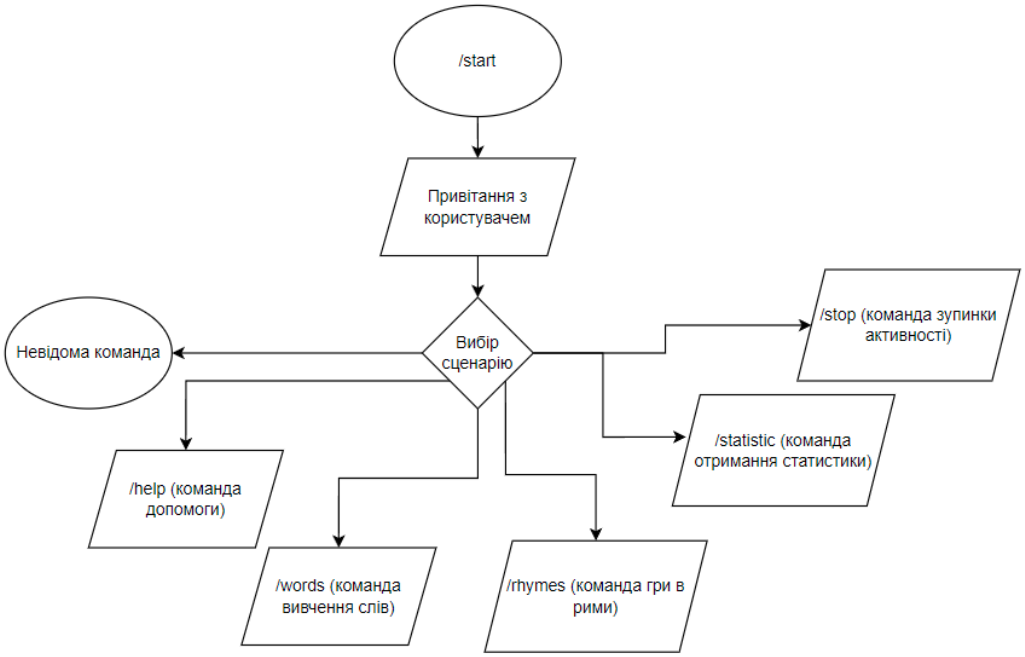


Рис 2.15 – Схема доступних команд

На стороні Java-додатку, люба команда, яка викликається користувачем, являється звичайним повідомленням, через це з об’єкту повідомлення ми не можемо зрозуміти в якому саме сценарії знаходиться користувач.

Наприклад, користувач знаходиться в сценарії гри в рими і пише риму на те слово, яке йому дав бот. Тепер на стороні додатку треба зрозуміти, до якого сценарію відносити це повідомлення. Через це було вирішено написати свою стейт-машину (state-machine), або як можна ще її назвати – спеціалізована сесія, про яку буде написано пізніше.

В сценарії вивчення слів (/words), треба дозволити користувачеві повідомити бота чи він зрозумів слово. Для цього в повідомленні з словом, користувачеві відправляється дві кнопки з текстом:

- GOT IT – дає команду ботові, що користувач зрозумів слово і можна переходити до наступного;
- EXPLAIN – команда ботові, що користувач не зрозумів слово і треба допомога медіа-ресурсів;

Якщо користувач обирає EXPLAIN, то бот розуміє, по якому слову

йому треба знайти медіа-файл в інтернет-ресурсах і показати користувачеві, щоб допомогти зрозуміти те чи інше слово, приклад на рисунку 2.16.

Рис 2.16. – Сценарій пояснення слова

Якщо медіа-матеріал виявився невдалим, можна натиснути повторно на кнопку EXPLAIN і бот надасть наступний матеріал по даному слову.

Наступна, не менш важлива команда – зупинка активності (/stop). Дана команда дає зрозуміти стейт-машині, що потрібно припинити любу активність і повернутися в стан вибору команди.

Команда статистики (/statistic), повинна давати можливість подивись, коли ви вперше написали ботові і скільки ви вивчили слів.

Команда гри в рими (/rhymes), дає можливість пограти з ботом і проявити свою креативність. Під час гри бот дає слово, на яке користувач відповідає римою і бот може визначити чи реально слово римується з введеним словом користувача чи ні.

2.8 Принцип роботи стейт-машини (state-machine)

Як описувалося вище, стейт-машина використовується для того, щоб зрозуміти на стороні додатку, в якому саме сценарії зараз знаходиться користувач.

Для того щоб уникнути не гарного по принципах чистого коду, Domain Driven архітектури та не практичного switch оператору при визові тої чи іншої команди, було написане «розумне перелічення», яке вирішує та перенаправляє в спеціалізований клас-сервіс, який безпосередньо виконує логічні операції з даними.

```
public enum Command {
    START( name: "/start", (state, commandHandler, update) -> state.handle(update, commandHandler)),
    HELP( name: "/help", (state, commandHandler, update) -> state.handle(update, commandHandler)),
    STOP( name: "/stop", (state, commandHandler, update) -> state.handle(update, commandHandler)),
    WORDS( name: "/words", (state, commandHandler, update) -> state.handle(update, commandHandler)),
    RHYMES( name: "/rhymes", (state, commandHandler, update) -> state.handle(update, commandHandler)),
    STATISTIC( name: "/statistic", (state, commandHandler, update) -> state.handle(update, commandHandler));

    public final String name;
    public final TriConsumer<IState, MessageHandler, Update> stateMachine;

    Command(String name, TriConsumer<IState, MessageHandler, Update> stateMachine) {
        this.stateMachine = stateMachine;
        this.name = name;
    }

    public static Command getByName(String command) {
        return Stream.of(values())
                .filter(e -> e.name.equals(command))
                .findAny().get();
    }
}
```

Був розроблений спеціальний інтерфейс ICommand, який повинен реалізовувати клас-сервіс для того, щоб стейт-машина перенаправила виконання логіки саме в цей клас. Це дає нам можливість слідувати принципу SOLID, KISS та DRY.

SOLID - акронім для перших п'яти принципів (single responsibility, open-closed, Liskov substitution, interface segregation и dependency inversion), що означали п'ять основних принципів об'єктно-орієнтованого програмування та проектування.

