МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

Медецький Сергій Юрійович

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т. В. Нескородєва
«_____» _____ 2022р.

Рекомендаційна система підбору лікарських препаратів на основі Telegram-боту

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Бабаков Р.М., доцент кафедри
інформаційних технологій
д.т.н., доцент
_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Медецький С.Ю. Рекомендаційна система підбору лікарських препаратів на основі Telegram-боту.

У магістерській роботі було проаналізовано предметну область та існуючі аналоги ботів.

Розроблено монолітну архітектуру для додатку, які дозволять вирішити цю задачу.

Розроблено додаток для адміністрування бота та взаємодії з Telegram API.

Проведено аналіз існуючих аналогів, які дозволять вирішити поставлену задачу.

Ключові слова: Java, Cuba Platform, PostgreSQL, Telegram API.

Табл. 21. Рис. 15. Бібліограф : 40 найм.

SUMMARY

Medetskyi S.Y Recommendation system for the selection of medicinal products based on the Telegram bot.

The master's thesis analyzed the subject area and existing analogues of bots.

Developed a monolithic architecture for the application, which will solve this problem.

An application for bot administration and interaction with the Telegram API has been developed.

The analysis of existing analogues which will allow to solve the set task is carried out.

Keywords: Java, Cuba Platform, PostgreSQL, Telegram API.

Tabl. 21. Fig. 15. Bibliography: 40 items.

ЗМІСТ

ВСТУП

Актуальність роботи: Telegram має 200 мільйонів активних користувачів на місяць. Кожен використовує Telegram у своїх особистих цілях. Більшість людей знають, що таке Telegram-боти і користуються ними на регулярній основі. Боти використовуються в багатьох сферах, таких як страхування, комунальні послуги, медична сфера, фінанси і так далі. Фактично, можна знайти спосіб їх використання в будь-якій справі.

Що ж насправді являє собою Telegram-бот? " Telegram-акаунт, який контролюється програмним забезпеченням, іноді з функціями штучного інтелекту- це і є Telegram-бот. Боти можуть виконувати різноманітні функції : проводити навчання, грати, користуватися пошуком, нагадувати про важливі моменти, інтегруватися з іншими сервісами або навіть надсилати команди іншим сервісам".

Існує безліч варіантів Telegram-ботів: боти, які стануть у нагоді при спілкуванні з клієнтами; боти, що нагадують про важливу подію в житті людини, боти на основі веб-сервісів, боти для спеціальних проектів та конкурсів тощо. Кожного дня збільшується кількість користувачів Telegram. Так само збільшується кількість людей, які починають використовувати Telegram-боти у сфері своєї діяльності.

При порівнянні двох компаній, які використовують бота і в яких всі ці дії виконуються людиною, то там, де є бот, все буде виконуватися набагато швидше. В даному випадку компанія, якій було розроблено певного бота для полегшення роботи, буде обробляти всі запити, що надходять, набагато швидше.

Враховуючи все вищезазначене, можна стверджувати, що використання Telegram-бота значно полегшує життя, також він може допомогти вирішити певну проблему. Як саме використовувати бота, залежить від самої людини. Завдяки боту можна отримувати насолоду від відпочинку,з легкістю працювати, згадувати забуті речі тощо.

Мета та завдання дослідження. Мета даної роботи - це полегшення підбору лікарських засобів в аптеці завдяки Telegram-боту. Для досягнення поставленої мети необхідно вирішити наступні завдання:

- зробити аналіз аналогів ботів.
- проаналізувати мови програмування, які існують для створення бота.
- на основі результатів даного аналізу та моделювання розробити Telegram-бота.

Об'єкт дослідження -це засоби обміну миттєвими повідомленнями.

Предмет дослідження- бот як засіб взаємодії з користувачем.

Теоретичне значення отриманих результатів полягає в описі загальноприйнятих принципів побудови бота в додатку Telegram.

Практичне значення отриманих результатів полягає в розробці сервісу для платформи Telegram, який допомагає обрати лікарський засіб.

РОЗДІЛ 1

ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ ТЕНДЕНЦІЙ ТА МОЖЛИВОСТЕЙ РОЗРОБКИ TELEGRAM БОТІВ

1.1 Telegram

Особливості використання Telegram, які були відзначені самими розробниками, продемонстровані в картинках (рис. 1.1)

Рисунок 1.1 – Особливості Telegram [3]

Конфіденційність і безпека. Одразу два важливих моменти. Безпека та захист особистих повідомлень від незнайомих, сторонніх людей стала тією основою, яку хотіли всі люди, тому Telegram став одним з найпопулярніших месенджерів. З кожним роком витоків інформації все більшає. Завжди варто замислюватися про те, що можна використати додатково, щоб все це обійти. У Telegram всі листи зашифровані і можуть самознищуватися.

Зламати дані в Telegram дуже складно, потрібно зламати мозок, навіть якщо диск з інформацією попаде в руки комусь чужому. А повідомлення, які видаляються самі собою, ніде не зберігаються. Дурови дуже серйозно ставляться до захисту особистої інформації. Щороку вони влаштовують конкурси на хакерські атаки, які виявили якісь недоліки в системі. Переможець отримує гроші, а Telegram стає все більш надійним.

Зайти в Telegram можна з будь-якого пристрою - комп'ютера на будь-якій ОС, а також планшета і телефону, адже для зберігання інформації використовуються віддалені сервери.

Швидкість. Дата-центри розташовані в різних частинах міст, країн, континентів, і кожен з них має свою юрисдикцію. Відправка повідомлень займає лічені секунди, буває й менше. Розробники запевняють, що їх сервіс найшвидший. Сервіс на сто відсотків швидше, ніж звичайна відправка звичайних повідомлень і пошти. А рахунок відбувається за долі секунди і саме так відбувається в більшості інших месенджерів.

Всеосяжність. Всі сервери розкидані, так би мовити, по всьому світу. Це робиться для того, щоб ускладнити роботу тим, хто виявить бажання зламати і отримати доступ до особистої інформації інших користувачів. Також це забезпечує стабільну і високу швидкість роботи. Єдиного центру немає, є лише п'ять дата-центрів. Ці центри відповідають за певний регіон.

Відкритість. Кожен користувач спроможний створити свою версію, яка можливо, на його думку, буде покращена або доповнена. І це все завдяки відкритому вихідному коду Telegram та API MTProto. Але насправді існує дуже багато неофіційних версій. Після створення своєї версії, і для того, щоб вона стала офіційною, співробітники компанії ознайомляться з вашим продуктом та приймуть рішення щодо того, наскільки він може бути повноцінним і чи досягає він рівня крутості. MTProto може бути використаним в комерційних цілях, але за умови, якщо гроші не розподіляються між акціонерами. Одним словом, Mail.ru API заборонено використовувати.

Безкоштовно. Завдяки досвіду з "ВКонтакте" новий проєкт Дурова неможна ні продати і не купити. З самого початку Дуров зробив певне пожертвування, і тепер додаток не вимагає ніяких вкладень. Хоча в майбутньому, якщо для Telegram настане чорний день, не виключено, що будуть введені додаткові платні функції, але не факт. Швидше за все, проєкт буде існувати за рахунок інвестицій. Не планується вводити навіть рекламу. А реєстрація в Telegram завжди буде безкоштовною. У компанії всі за безпеку спілкування і просувають її в маси.

Кросплатформеність. Офіційних клієнтів всього три, але зважаючи на це, люди створили безліч версій для будь-якої платформи, навіть створені версії для платформи Linux. Так само і кожен з нас має можливість створити щось своє.

Потужність. Telegram не має жодних обмежень, будь то кількість відправлених за добу повідомлень чи документів, музики чи відео, зображень чи використання додатку як такого. В додатку не заборонено спілкуватися всі 24 години на добу та, при бажанні, сім днів на тиждень. Навіть двісті чоловік може спілкуватися в одному чаті без обмежень. Може бути один чат на всю компанію.

Підтримка. Останній момент, який теж не менш важливий, і про який потрібно знати кожному користувачу. Telegram - це безпечний зв'язок в інтернеті, і чим більше людей розповідають один одному про нього, тим більше з'являється нових користувачів, адже кожен хоче, щоб його дані були недоступні нікому. У розробників немає сторінки в Facebook, і вони не спілкуються ВКонтакте, але у них є офіційний Twitter, де вони розповідають про якісь новинки, про розробку в майбутньому, і іноді відповідають на питання, що часто задаються.

1.1.1 Огляд сучасних ботів-помічників.

Здається, за останні пару років Telegram став чимось більшим, ніж просто месенджер. Тут можна не тільки обмінюватися повідомленнями, а й спілкуватися по відео, а також влаштовувати голосовий чат. Важко повірити в те, що Telegram може стати додатком, в якому буде багато речей, і всі вони будуть непотрібні. Насправді, кожна з цих речей може принести велику користь і зробить користування додатком для багатьох людей набагато простішим.

На даний момент, мабуть, важко знайти людину, яка б не користувалася ботом в Telegram. Можливість створювати такі боти з'явилася в 2015 році, і з того часу дуже велика кількість установ, банків, компаній та інших сфер почали створювати власних ботів. Всім зрозуміло, що боти в Telegram можуть прискорити їх роботу і забезпечити комфортну взаємодію з їх сервісами.

Приклад найвідоміших та найзручніших ботів:

Рисунок 1.2 – бот «Yandex.Translate»

При спілкуванні з людиною набагато зручніше і швидше використовувати бота-перекладача в месенджері. На допомогу прийде Telegram-бот " Yandex Translate ", який інтегрований з перекладачем від Yandex.

Функціонал цього бота дуже простий. Він підтримує 12 мов, серед яких англійська, німецька та інші мови.

Команда /setmylang дозволяє вибирати мову, з якої потрібно перекласти, а команда /tolang – вибирає мову, на яку потрібно перекласти.

Бот «Voicy» здійснює переклад голосових повідомлень на текстові.

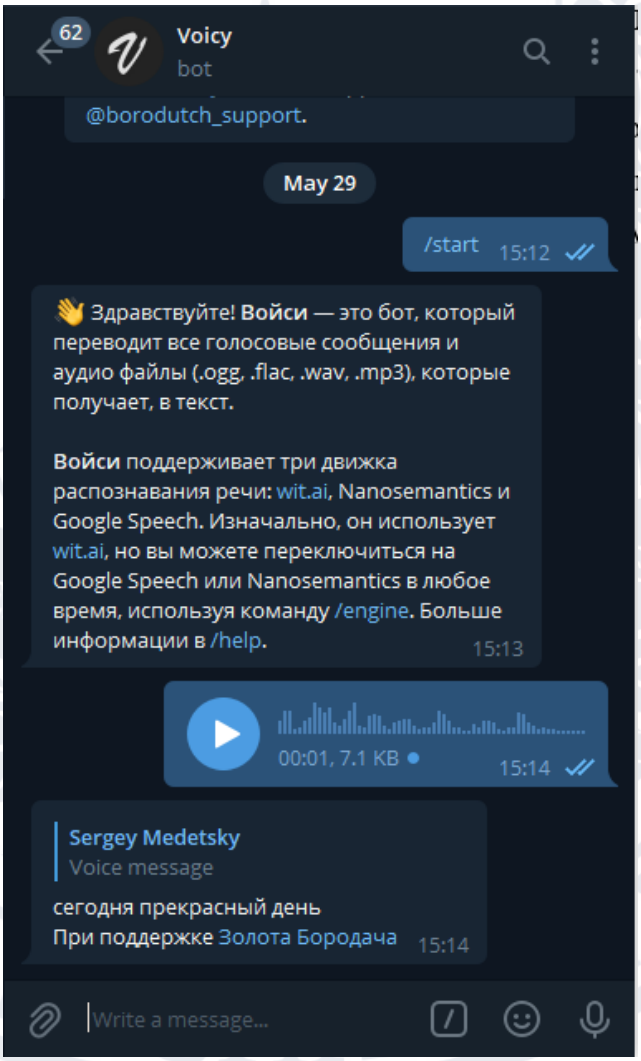


Рисунок 1.3 – бот «Voicy»

Використання бота **Voicy** для людей з певними обмеженими фізичними можливостями – це просто порятунок. Людина може з легкістю прочитати голосові повідомлення, якщо використає цей бот. Адже він працює за дуже простим принципом: перекладає всі голосові повідомлення, а також аудіо-треки таких форматів, як .oog, .flac, .wav, .mp3, в текст. Погодьтесь, що не завжди зручно слухати голосові повідомлення навіть для звичайної людини , адже ситуації можуть бути різними. Саме тому був створений Voicy бот.

Цей бот вже продемонстрував при використанні свою надійність. Він також може обробляти велику кількість мов. Для такого складного завдання команда Voice використовує два двигуни розпізнавання мови - Wit.ai та Google Speech. За бажанням ними можна керувати в налаштуваннях бота.

Бот «SaveAsBot» -бот-завантажувач улюбленого контенту з таких додатків, як Instagram та TikTok.

Рисунок 1.4 – бот «SaveAsBot»

Дуже зручний бот для завантаження матеріалів з Instagram і TikTok. Іноді є бажання завантажити звідти якесь смішне відео. Цей бот займається саме цим і дає можливість завантажити все у відмінній якості.

Коли бот завантажує відео з TikTok, він надсилає разом з відео окремим файлом звукову доріжку, а у випадку з Instagram користувач отримує всі матеріали у файлах для збереження високої якості зображення.

1.1.2 Мови, які підходять для створення Telegram ботів

На сьогодні існує велика кількість мов програмування. Найбільш відомі з них-це Python, C#, Java. Розглянемо їх детальніше.

Мова програмування Python - мова програмування високого рівня, інтерпретована та об'єктно-орієнтована, яка має строгу динамічну типізацію. Динамічна семантика та динамічне зв'язування разом з структурами даних

високого рівня додають їй привабливості при швидкій розробці програм. Також мова Python виступає засобом об'єднання існуючих компонентів [4].

Мова програмування Python базується на підтримці модулів та пакетів модулів, завдяки чому можна повторно використовувати код. Python дозволяє писати код там, де його можна добре прочитати. Через те, що Python дуже лаконічний, всі програми, написані з допомогою цієї мови, будуть набагато меншими за своїм розміром, ніж їх аналоги, які були написані іншими мовами програмування. А ще Python-це багатоплатформова мова і для багатьох це дуже важливий момент. Завдяки цьому програми, які написані на мові Python, можна буде запустити на будь-якій операційній системі без будь-яких змін в коді [5].