Ці принципи, коли застосовуються разом, призначені для підвищення ймовірності того, що програміст створить систему, яку легко

підтримувати і розширювати протягом тривалого часу. Принципи SOLID - це довідник, який може застосовуватися під час роботи над програмним забезпеченням для видалення коду який "неприємно пахне" показуючи програмісту, де саме проводити рефакторінг коду, поки той почне задовольняти сучасні стандарти написання коду. Це частина загальної стратегії гнучкої та адаптивної розробки.

Single Responsibility Principle - принцип єдиного обов'язку (на кожен клас має бути покладено один-єдиний обов'язок). Якщо один клас реалізує 2 функцій, які не мають єдиної цілі. Їх злиття створює ситуацію, коли зміна одного елементу програми може зламати функціональність вже давно готової частини функціоналу.

Open\Closed Principle - принцип об'єктно-орієнтованого програмування, що встановлює таке положення: "програмні сутності (класи, модулі, функції тощо) повинні бути відкриті для розширення, але закриті для зміни"; це означає, що такі сутності можуть дозволяти змінювати свою поведінку без зміни їхнього коду.

Liskov Substitution Principle - принцип підстановки Барбари Лісков, функції, що використовують базовий тип, повинні мати можливість використовувати підтипи базового типу, не знаючи про це. Підкласи не можуть замінювати поведінку базових класів. Підтипи повинні доповнювати базові типи.

Interface Segregation Principle - принцип поділу інтерфейсу говорить про те, що багато спеціалізованих інтерфейсів краще, ніж один універсальний.

Іншими словами великі, об'ємні інтерфейси треба розбивати на дрібні таким чином, щоб клієнти маленьких інтерфейсів знали тільки про ті методи, які необхідні їм у роботі. І щоб за зміни методу інтерфейсу нічого не винні змінюватися клієнти, які цей метод не використовують.

Dependency Inversion Principle - принцип інверсії залежностей, обумовлює те, що залежності всередині системи будуються на основі

абстракцій. Модулі верхнього рівня не залежать від модулів нижнього рівня. Абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій.

Принцип KISS (Keep it short and simple або Keep it simple, stupid) достатньо простий для розуміння, він говорить про те, що при проектуванні та програмуванні, не треба використовувати складніші засоби, ніж необхідно. Принцип декларує простоту системи як основну мету та цінності.

DRY (Don’t repeat yourself) також зрозуміє навіть початківець у програмуванні, він говорить про те, що програміст не повинен повторювати та дублювати код, а використовувати вже написані класи та методи. Цей принцип розробки програмного забезпечення, націлений на зниження повторення інформації різного роду, особливо в системах з безліччю шарів абстрагування, простими словами не пишіть коду, що повторюється, використовуйте принцип абстракції, узагальнюючи прості речі в одному місці

Зараз ми розглянули стейт-машину, яка може зрозуміти, яка саме команда виконується, а наступна визначає до якої команди відноситься відправлене користувачем повідомлення.

```java
@Service
@RequiredArgsConstructor
public class StateHandler implements InitializingBean {

    private final List<IState> states = new ArrayList<>();

    private final AuditEntityRepository auditRepository;
    private final UserMapper userMapper;
    private final StartState startState;
    private final HelpState helpState;
    private final StopState stopState;
    private final WordState wordState;
    private final RhymesState rhymesState;

    @Override
    public void afterPropertiesSet() {
        addStates(List.of(startState, helpState, stopState, wordState, rhymesState));
    }

    public void handle(MessageHandler context, Update message) {
        AuditEntity audit = auditRepository.getLastCommandExecuted(userMapper.telegramUserToUserEntity(MessageProcessor.getUser(message)));
        Command command = Command.getByName(audit.getUserAction());
        IState currentState = getState(command);
        command.stateMachine.accept(currentState, context, message);
    }

    private void addStates(List<IState> states) {
        states.stream()
                .distinct()
                .forEach(this.states::add);
    }

    private IState getState(Command command) {
        return states.stream()
                .filter(state -> state.getName().equals(command.name))
                .findFirst().get();
    }
}
```

Її принцип роботи полягає в тому, що коли користувач виконує якусь команду, ця дія записується в спеціальну таблицю AUDIT (її структура на рисунку 2.17), де зберігаються всі його активності.

За допомогою SQL-запиту знаходимо останнє повідомлення користувача, яке містить знак «/». Цей запит поверне запис з аудиту останнього повідомлення, яке містить якусь команду. Виходячи з цього ми можемо зрозуміти, в якому саме сценарії знаходиться користувач.

AUDIT | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

	ID	CREATED_AT	UPDATED_AT	MESSAGE	USER_ACTION	USER_ID
1	21	2022-05-15 12:55:03	[NULL]	[NULL]	/stop	383,074,416
2	22	2022-05-15 12:55:06	[NULL]	sheriff's sale	/rhymes	383,074,416
3	23	2022-05-15 12:55:54	[NULL]	past	/rhymes	383,074,416
4	24	2022-05-15 12:59:17	[NULL]	[NULL]	/words	383,074,416
5	25	2022-05-16 20:13:41	[NULL]	[NULL]	/start	383,074,416
6	26	2022-05-16 20:13:51	[NULL]	watch spring	/rhymes	383,074,416
7	27	2022-05-16 20:14:38	[NULL]	[NULL]	/stop	383,074,416
8	28	2022-05-16 20:14:39	[NULL]	alcman	/rhymes	383,074,416
9	29	2022-05-16 20:15:22	[NULL]	[NULL]	/stop	383,074,416
10	30	2022-05-16 20:15:24	[NULL]	armorbearer	/rhymes	383,074,416
11	31	2022-05-16 20:15:34	[NULL]	[NULL]	/stop	383,074,416
12	32	2022-05-16 20:15:38	[NULL]	chest	/rhymes	383,074,416
13	33	2022-05-16 20:19:05	[NULL]	[NULL]	/stop	383,074,416
14	34	2022-05-16 20:19:13	[NULL]	salt merchant	/rhymes	383,074,416
15	35	2022-05-16 20:21:49	[NULL]	[NULL]	/stop	383,074,416
16	36	2022-05-16 20:21:57	[NULL]	hypergamy	/rhymes	383,074,416
17	37	2022-05-16 20:22:20	[NULL]	[NULL]	/stop	383,074,416
18	38	2022-05-16 20:22:22	[NULL]	technical flaw	/rhymes	383,074,416
19	39	2022-05-16 20:22:31	[NULL]	gravity-circulation	/rhymes	383,074,416
20	40	2022-05-16 20:22:56	[NULL]	[NULL]	/stop	383,074,416
21	41	2022-05-16 20:22:59	[NULL]	regressiveness	/rhymes	383,074,416
22	42	2022-05-16 20:23:04	[NULL]	[NULL]	/stop	383,074,416
23	43	2022-05-16 20:23:05	[NULL]	creosoted	/rhymes	383,074,416
24	44	2022-05-16 20:23:37	[NULL]	[NULL]	/stop	383,074,416