Розглянемо мову, яка є не менш відомою та зручною для написання бота в Telegram - C#. C# - об'єктно-орієнтована мова програмування, яка має безпечну систему типізації для платформи .NET [6].

Мова програмування C# була розроблена компанією Microsoft. Вона є однією з найпопулярніших сучасних мов програмування. Ця мова користується попитом на ринку розробки в різних країнах, C# використовується при роботі з програмами для ПК, при створенні складних веб-сервісів чи додатків для мобільних телефонів. Вона з'явилася як мова, яка забезпечує власні потреби платформи Microsoft .NET, але поступово ця мова стала дуже популярною [6].

Синтаксис C# близький до C++ та Java. Для мови характерна сувора статична типізація. Вона підтримує перевантаження операторів, поліморфізм, вказівники на функції, якими можуть бути члени певного класу, різні атрибути чи події, певні властивості,а також виключення та коментарі у форматі XML [6].

Для створення свого бота я вибрав мову програмування Java. Java — об'єктно-орієнтована мова програмування, розроблена Sun Microsystems.

У лютому нового року компанією з відстеження якості програмного забезпечення TIOBE Software було представлено оновлену статистику

найпопулярніших мов програмування. Java посідає друге місце з рейтингом 11,29% [7] (рис. 1.5).

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	^	C	16.34%	-0.43%
2	1	v	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%
6	6		Visual Basic	4.33%	-1.53%
7	7		JavaScript	2.27%	+0.21%
8	8		PHP	1.75%	-0.27%
9	9		SQL	1.72%	+0.20%
10	12	^	Assembly language	1.65%	+0.54%

Рисунок 1.5 Рейтинг мов програмування [7]

Програми, що пишуться на мові Java, майже завжди транслюються в спеціальний байт-код. Ось чому вони можуть бути виконані на будь-якій комп'ютерній архітектурі з реалізацією віртуальної JVM [8].

По-перше, дуже великою перевагою є повна незалежність байткоду від будь-якої операційної системи і устаткування. Це дозволяє програмам, написаним на мові програмування Java, працювати на будь-якому пристрої з відповідною JVM. Ще однією дуже важливою перевагою є гнучка система безпеки. Тому виконання програми повністю контролюється JVM. Програми, які перевищують встановлені дозволи, призводять до негайного переривання [8].

Фреймворк Cuba Platform

Cuba поєднує широко використовувану технологію JVM в ефективну платформу, яка відповідає сучасним стандартам розробки та типовим вимогам корпоративних програм.

CUBA забезпечує більшість функцій Spring природним шляхом, дозволяючи вам покладатися на його велику екосистему та ваш наявний досвід. Якщо ви новачок у Spring, CUBA пропонує простий спосіб розпочати роботу.

Архітектура платформи дозволяє включати будь-яку програму CUBA до складу іншої програми. Це дозволяє легко досягти модульності, розробляючи окремі програмні компоненти та об'єднуючи їх у єдину систему.

Платформа дає можливість створювати горизонтально та вертикально масштабовані рішення. Залежно від запланованого навантаження програми та допустимого часу простою підтримуються різні варіанти розгортання.

Програми CUBA сумісні з популярними реляційними СУБД і можуть працювати в будь-якому контейнері сервлетів Java. Програми можна розповсюджувати як WAR, образи Docker, UberJars або розгортати в хмарі.

Зовнішній інтерфейс користувача. Загальнодоступні інтерфейси часто відрізняються нетиповим дизайном, а також можливістю комфортної роботи з будь-якого пристрою в умовах непередбачуваного навантаження. Для досягнення цих вимог платформа надає генератор коду для розробки інтерфейсу користувача на React.js або Google Polymer і бекенд на CUBA (рис. 1.6).

CUBA пропонує модульну та розширювану архітектуру на основі популярних фреймворків, розроблену для роботи в будь-якому середовищі. Модуль Generic UI значно прискорює розробку внутрішніх інтерфейсів користувача.

Нефункціональні вимоги є підводною частиною айсберга корпоративних додатків, і автоматизація навіть найпростіших бізнес-процесів може стати головним болем. Додатки CUBA підтримують типові потреби, такі як керування доступом користувачів і даних, інструменти адміністрування, звітування та підтримка BPM, майже в один клік.

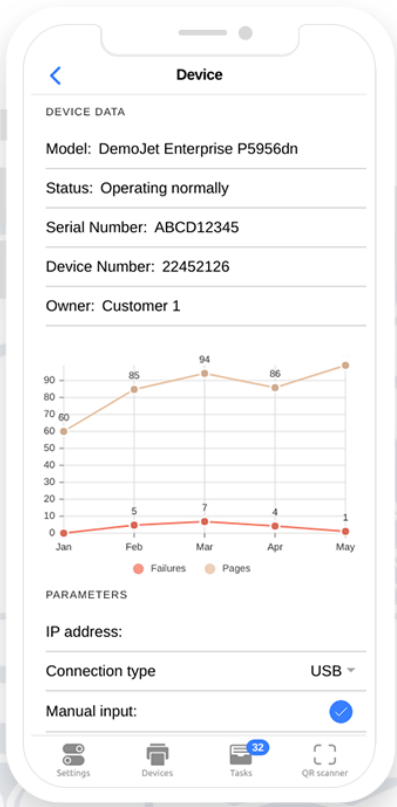


Рисунок 1.6 інтерфейс для зовнішніх користувачів

Завдяки зручним інструментам розробки ви можете швидко інтегруватися в екосистему CUBA. Завдяки генерації коду, візуальному редактору, контекстно-залежним підказкам та інтуїтивно зрозумілій навігації ці інструменти забезпечують справді зручний та ефективний процес розробки.

Перш за все, платформа CUBA спрямована на розробку корпоративних інформаційних систем. Типові особливості цього типу додатків: добре розроблена модель даних, десятки або навіть сотні типових екранів, підтримка великої кількості складних бізнес-процесів, багато звітів, вимоги до розподілу прав доступу тощо.

Ключові особливості платформи:

• Високорівнева абстракція базових технологій - Vaadin, Spring, Ecli pseLink тощо.

• Готові інтегровані компоненти, які допомагають вирішувати типов і для корпоративних систем завдання

• Інструменти візуальної розробки та розвинена генерація шаблонного коду

У цілому це дозволяє мінімізувати час, витрачений на «системні» завдання — налаштування інфраструктури проекту, інтеграцію технологій і компонентів, розробку базового функціоналу — і зосередитися на реалізації бізнес-вимог. У той же час CUBA не обмежує доступ до основного коду, забезпечуючи його здатність адаптуватися до вимог проекту.

Програми на основі CUBA мають стандартну трирівневу архітектуру. Одним із елементів системи є метадані — інформація про модель даних програми. Завдяки метаданим візуальні компоненти знають, з якими даними вони мають справу. Так, наприклад, таблиця знає, що вона відображає властивості сутності «драйвер» і автоматично заповнює імена та формати стовпців. Подібним чином метадані допомагають візуальним компонентам працювати з базами даних через ORM, вказуючи граф об'єктів для завантаження або оновлення. Такий самий принцип застосовується до підсистеми безпеки, генерації звітів і іншим частинам платформи.

CUBA пропонує велику кількість можливостей для створення бізнес-додатків. Якщо вбудованих інструментів платформи недостатньо, додаток можна доповнити зовнішніми компонентами. Платформа включає такі набори функцій:

- управління користувачами та засоби управління
- формування звітів
-керування бізнес-процесами у вбудованому
  візуальному конструкторі
-багатомовний інтерфейс і підтримка різних часових поясів
- пошук повного тексту
- універсальний REST API
- інтеграція з LDAP
- ... та інше

Якщо потрібно розробити інтерфейс користувача за допомогою бібліотек JavaScript, таких як ReactJS, Angular або Vue.js, а також «стандартних» наборів екранів, CUBA надає модуль REST API і генератор TypeScript SDK. Згенерований SDK міститиме всі необхідні класи моделі даних і виклики служб, які можна використовувати під час розробки клієнтських програм.

1.3 База даних PostgreSQL

PostgreSQL — вільна об’єктно-реляційна система керування базами даних (СУБД).

Вона існує в реалізаціях багатьох UNIX-подібних платформ, включаючи AIX, різні системи BSD, HP-UX, IRIX, Linux, macOS, Solaris / OpenSolaris, Tru64, QNX і Microsoft Windows. [13]

Перевагами PostgreSQL вважаються:

• Висока продуктивність і надійність механізмів транзакцій і реплікації;

• Розширена підтримка різних мов програмування: Standard Edition підтримує PL / pgSQL, PL / Perl, PL / Python і PL / Tcl; додатково можна використовувати PL / Java, PL / PHP, PL / Py, PL / R, PL / Ruby, PL / Scheme, PL / sh і PL / V8 і підтримує завантаження модулів розширення мови C[10];

• спадкування;

• Можливість індексації геометричних (особливо географічних) об'єктів і наявність на її основі розширень PostGIS;

• Вбудована підтримка слабо структурованих даних у форматі JSON і можливість індексації;

• Розширюваність , тобто можливість створення нових типів даних, типів індексів, мов програмування, модулів розширення, підключення до будь-якого зовнішнього джерела даних.

Функція SQL виконує довільний набір інструкцій SQL і повертає резул ьтат останнього запиту в списку. У простому випадку буде повернено перши й рядок останнього результату запиту. Якщо останній запит не повернув жод

ного рядка, буде повернено NULL.

Тіло функції SQL має бути списком операторів SQL, розділених крапками з к омою. Крапка з комою після останнього оператора може бути втрачена.

Функції — це блоки коду, які виконуються на сервері, а не на клієнті бази да них. І хоч вони можуть бути написані на чистому SQL, реалізація додаткової логіки, такої як умовні перетворення та цикли, буде виходити за межі SQL і в имагати використання деяких мовних розширень. Функції можуть бути напис ані на одній з наступних мов [14]:

- Вбудована процедурна мова PL/pgSQL, багато в чому схожа на мову PL/SQL, що використовується в СУБД Oracle;
- Скриптові мови - PL/Lua, PL/LOLCODE, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl, PL/Scheme, PL/v8 (Javascript);
- Класичні мови - C, C++, Java (через модуль PL/Java);
- Статистична мова R (через модуль PL/R).

PostgreSQL може підтримувати модифікацію бази даних декількома користувачами, і все це завдяки механізму Multiversion Concurrency Control (MVCC). Через це практично немає необхідності в блокуванні читання і в той же час потрібно дотримуватися вимоги ACID.

Користувач може розширити PostgreSQL під власні потреби практично в будь-якому аспекті. Також є можливість додати свої власні:

-перетворення типів

- типи даних

- домени (визначені користувачем типи з наперед визначеними обмеженнями)

- індекси

-функції (включаючи агрегатні функції)

- оператори (в тому числі перевизначення існуючих)

- процедурні мови

Висновки до розділу 1.

У цьому розділі нами було розглянуто три мови програмування, які є найбільш підходящими для створення Telegram-бота, а також дізналися більше про фреймворк Cuba Platform та розглянули базу даних PostgreSQL.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ПІДБОРУ ЛІКАРСЬКИХ ПРЕПАРАТІВ НА ОСНОВІ TELEGRAM -БОТУ

2.1 Розробка архітектури додатку

Перш за все, важливо правильно розуміти принцип роботи Telegram-бота, з’ясувати, які функції він буде виконувати. Така інформація представлена на рисунку 2.1.

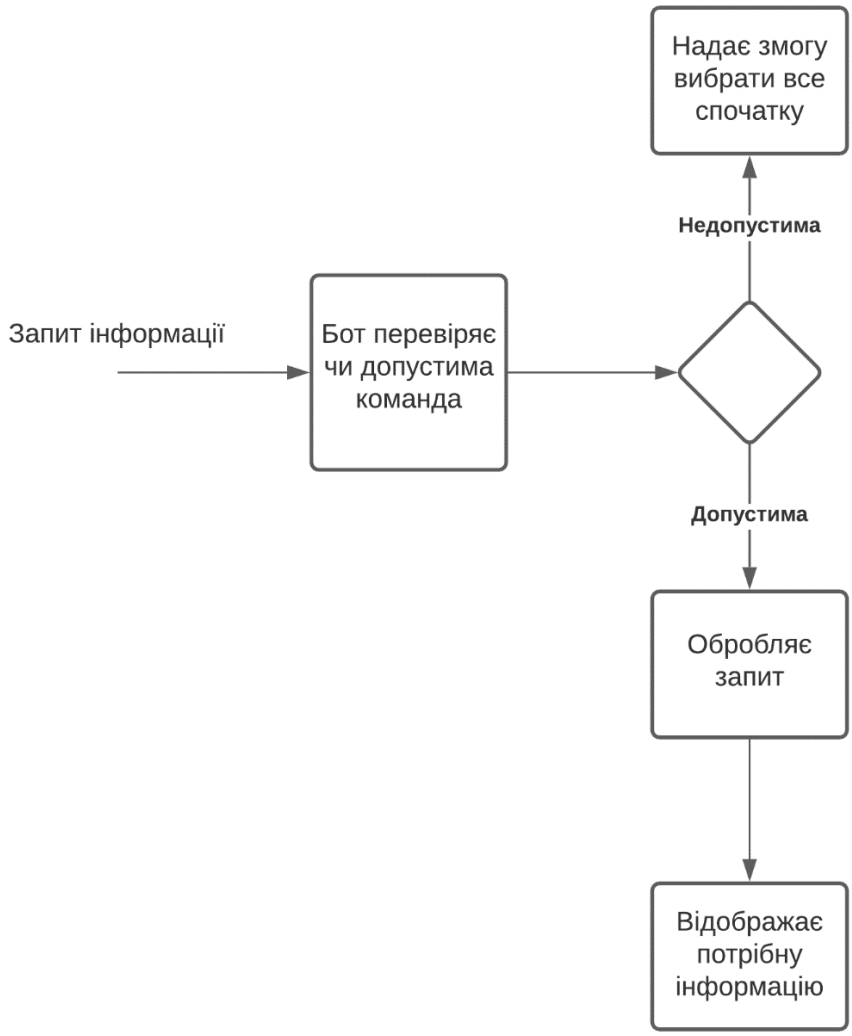


Рисунок 2.1 – Приклад роботи бота

Коли користувач надсилає повідомлення, воно надсилається в зашифрованому вигляді на сервер Telegram. Звідти воно автоматично перенаправляється на Telegram-бота. Далі бот обробляє повідомлення. Якщо воно надіслане для здійснення налаштувань, нові дані записуються і повідомлення надсилається користувачеві.