Рис 2.17. – Структура таблиці AUDIT

2.9 Розробка гри в рими

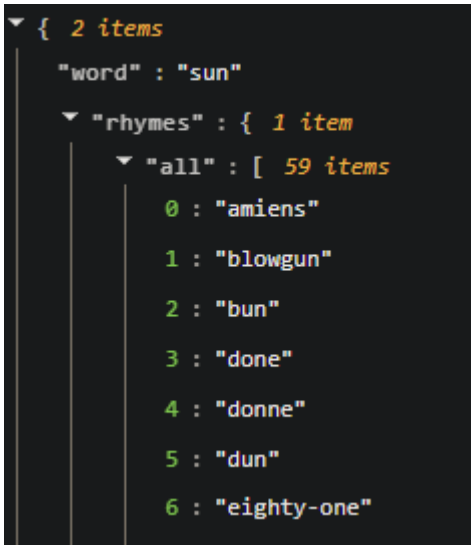
Для того щоб бот міг визначити, чи римується слово насправді яке вводить користувач, використовується спеціальний зовнішній API під назвою WordsAPI. Він дозволяє отримати масив слів які римуються зі словом користувача.

Комунікація з даним API відбувається за допомогою спеціального

API ключа, який грає роль авторизації у їхньому сервісі. Токен зберігається в спеціальному файлі, по тому ж принципу, як зберігається токен від Telegram-боут. Для виклику цього API використовується RestTemplate – інструмент Spring який дозволяє надсилати HTTP запити.

```java
@Component
public class WordsAPI implements WordsProvider {

    @Value("wordsapiv1.p.rapidapi.com")
    private String url;
    @Value("https://wordsapiv1.p.rapidapi.com/words/?random=true")
    private String randomWordUrl;
    @Value("https://wordsapiv1.p.rapidapi.com/words/{word}/rhymes")
    private String rhymesUrl;

    private final RestTemplate restTemplate;

    public WordsAPI(RestTemplate restTemplate) { this.restTemplate = restTemplate; }

    @Override
    public RandomWordModel getRandomWord() {
        HttpEntity<Void> entity = new HttpEntity<>(getWordsApiHeaders());
        ResponseEntity<WordsAPIModel> result = restTemplate.exchange(randomWordUrl, HttpMethod.GET, entity, WordsAPIModel.class);
        return RandomWordModel.of(result.getBody().getWord());
    }

    @Override
    public RhymeWordsModel getRhymesByWord(String word) {
        HttpEntity<Void> entity = new HttpEntity<>(getWordsApiHeaders());
        String url = UriComponentsBuilder.fromUriString(rhymesUrl).buildAndExpand(word).toUriString();
        WordRhymesModel result = restTemplate.exchange(url, HttpMethod.GET, entity, WordRhymesModel.class).getBody();
        return RhymeWordsModel.of(result.getWord(), result.getRhymes().getRhymeWords());
    }

    private HttpHeaders getWordsApiHeaders() {
        HttpHeaders headers = new HttpHeaders();
        headers.add(WORDS_HOST_HEADER, url);
        headers.add(WORDS_API_KEY_HEADER, ConfigurationHandler.getWordsApiKey());
        return headers;
    }
}
```

Відповідь від WordsAPI, також має своєрідну JSON структуру, розглянемо на прикладі слова «Sun»:



Використовуючи відповідь від WordsAPI, ми точно знаємо слова, які підходять. Після її отримання можемо сповістити користувача, чи

справді римується слово чи ні. Якщо воно римується, бот відповідає, що користувач вказав правильне слово, а в випадку коли слово не римується, бот пропонує спробувати написати ще якесь інше слово.

2.10 Розробка команди статистики

Для того щоб правильно зібрати статистику, всі дії користувача зберігаються в базу даних. Наприклад, вивчені слова зберігаються в таблицю LEARNED_WORDS.

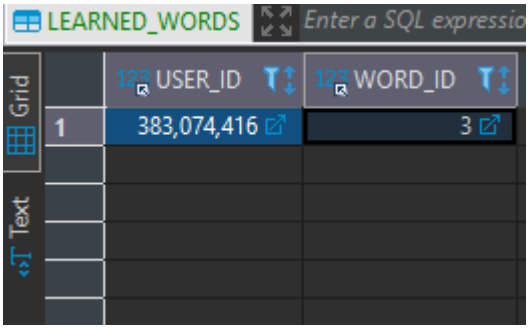


Рис 2.18. – Структура таблиці LEARNED_WORDS

USER_ID – вказує унікальний номер користувача, а WORD_ID – унікальний номер слова з таблиці WORDS. За допомогою цих записів можна вирахувати точне число, скільки слів вивчив користувач.

Завдяки цим даним можна скомпонувати повідомлення та відправити його користувачеві, приклад повідомлення статистики на рисунку 2.19.

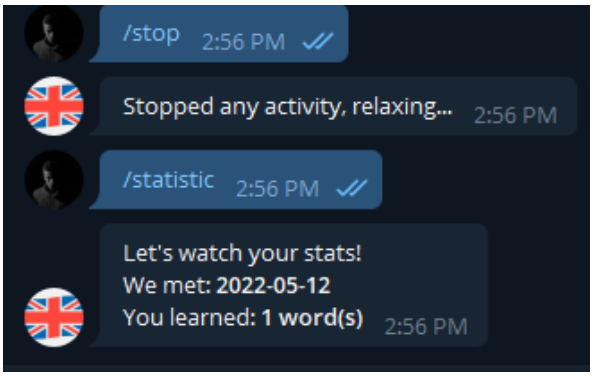


Рис 2.19. – Повідомлення статистики

2.11 Розробка команди вивчення слів

Як описувалося вище, з кожним словом, яке відсилає бот, надсилається клавіатура з кнопками: GOT IT та EXPLAIN. Якщо у випадку натискання кнопки GOT IT все зрозуміло, бот просто надає наступне слово і записує статистику, а якщо натискається кнопка EXPLAIN все не так очевидно, так як зберігати в базі даних медіа-матеріали для сотні тисяч слів спричинить ряд проблем:

- доведеться створити таблицю для зберігання медіа-матеріалів, яка «роздується» до дуже великих розмірів;
- пошук по даній таблиці може зайняти багато часу;
- не дає змоги отримувати безліч медіа-матеріалів по слову, так як ми будемо обмежені кількістю колонок.