Переваги правильної архітектури:

- Система має змогу масштабуватись без радикального рефакторинга. Тобто, під час розвитку вам не потрібно буде змінювати фундамент, є змога тільки покращувати. Зміни, яку будуть вноситись в систему, не призведуть до її глобальної переробки.
- Якщо людина захоче залучити програмістів для покращення і розвитку системи, можна найняти різні команди, для того щоб вони не були пов'язані один з одним. Завдяки цьому можна зменшити ризик витоку даних.
- В системі є можливість налаштувати різні категорії для користувачів, і надавати їм доступ до інформації яка їм потрібна.
- Так, як зроблено для одного бота, ви можете підключити багато різних як ботів, так і сервісів.

2.2 Розробка бази даних

Необхідно створити базу даних, яка буде відповідати наступним вимогам:

- збереження вибору, який створив користувач (час створення запису; інформація про людину, яка створила цей запис; час, коли запис було оновлено; інформація про те, хто редагував запис востаннє, час видалення запису, хто видалив запис, варіанти вибору користувача);
- збереження виробника препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис

востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);

- збереження форми випуску препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збереження країни виробника препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збереження фармакотерапевтичної групи (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збереження типу препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збереження приготування препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збережння терміну придатності препаратів (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис та назва препарату);
- збереження реляційного відношення (два унікальних ідентифікатора);
- збереження всієї інформації про препарат (час створення запису; людина, яка створила запис; час, коли оновився запис; людина, яка

редагувала запис востаннє; час, коли запис було видалено; людина, яка видалила запис, про торгову марку, умови відпуску, лікарський склад препарату, адресу виробника, номер свідоцтва про реєстрацію, інформацію про те, чи препарат біологічного походження, чи рослинного походження, чи є препарат-аналог, чи цей препарат є гомеопатичним, дату інструкції, термін придатності).

2.3 Обґрунтування технологій

2.3.1 Мова програмування Java

Мовою програмування для написання Telegram-бота я обрав Java. Java має багато переваг, які змушують багатьох розробників обирати саме її.

Простота - це перша технічна перевага Java. Вона має чіткі правила синтаксису та зрозумілу семантику. Раціональність і лаконічність дуже корисні для обробки коду машинами, які мають обмежений обсяг ресурсів. Для вбудованих пристроїв створено спеціальну платформу Java Micro Edition.

Об'єктно-орієнтований підхід. За 3 десятиліття довів свою ефективність. Суть полягає в тому, що основна увага приділяється даним (об'єктам), інтерфейси та алгоритми є вторинними. Іншими словами, ми відштовхуємося від результату при виборі інструментів, способів їх застосування.

Безпека. Найважливіший критерій, враховуючи використання мови в мережевих/розподілених середовищах. Розробники провели величезну роботу по захисту платформи Java. І вона продовжується. Обійти або зламати механізми захисту надзвичайно складно. За приклад можна взяти використання класів з цифровим підписом. Повні права надаються тільки, коли є повна довіра до автора класу.

Продуктивність. Спочатку викликала питання. Нові версії динамічних компіляторів Java не поступаються традиційним компіляторам, які є на інших платформах. Значний приріст швидкості обробки дається за рахунок оптимізації тих фрагментів коду, які виконуються частіше. При необхідності ті чи інші прийоми оптимізації включаються або відключаються JIT-компілятором.

Надійність -це одна з найважливіших переваг. Програми на Java за будь-яких умов працюють стабільно. Компілятор здатний на ранніх стадіях виявляти помилки, тобто ще до виконання коду. Контроль виконання дозволяє запобігти збоям в пам'яті (наприклад, через неточний вказівник). Самі показники можуть використовуватись не скрізь, а тільки там, де це необхідно (наприклад, при роботі з зв'язковими списками).

Незалежність від апаратного забезпечення та ОС. Важлива лише наявність виконавчого середовища та JVM. А архітектура комп'ютера в цілому не має значення. Байт-код легко інтерпретується на будь-якій машині. Підхід довів свою спроможність багато в чому завдяки динамічній компіляції. Інтерфейс, реалізований в системних бібліотеках, також є кросплатформенним.

Динамічність та адаптивність. Ця особливість дозволяє Java не загубитися в середовищі, яке постійно змінюється. При необхідності в бібліотеки можуть додаватися нові об'єкти та методи. При цьому для використання даних бібліотеки не потрібно чіпати програму. Дуже легко відбувається відстеження інформації про структуру об'єктів, їх поведінку, хід виконання програми.

Зручні та ефективні мережеві можливості. Додатки без проблем знаходять необхідні об'єкти в мережі та відкривають доступ до них. Причому так само легко, як би ми мали справу з локальною файловою системою. Існує велика програмна бібліотека для передачі даних по найбільш поширеним протоколам: FTP, HTTP, TCP/IP. Є механізм виклику віддалених методів.

Не варто забувати, що Java - це тріо, що складається з мови програмування, значної бібліотеки та потужного універсального процесора. Програмістам доступні всі ці напрацювання. Їм не потрібно розробляти з нуля безліч необхідних процедур (наприклад, доступ до мережі, баз даних тощо). Це також є вагомим аргументом на користь Java.

2.3.2 СУБД PostgreSQL

PostgreSQL - вільна об'єктно-реляційна система управління базами даних, яка є дуже популярною. PostgreSQL базується на мові SQL. PostgreSQL розширює її можливості.

Переваги PostgreSQL[18]:

- Можливість підтримки баз даних необмеженого розміру;
- Існування надійних, потужних механізмів для транзакцій і реплікації;
- Можливість розширення систем за рахунок вбудованих мов програмування та підтримка завантаження С-сумісних модулів;
- Можливість успадкування;
- Легка та зручна розширюваність.

Поточні обмеження PostgreSQL:

- Відсутнє обмеження щодо максимального розміру бази даних
- Відсутнє обмеження на кількість записів в таблиці
- Відсутнє обмеження на кількість індексів у таблиці
- Максимальний об’єм таблиці - 32 ТБ
- Максимальний об’єм запису - 1,6 Тбайт
- Максимальний об’єм поля - 1 Гбайт
- Максимальна кількість полів у записі 250-1600 (все залежить від типів полів)

Особливості PostgreSQL:

У СУБД PostgreSQL функції - це блоки коду, які виконуються тільки на сервері, і ніколи на клієнтській базі даних. SQL вимагає використання певних розширень мови, через що виходить за її межі, і завдяки цьому є можливість писати на чистому SQL, де реалізована додаткова логіка, така як переходи і цикли. Є можливість писати функції на різних мовах програмування. Можна використовувати набір записів так, щоб виконання результату запиту було

однаковим, це все допускається при використанні функцій в PostgreSQL. Функції можна використовувати в двох режимах: з правами творця і з правами користувача. Іноді функції порівнюють зі збереженими процедурами, але це дуже різні поняття"[24].

У СУБД PostgreSQL є таке поняття як тригери, вони ініціалізуються як функції. Також вони ініціюються DML операціями. Прикладом може служити операція INSERT, вона може запустити тригер. Ця операція перевіряє доданий запис на відповідність умовам. Для того, щоб написати функцію для тригера, можна використовувати багато різних мов програмування. Також, тригери можна порівняти з таблицями. Якщо тригерів багато, то вони використовуються в афлавітному порядку. [24]

PostgreSQL має так званий механізм правил. Завдяки цьому можна створювати операції вибірки, а також створювати користувальницьку обробку DML-операцій. Є ряд відмінностей від так званого тригерного механізму, але основна відмінність полягає в тому, що правила будуть спрацьовувати на початку створення запиту, до тих пір, поки не буде обраний план виконання, який можна назвати оптимальним і до тих пір, поки цей план не буде виконаний. Також ці правила надають можливість перевизначати стан системи при виконанні SQL-операцій"[24].

У СУБД PostgreSQL існує таке поняття як індекси. Вони бувають наступних типів: R-дерево, GIN, GIST, хеш, B-дерево. Існує можливість створення інших типів індексів, але це не простий процес.

PostgreSQL підтримує багатоверсійність, що означає можливість одночасної модифікації однієї бази даних декількома користувачами. Це відбувається завдяки механізму, який називається Multiversion Concurrency Control, або скорочено MVCC. В результаті роботи цього механізму практично не буде потрібно блокування на читання, а також будуть виконані вимоги ACID.

PostgreSQL розширюється практично для будь-якого виду діяльності. Існує безліч різних можливостей, таких як: додавання власних перетворень типів, додавання типів даних, доменів (призначених для користувача з самого початку з накладеним обмеженням), функцій (є можливість включення агрегатних функцій), індексів, операторів і процедурних мов.

PostgreSQL реалізує спадкування на рівні таблиць. Всі таблиці успадковують набори полів і певну інформацію з інших таблиць, так званих батьківських таблиць. При цьому дані, додані в похідну таблицю, будуть автоматично сприяти (якщо не вказано інше) виконанню запитів, які знаходяться в батьківській таблиці[24].

2.3.3 Фреймворк Cuba Platform

Платформа CUBA є результатом узагальнення більш ніж десятирічного досвіду побудови корпоративних додатків у поєднанні з найсучаснішими технологіями розробки програмного забезпечення.

Продумана архітектура і широкий спектр модулів платформи CUBA надають можливість швидко і успішно вирішувати будь-які завдання автоматизації бізнес-процесів.

Особливості платформи CUBA:

- Зниження витрат замовника на придбання ліцензій за рахунок використання вільно розповсюджуваних технологій.
- Кросплатформеність: можливість роботи з будь-якою операційною системою і практично будь-якою системою управління базами даних (СУБД), в тому числі вільно розповсюджуваними.
- Застосування ефективних засобів мережевої безпеки та розмежування прав доступу користувачів.
- Автоматичний аудит дій користувачів, включаючи історію змін даних.
- Масштабованість та відмовостійкість.

- Простота інтеграції з зовнішніми додатками.
- Простота і швидкість розробки, впровадження та оновлення бізнес-додатків; можливість оновлення без зупинки системи (для критично важливих систем, що працюють цілодобово).
- Можливість роботи в системі, створеній на платформі, з будь-якої точки світу, де є Інтернет.
- Відкритий вихідний код: можливе подальше розширення додатків власними силами замовника.
- Архітектура додатків на платформі CUBA.

Фреймворк дає змогу будувати багаторівневі додатки з виділеними клієнтським рівнем, середнім шаром і рівнем бази даних. Надалі йтиметься здебільшого про середній шар і клієнтів, тому для стислості вираз "усі рівні" означає два ці рівні.

На кожному рівні можливе створення одного або декількох блоків програми. Блок являє собою відокремлену виконувану програму, що взаємодіє з іншими блоками програми. Зазвичай блоки реалізуються у вигляді веб-додатків, що виконуються на JVM.

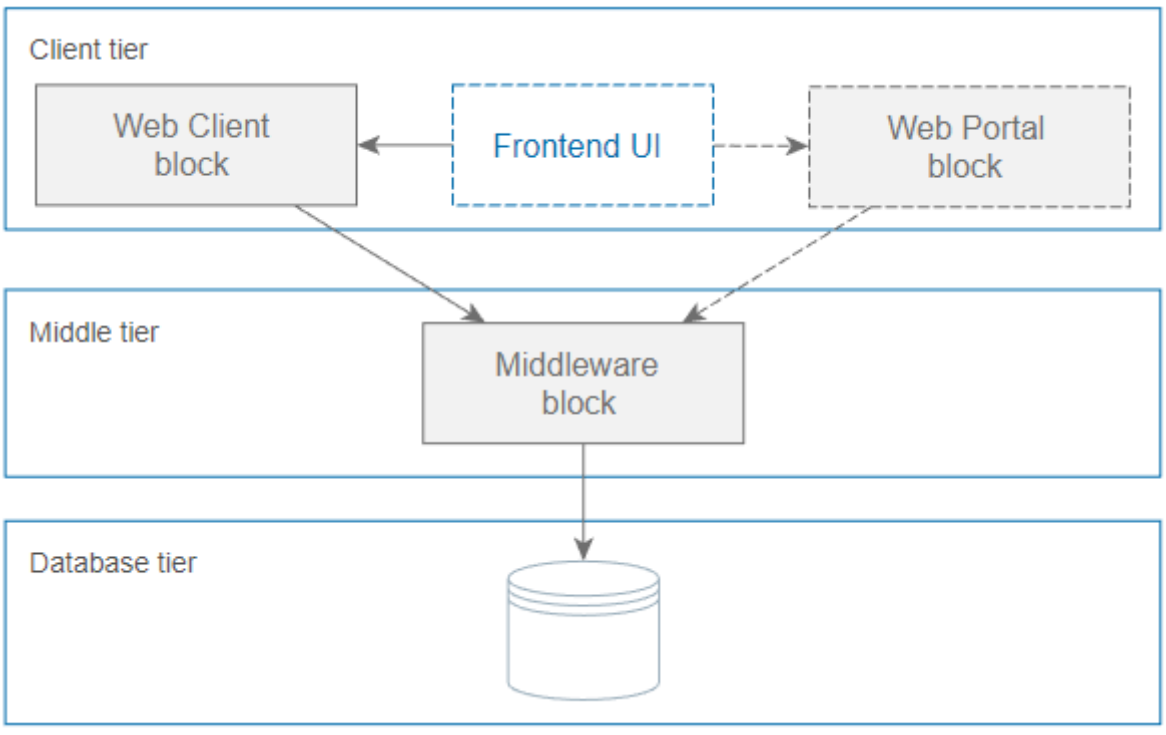


Рисунок 2.2 – Рівні та блоки додатку

Middleware

Середній шар, що містить основну бізнес-логіку додатка і виконує звернення до бази даних. Являє собою окремий веб-додаток під керуванням стандартного контейнера сервлетів Java.

Web Client

Основний блок клієнтського рівня. Містить інтерфейс, призначений, як правило, для внутрішніх (back-office) користувачів організації. Являє собою окремий веб-додаток під керуванням стандартного контейнера сервлетів Java. Реалізація користувацького інтерфейсу заснована на фреймворку Vaadin.

Web Portal

Додатковий блок клієнтського рівня. Може містити інтерфейс для зовнішніх користувачів і засоби інтеграції з мобільними пристроями та сторонніми додатками. Являє собою окремий веб-додаток під керуванням стандартного контейнера сервлетів Java. Реалізація користувацького інтерфейсу заснована на фреймворку Spring MVC.