Через це було прийнято рішення використати зовнішній API для отримання медіа-ресурсів по ключовим словам під назвою GIPHY.

Взаємодія з даним API відбувається по принципу відсилання в кожному HTTP-запиті, спеціальний токен в параметрах. Токен зберігається в надійному сховищі. При запиті на отримання ресурсів, отримуємо наступну структуру даних:

```
{
  "data": {
    "type": "gif",
    "images": {
      "original": {
        "height": "480",
        "width": "640",
        "size": "610044",
        "url":
"https://media2.giphy.com/media/jk9L41aToGZQA/giphy.gif?cid=0fe95b02whioopd0l1gpnky8k
g62ui3t0eiywossulufhm1a&rid=giphy.gif&ct=g",
        "mp4_size": "130634",
        "mp4":
"https://media2.giphy.com/media/jk9L41aToGZQA/giphy.mp4?cid=0fe95b02whioopd0l1gpnky8
kg62ui3t0eiywossulufhm1a&rid=giphy.mp4&ct=g",
        "webp_size": "378672",
      "original_mp4": {
        "height": "360",
        "width": "480",
        "mp4_size": "130634",
        "mp4":
"https://media2.giphy.com/media/jk9L41aToGZQA/giphy.mp4?cid=0fe95b02whioopd0l1gpnky8
kg62ui3t0eiywossulufhm1a&rid=giphy.mp4&ct=g"
    "analytics_response_payload":
"e=Z2lmX2lkPWprOUw0MWFUb0daUUEmZXZlbnRfdHlwZT1HSUZfVFJBTlNMQVRFJmN
pZD0wZmU5NWIwMndoaW9vcGQwbDFncG5reThrZzYydWkzdDBlaXl3b3NzdWx1ZmhtMW
EmY3Q9Zw"
  }
  "meta": {
    "status": 200,
    "msg": "OK",
    "response_id": "whioopd0l1gpnky8kg62ui3t0eiywossulufhm1a"
  }
}
```


Це сильно скорочена структура, але в ній цікавить тільки посилання на файл формату «mp4». Робимо додатковий HTTP запит (GET) по посиланню, яке знаходиться в полі «mp4». В результаті отримуємо масив байтів даного медіа-ресурсу.

Тепер щоб це відправити користувачу в телеграм, будуємо спеціальний об’єкт SendVideo. В цього об’єкту є влаштований «будівельник», реалізація шаблону програмування під назвою «Builder». Даний метод дозволяє в зручному вигляді налаштувати наш об’єкт, якому обов’язково треба вказати унікальний ідентифікатор чату, для того щоб бот знав куди саме відправити повідомлення.

```
private Function<VideoMessage, Message> getMediaContent(MessageHandler messageHandler, Chat chat) {
    return Unchecked.function(content -> messageHandler.execute(SendVideo.builder()
            .chatId(chat.getId().toString())
            .video(new InputFile(content.getVideoResource().getInputStream(), fileName: "mediaContent"))
            .build()));
}
```

Щоб відправити повідомлення користувачу, використовуємо метод execute(), який в ролі параметру приймає нащадків об’єкта BotApiMethod, саме таким і являється наш об’єкт SendVideo.

В результаті виконання методу, користувач отримує відповідь від бота з медіа-контентом слова, яке він не знає.

3. ТЕСТУВАННЯ ПЗ

3.1 Аналіз видів тестування

Для того щоб протестувати роботу даного боту, потрібно розібратися, який саме метод тестування підходить для даного програмного продукту. Тестування ділиться на функціональне та не функціональне.

До функціонального тестування відноситься:

Юніт-тестування - проводиться на окремому блоці або компоненті для перевірки його функціональності. Зазвичай, юніт-тестування проводиться розробником на етапі розробки програми. Кожен блок у модульному тестуванні можна розглядати як метод або функцію. Розробники часто використовують інструменти автоматизації тестування, такі як NUnit, Xunit, JUnit.

Модульне тестування важливе, оскільки ми можемо знайти більше дефектів на рівні модульного тестування. Корисним юніт-тестом можна вважати той тест, який хоча б 1 раз спрацював (показав помилку).

Юніт-тестування має декілька підходів:

- тестування білого ящика — це методика тестування, за якої внутрішня структура або код програми є видимою та доступною для тестувальника. У цій техніці легко знайти лазівки в дизайні програми або помилку в бізнес-логіці. Покриття умов та їх розгалуженням є прикладами методів тестування білої коробки.
- тестування Gorilla — це методика тестування, при якій тестувальник перевіряє модуль програми в усіх деталях. Даний вид тестування проводиться щоб перевірити, наскільки надійна працює програма.

  Наприклад, тестувальник тестує веб-сайт страхової компанії для домашніх тварин, яка надає послугу придбання страхового полісу.

Тестувальник може сконцентруватися на модулі, страхового полісу, і ретельно протестувати його дивлячись на позитивні і негативні тестові кейси (сценаріїв).

Юніт-тестування має низку недоліків:

- 80% написаних юніт-тестів ніколи не знадобляться, так як вони ніколи не спрацюють;
- написання забирає багато часу, коли за цей час можна було написати щось корисніше;

Інтеграційне тестування – це комплексне тестування, не декілька модулів або вся програма тестується як одне ціле. Основними цілями цього виду тестування є пошук дефектів інтерфейсу, з'ясування зв'язків модулів програми та потоку даних між модулями.

Цей тип тестування проводиться на окремих спеціальних екземплярах додатку. Наприклад, користувач хоче купити авіаквиток на веб-сайті якоїсь авіакомпанії. К Користувачі бачать деталі рейсу та інформацію про оплату під час покупки квитка, але деталі рейсу та обробка платежів – це дві різні частини додатку, або навіть різні мікро-сервіси. Інтеграційне тестування має проводитися під час інтеграції веб-сайту авіакомпанії з системою обробки платежів.