Frontend UI

Додатковий клієнтський блок, що надає інтерфейс для зовнішніх користувачів. На відміну від Web Portal, є додатком, що виконується на стороні клієнта (приклад: JavaScript-додаток, що виконується у веб-браузері). Працює із середнім шаром через REST API, запущений у блоці Web Client або Web Portal. Може бути заснований на React або інших фронтенд бібліотеках і фреймворках.

Обов'язковим блоком будь-якого застосунку є середній шар - Middleware. Для реалізації користувацького інтерфейсу, як правило, використовується один або кілька клієнтських блоків, наприклад, Web Client і Web Portal.

Усі засновані на Java клієнтські блоки взаємодіють із середнім шаром однаковим чином за допомогою протоколу HTTP, що дає змогу розміщувати середній шар довільним чином, зокрема за мережевим екраном. Слід зазначити, що під час розгортання в найпростішому випадку середнього шару

і веб-клієнта на одному сервері між ними організовується локальна взаємодія в обхід мережевого стека для зниження накладних витрат.

Модуль - найменша структурна одиниця CUBA-додатку. Являє собою один модуль проєкту додатка і відповідний йому JAR файл з виконуваним кодом.

Стандартні модулі:

global - містить класи сутностей, інтерфейси сервісів та інші загальні для всіх рівнів класи. Використовується у всіх блоках програми.

core - реалізація сервісів і всіх інших компонентів середнього шару.

gui - загальні компоненти Універсальний користувацький інтерфейс. Використовується в модулі Web Client.

web - реалізація універсального користувацького інтерфейсу на Vaadin.

portal - опціональний модуль - реалізація веб-порталу на Spring MVC.

front - опціональний модуль - реалізація Фронтенд інтерфейсу на JavaScript.

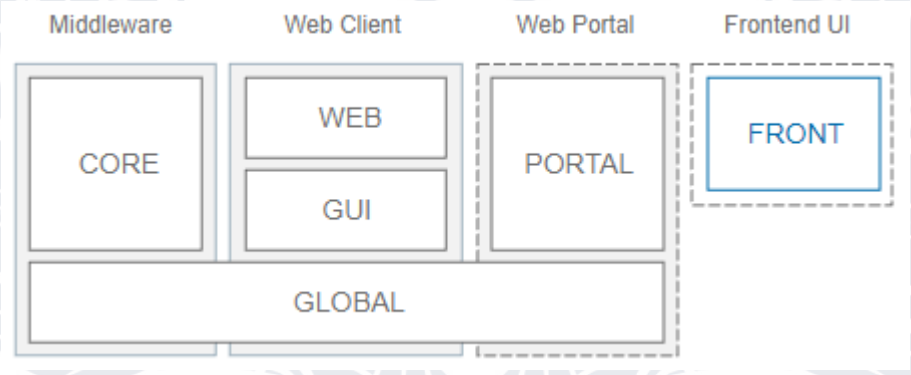


Рисунок 2.3 – Модулі додотку

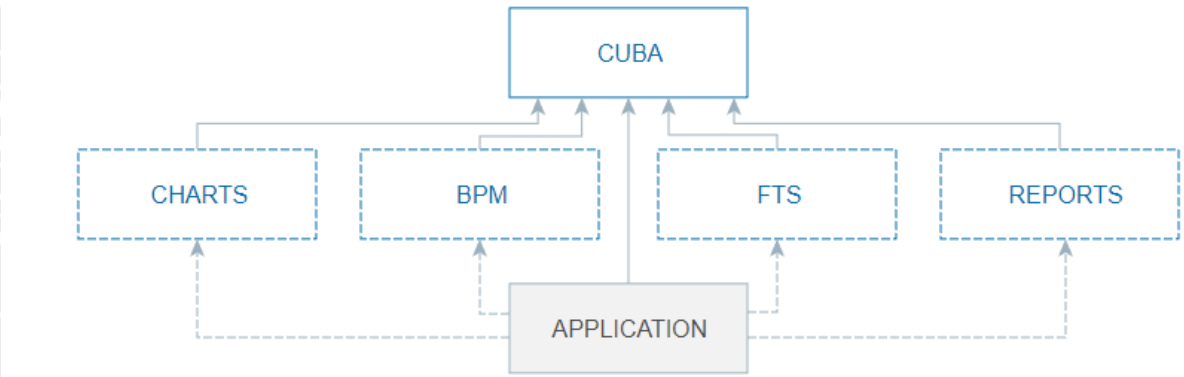
Фреймворк дає змогу розділяти функціональність додатка на компоненти. Кожен компонент (званий також add-on) може мати власну модель даних, бізнес-логіку та користувацький інтерфейс. Додаток використовує компоненти як бібліотеки і включає їхню функціональність.

Концепція компонентів застосунку дає нам змогу зберігати фреймворк відносно невеликим, і водночас надавати опціональну бізнес-функціональність у компонентах, таких як Reporting, Full-Text Search, Charts, WebDAV та інших. Водночас розробники застосунків можуть

використовувати цей самий механізм для декомпозиції великих проєктів на набір функціональних модулів, які розробляються незалежно і мають різний цикл релізів. Звісно, компоненти застосунків можуть бути перевикористовуваними і забезпечувати проблемно-специфічний рівень абстракції поверх фреймворку.

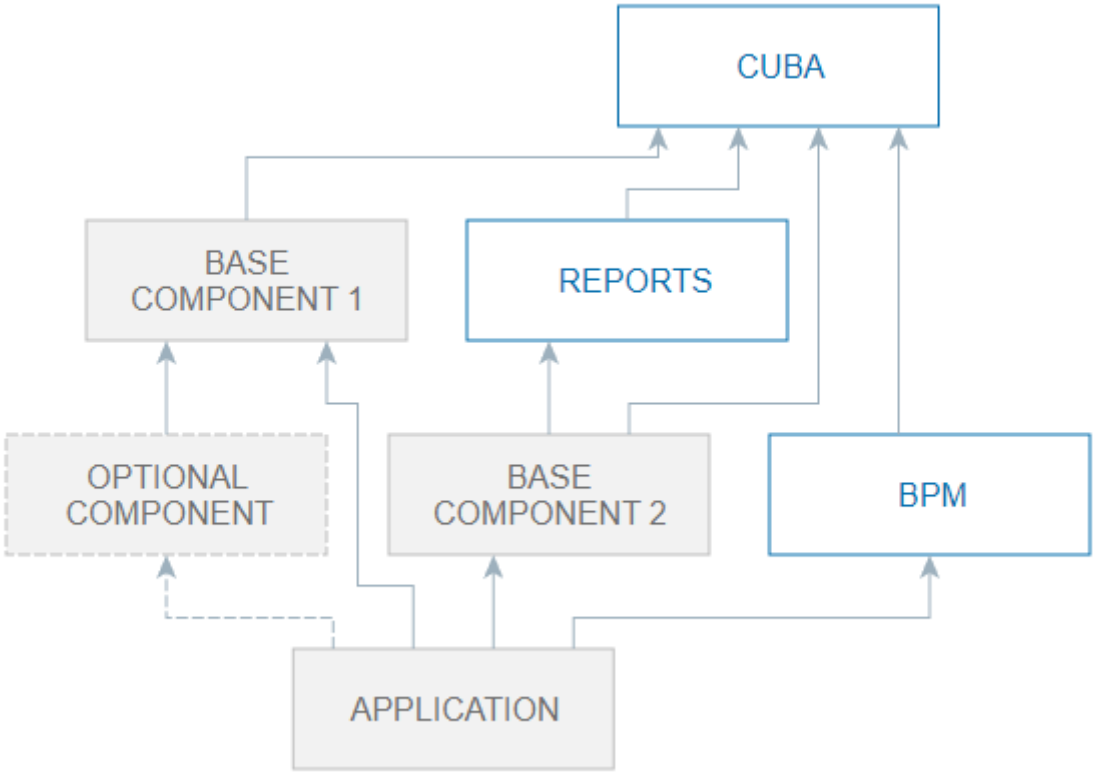
З технічної точки зору, ядро фреймворку також є компонентом під назвою cuba. Єдина його відмінність від інших компонентів це те, що він є обов'язковим для будь-якого застосунку. Усі інші компоненти залежать від cuba і можуть також мати залежності між собою.