Недоліки даного тестування:

- дуже дорога по часу та ресурсам для підтримки, так дані тести треба постійно дописувати та оновлювати;
- не має ніякого впливу в невеликих проектах.

Тестування end-to-end - включає в себе тестування середовища в ситуації, яка імітує реальні умови, наприклад, з'єднання з базою даних, використання HTTP викликів.

Наприклад, тестувальник тестує веб-сайт страхування домашніх тварин. Дане тестування повинно включати в себе перевірку покупки

страхового полісу, додавання ще однієї тварини до страховки, оновлення інформації про кредитні картки в облікових записах користувачів та оновлення інформації про адресу користувача.

Недоліки такого тестування:

- вимагає хорошої технічної підготовки тестувальника.

Smoke тестування – слугує для перевірки критично важливого функціоналу після крупного оновлення додатку. Перевіряється, чи не зачепило оновлення якийсь важливий компонент програми.

Щоразу, коли команда розробників надає нову збірку, команда тестувальників перевіряє її предмет серйозних проблем.

Недоліки даного тестування:

- потребує автоматизації цього процесу, так як цей процес буде займати багато часу у команди тестувальників;
- вимагає високої кваліфікації автоматизаторів.

Для даного програмного продукту найкраще підходить мануальне (ручне) тестування, так як, функціонал любого боту дуже обмежений і витрачати час на покриття інтеграційними тестами, які не дадуть переваг це контрпродуктивно.

3.1 Тестування боту

Мануальне тестування – в даному виді тестування, перевіряються випадки (кейси), які тестувальник проводить вручну без використання будь-яких автоматизованих інструментів. Ціль ручного тестування є виявлення помилок, проблеми та багів в програмному продукті. Саме це тестування допоможе знайти критичні помилки програми.

Тож можна почати зі складання таблиці тест-кейсів(таблиця 3.1).

Команда/Дія	Ціль	Очікуваний результат

/start	Перевірка початкової команди	Повідомлення першого знайомства
Написати будь-що після команди /start	Перевірка на відмовостійкість боту	Повідомлення з закликом почати активність
/stop	Перевірка припинення активності	Повідомлення про припинення активності
Написати будь-що після команди /stop	Перевірка на реакцію бота під час зупиненої активності	Повідомлення для заклику почати нову активність
/help	Перевірка виклику команди допомоги	Повідомлення з допомогою
Написати будь-що після команди /help	Перевірка реакції бота після виклику команди допомоги	Повідомлення з додатковою допомогою
/words	Перевірка функціоналу вивчення слів	Повідомлення зі словом для вивчення та клавіатура з кнопками GOT IT та EXPLAIN.
Натиснути на GOT IT	Перевірка функціоналу отримання слова	Повідомлення з наступним словом
Натиснути EXPLAIN	Перевірка отримання медіа-ресурсу	Повідомлення з медіа-ресурсом
/rhymes	Перевірка гри в рими	Повідомлення зі словом, на яке треба придумати риму
Написати	Перевірка слова на	Повідомлення

будь-що після /rhymes	римування	про вдалу або невдалу риму
/statistic	Перевірка статистики	Повідомлення зі статистичними даними

Таб. 3.1. – Таблиця тест-кейсів

1. Перевіряємо команду /start (рисунок 3.1), отримуємо повідомлення привітання та кнопки активностей.

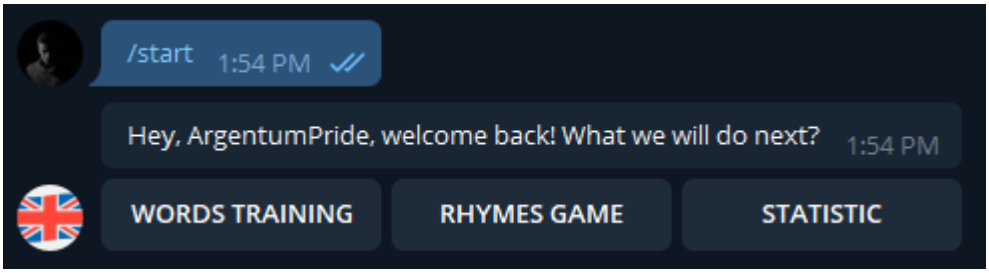


Рис 3.1. – Команда /start

2. Пишемо будь-що після команди /start (рисунок 3.2), отримуємо повідомлення з підказкою, як обрати активність.

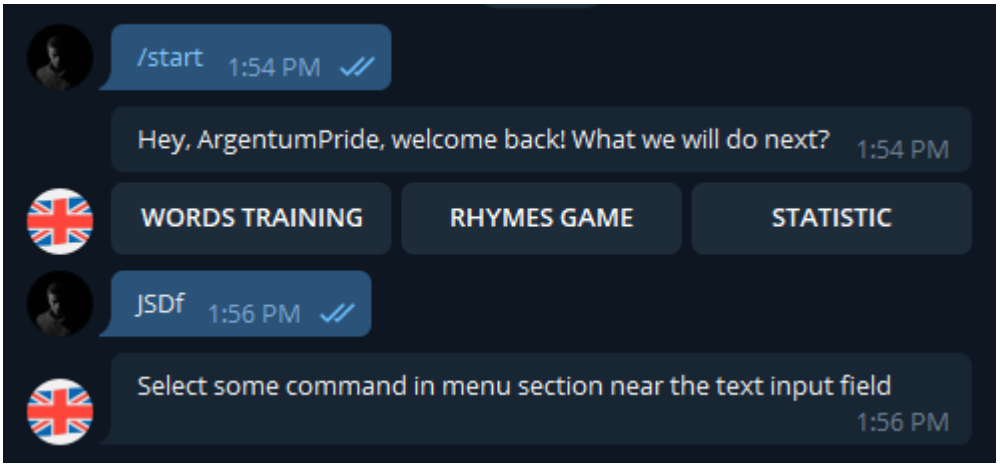


Рис 3.2. – Текст після команди /start

3. Перевіряємо команду /stop(рисунок 3.3), отримуємо повідомлення про те, що активності були припинені.

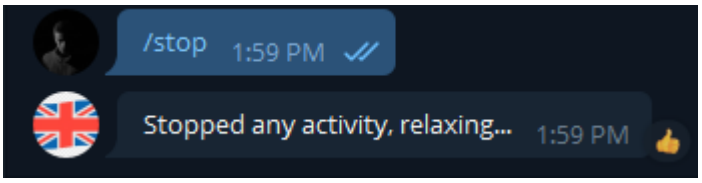


Рис 3.3. – Команда /stop

4. Пишемо будь-що після команди /stop (рисунок 3.4), отримуємо заохочувальне повідомлення.

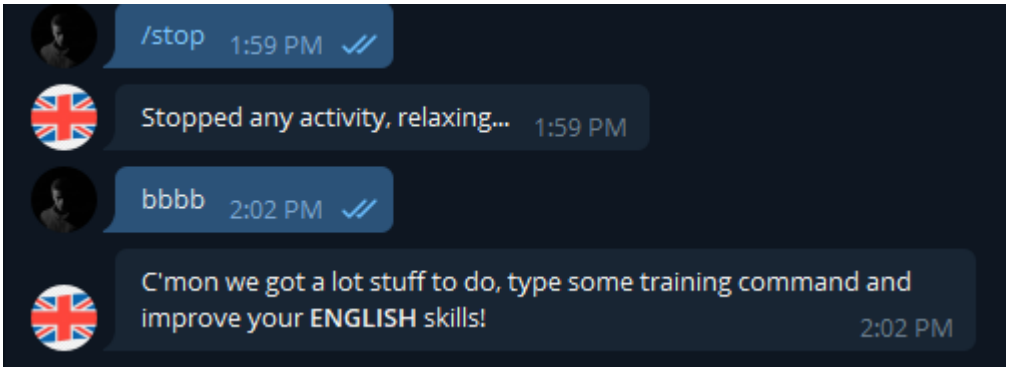


Рис 3.4. – Текст після команди /stop

5. Перевіряємо команду /help (рисунок 3.5), отримуємо повідомлення з допомогою.

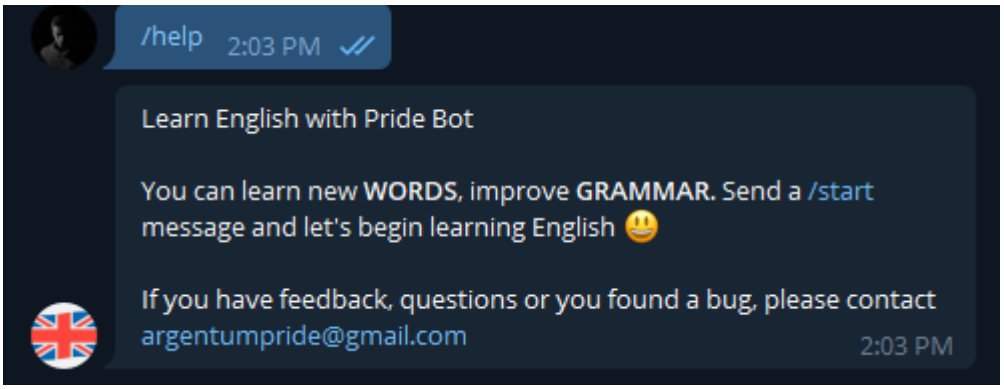


Рис 3.5. – Команда /help

6. Пишемо будь-що після команди /help (рисунок 3.6), отримуємо повідомлення для зворотного зв’язку з розробником.

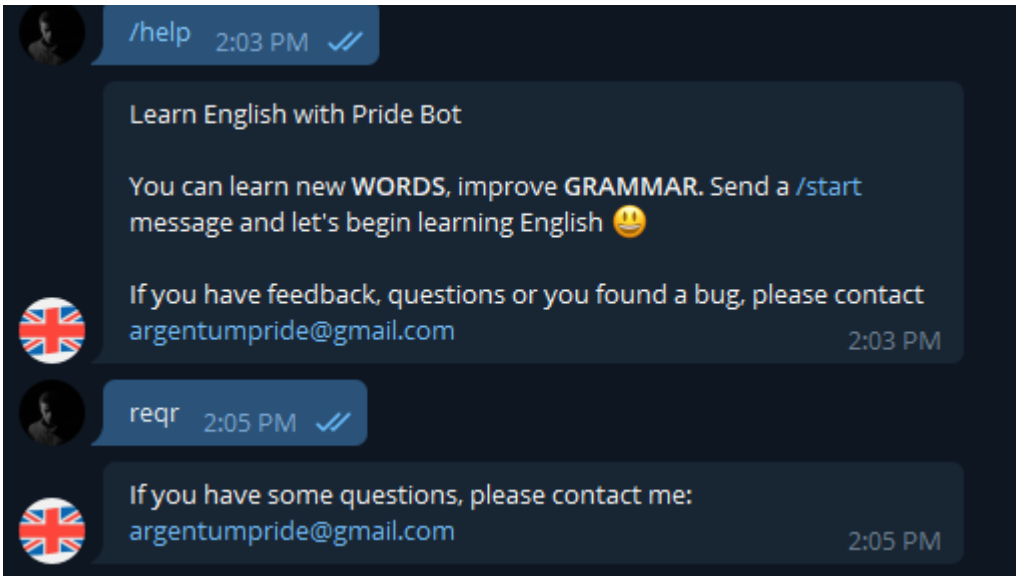


Рис. 3.6. – Текст після команди /help

7. Перевіряємо команду /words (рисунок 3.7), отримуємо повідомлення з словом для вивчення.

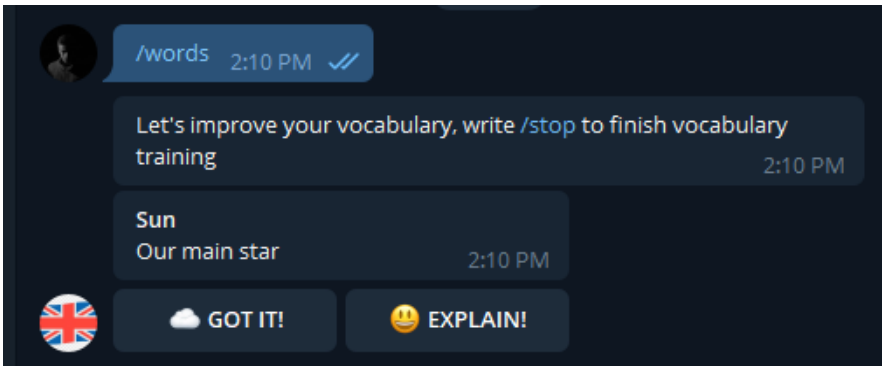


Рис. 3.7. – Команда /words

8. Натискаємо кнопку «GOT IT!» і в результаті отримуємо наступне слово (рисунок 3.8).

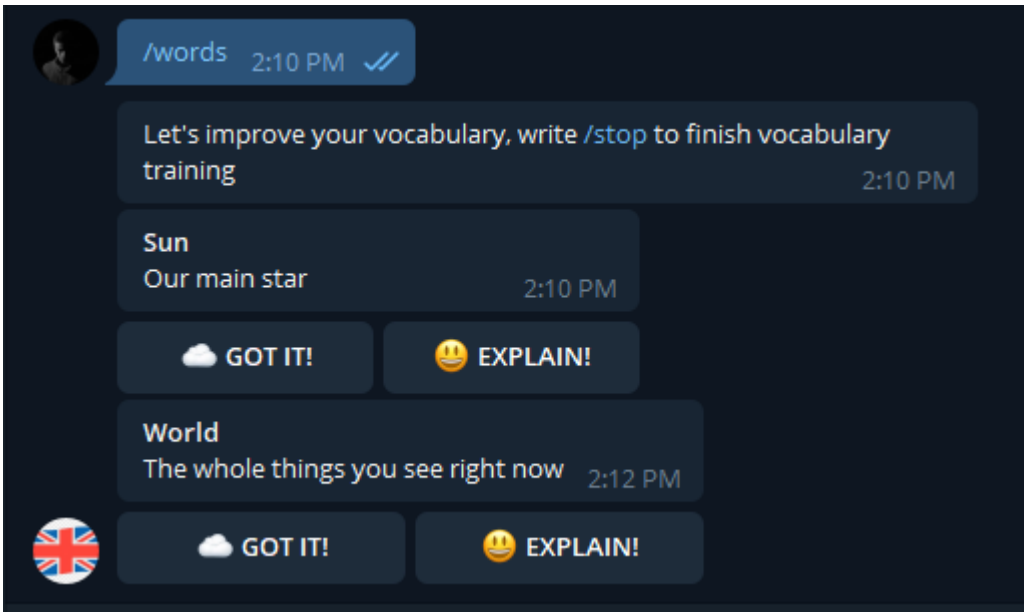


Рис. 3.8. – Результат натискання на кнопку «GOT IT»

9. Натискаємо кнопку «EXPLAIN» і в результаті отримуємо медіа-ресурс, який пояснює дане слово (рисунок 3.9).

Рис. 3.9. – Результат натискання на кнопку «EXPLAIN»

10. Перевіряємо команду /rhymes (рисунок 3.10), отримуємо слово, на

яке треба відповісти римою.

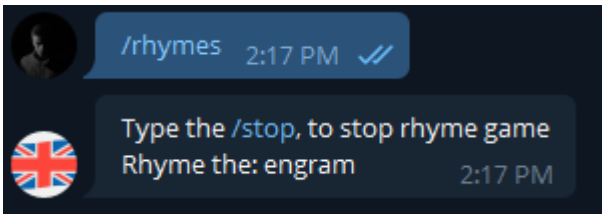


Рис. 3.10. – Команда /rhymes

11. Пишемо у відповідь слово, яке римується зі словом «engram», отримуємо повідомлення про вдалу риму і наступне слово (рисунок 3.11).

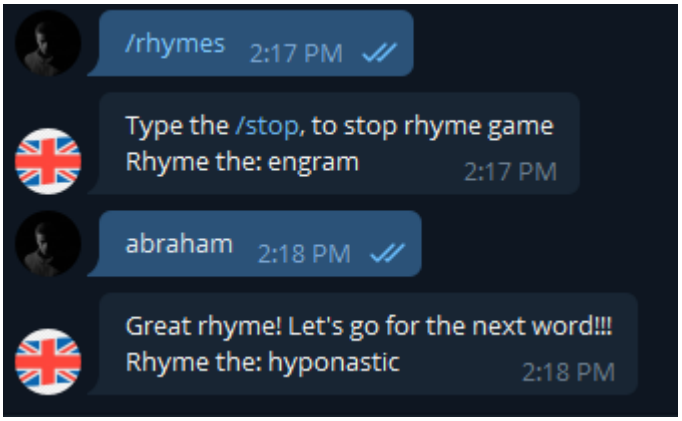


Рис. 3.11. – Гра в рими

12. Перевіряємо команду /statistic (рисунок 3.12), отримуємо кількість вивчених слів та коли було знайомство з ботом.

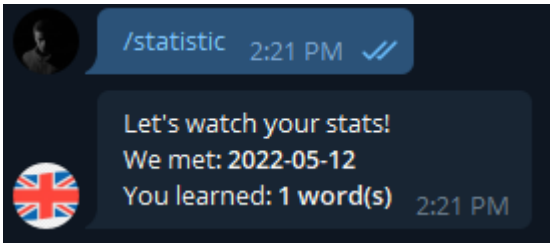


Рис. 3.12. – Команда /statistic

ВИСНОВКИ

В ході виконання дипломної роботи було проведено дослідження предметної області, визначено головні вимоги до телегам-боту. Проаналізовано вимоги до системи в цілому, вимоги до функцій боту, програмного і технічного забезпечення.

1. Було обгрунтовано актуальність розробленої системи шляхом аналізу вже існуючих аналогів та методів вивчення англійської мови. Було досліджено типові методики вивчення англійської мови та виявлено, що в сучасному світі, онлайн навчання стає все актуальнішим з кожним роком.
2. Було розроблено додаток по Domain Driven архітектурі, щоб забезпечити безпеку, гнучкість та розширюваність програмного продукту.
3. Було вибрано інструменти для побудови системи, такі як Java, Spring Framework, Hibernate, модуль TelegramBots, Telegram API, база даних MySQL та Domain Driven архітектура.

В ході розробки системи було спроектовано та реалізовано спеціальна сесія (стейт-машина) для відслідковування теперішньої активності користувача. Було розроблено спеціальну команду для допомоги вивчення слів, також ця команда забезпечує користувача медіа-матеріалами, у випадку коли він не знає те чи інше слово.

4. Високу якість та відповідність боту визначеним вимогам забезпечує розроблена таблиця тест-кейсів, яка покриває всі функціональні вимоги системи, з урахуванням нестандартних випадків, та перевірки їх обробки додатком.
5. Роботу було апробовано серед учнів англійської мови рівнів B1 – B2. Було проаналізовано та виявлено ряд переваг та недоліків боту. Виявлено, що основними перевагами боту є:

- Простота використання;
- Стабільність роботи;

- Наявність різноманітного медіа-матеріалу;
- Наявність гри в рими.

Результати дослідження бакалаврської роботи апробовані на науково-технічній конференції: «Застосування програмного забезпечення в ІКТ».

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційний веб-сайт Java 11 Documentation [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://docs.oracle.com/en/java/javase/11 – Дата звернення 20.04.2022.
2. Java: The Complete Reference, Eleventh Edition./ - Herbert Schildt, 2018 – с. 100-107.
3. Spring in Action./ - Craig Walls, 2018 – с. 247-257.
4. Spring Boot in Action./ - Craig Walls, 2015 – с. 153-160.
5. Офіційний веб-сайт Hibernate ORM [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://hibernate.org/orm/documentation/6.0/ – Дата звернення 21.04.2022.
6. Офіційний веб-сайт MySQL Documentation [Електронний ресурс]: [Веб-сайт]. – електронні дані. –Режим доступу: https://dev.mysql.com/doc/ – Дата звернення 21.04.2022.
7. Офіційний веб-сайт Telegram API [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://core.telegram.org/ – Дата звернення 18.04.2022.
8. Офіційний веб-сайт Telegram Bot API [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу:https://core.telegram.org/bots – Дата звернення 18.04.2022.
9. Ready for changes with Hexagonal Architecture./ - Damir Syrtan, Sergii Makagon – 2020 [Електронний ресурс]: [Веб-сайт]. – електронні дані. Режим доступу: https://netflixtechblog.com/ready-for-changes-with-hexagonal-architecture-b315ec967749 – Дата звернення 20.04.2022.
10. Murach's MySQL./ - Joel Murach, 2014 – с. 85-95.
11. HTTP/2 in Action./ - Barry Pollard, 2019 – с. 127-150.

12. Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design./ - James A. Whittaker, 2019 – c. 165 – 200.
13. Domain Driven Design : How to Easily Implement Domain Driven Design - A Quick & Simple Guide./ - Jason Scotts, 2020 – c. 98 – 153

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУВОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ТЕЛЕГРАМ-БОТУ МОВОЮ JAVAДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

Виконав студент 5 курсу
Групи ППЗ-51
Гончар В.В
Керівник роботи
Зінченко О.В.

Київ - 2022

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

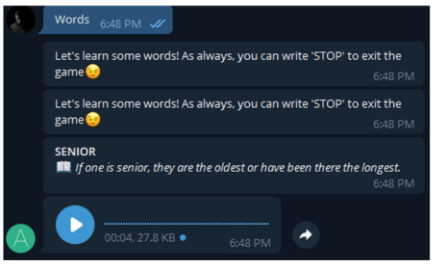
- ***Мета роботи –*** розробка телеграм-боту для вивчення англійської мови
- ***Об'єкт дослідження –*** телеграм-бот для вивчення англійської мови
- ***Предмет дослідження –*** чат-боти, методи вивчення англійської мови

АКТУАЛЬНІСТЬ РОБОТИ

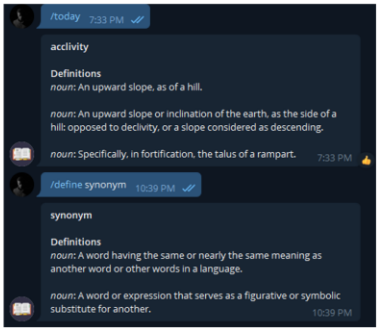
- Актуальність даної роботи заключається в тому, що розроблений телеграм-бот допомагає користувачам (учням), зручно, безкоштовно та ефективно вивчати англійську мову.

- Розроблена система цілеспрямована на допомогу або як додатковий інструмент для тримання рівню знання англійської мови, прямо в кишені. Саме через це було вирішено розробити телеграм-бот, який може задовільнити дану потребу.

АНАЛОГИ



- AndyRobot



- WordBot

ПОРІВНЯННЯ З АНАЛОГАМИ

Назва телеграм-боту	Переваги	Недоліки
AndyRobot	• Надає медіа-матеріали, для кращого розуміння слова • Має гру з використанням емодзі	• має команди-заглушки, які виконують рекламну роль • не дає свободи користувачу у вигляді ігор
WordBot	• Має функції знаходження синонімів, антонімів. • Має функцію «слово дня»	• Не дає можливості потренувати словниковий запас • Не має ігрових команд

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

- Для реалізації телеграм-боту, використовувалась мова програмування Java, в ролі бази даних виступає MySQL і наступні фреймворки та бібліотеки:
- Spring Framework;
- Hibernate;
- TelegramBots;
- MapStruct;

АРХІТЕКТУРА БОТУ

- Бот розроблений за Domain Driven архітектурою (DDD). Її основні принципи полягають у наступному:
- чітке розділення логіки програми (домен) та адаптерів для неї (порт);
- домен не може опиратися на інфраструктуру;
- легка підміна адаптерів (домену не важливо з яким інструментом він працює, його робота заточена на інтерфейсах)

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Гончар В.В. Розробка додатку для вивчення англійської мови: Матеріали науково-технічної конференції: «Застосування програмного забезпечення в ІКТ».

ВИСНОВКИ

- Було обгрунтовано актуальність розробленого чат-бота.
- Було розроблено Domain Driven архітектуру додатку з використанням сучасних ентерпрайз фреймворків.
- Описано програмні засоби, котрі було застосовано для розробки програмного забезпечення.
- Доведено, що розроблений бот підходить для практичного використання людьми, які хочуть підтримувати рівень англійської мови в тонусі.

Перспективи подальших досліджень та розвитку даної роботи полягають у наступному:

- Покращення дизайну повідомлень, які відправляє бот.
- Додання функцій пошуку синонімів, антонімів.
- Розширення звіту у команді статистики.

ДЯКУЮ ЗА УВАГУ