Нижче наведено діаграму залежностей між стандартними компонентами, часто використовуваними в додатку. Суцільними лініями зображено обов'язкові залежності, пунктирними - опціональні.



Діаграма 2.1 – Діаграма залежностей

Нижче наведено діаграму можливої структури залежностей між стандартними та кастомними компонентами додатка.



Діаграма 2.2 – Діаграма можливої структури залежностей

Будь-який CUBA-додаток може бути легко перетворено на компонент і надавати функціональність іншому додатку. Для того щоб застосунок можна було використовувати як компонент, він має містити дескриптор app-component.xml і спеціальний елемент у маніфесті JAR модуля global. CUBA Studio дозволяє автоматично згенерувати дескриптор і маніфест для поточного проєкту.

Вищеописані архітектурні принципи безпосередньо відображаються в структурі зібраного додатка. Розглянемо її на прикладі простого застосунку, який складається з двох блоків - Middleware і Web Client; і містить у собі функціональність компонентів cuba і reports.

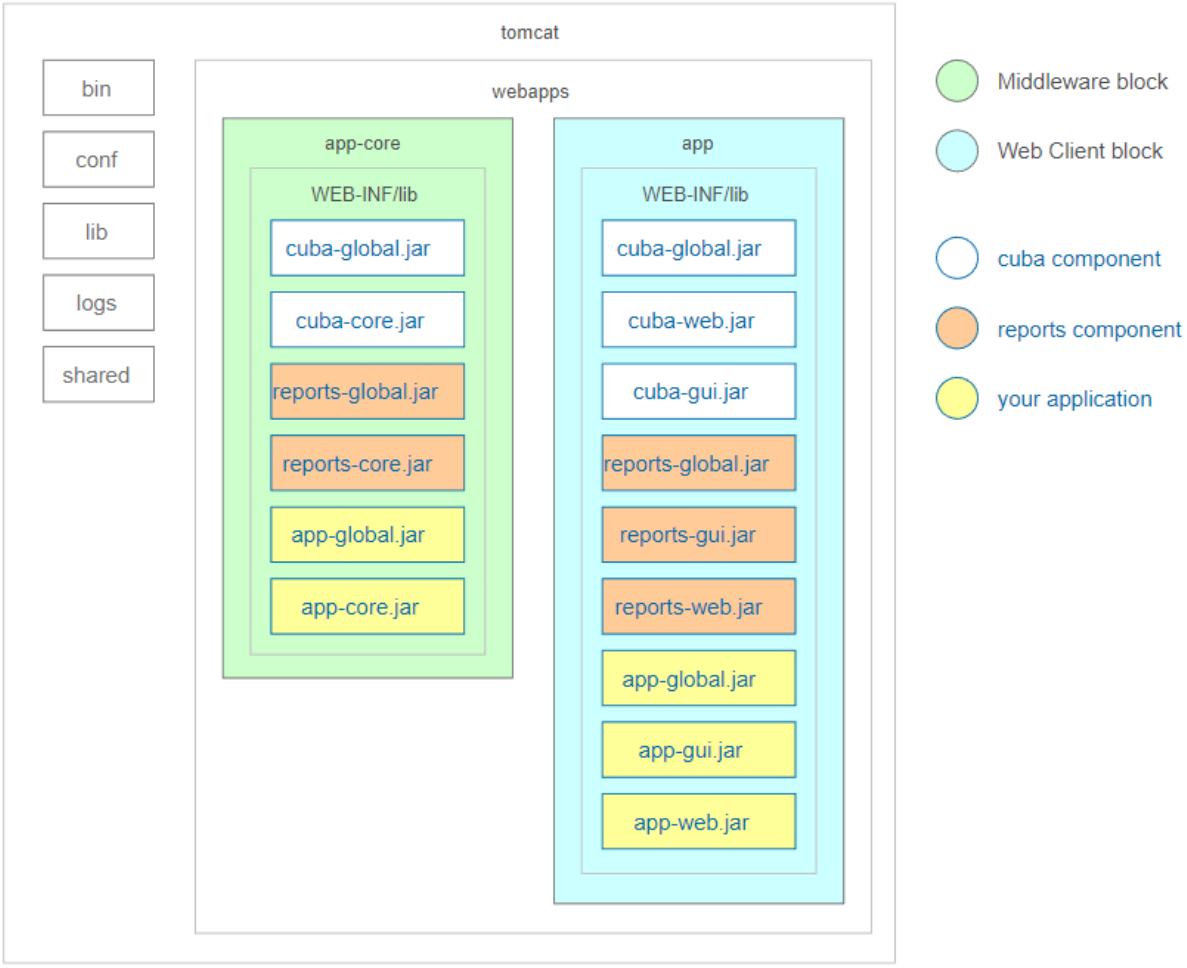


Рисунок 2.3 – Склад простого додатку

На малюнку зображено вміст деяких каталогів сервера Tomcat із розгорнутим у ньому додатком.

Блок Middleware реалізовано веб-додатком app-core, блок Web Client - веб-додатком app. Виконуваний код веб-додатків міститься в каталогах WEB-INF/lib у наборі JAR-файлів. Кожен JAR являє собою результат складання (артефакт) одного з модулів додатка або його компонента.

Наприклад, склад JAR-файлів веб-додатка середнього шару app-core визначається тим, що блок Middleware складається з модулів global і core, і додаток використовує компоненти cuba і reports.

Основне джерело формування структури метаданих - анотовані класи сутностей.

Клас сутності відображається в метаданих у таких випадках:

Клас персистентної сутності анотований @Entity, @Embeddable, @MappedSuperclass і розташований у межах кореневого пакета, зазначеного в metadata.xml.

Клас неперсистентної сутності анотований @MetaClass і розташований у межах кореневого пакета, зазначеного в metadata.xml.

Усі сутності всередині одного кореневого пакета поміщаються в один екземпляр MetaModel, якому присвоюється ім'я цього пакета. Між сутностями всередині однієї MetaModel можна встановлювати довільні зв'язки, між різними - у порядку оголошення файлів metadata.xml у властивості cuba.metadataConfig.

Атрибут сутності відображається в метаданих, якщо:

поле класу анотовано @Column, @OneToOne, @OneToMany, @ManyToOne, @ManyToMany, @Embedded

поле класу або метод доступу на читання (getter) анотований @MetaProperty

Параметри мета-класу і мета-властивостей формуються на основі параметрів перерахованих анотацій, а також типів полів і методів класу. Крім того, якщо в атрибута відсутній метод доступу на запис (setter), атрибут стає незмінним (read only).

Для ефективної роботи з моделлю даних у CUBA-додатках використовується фреймворк метаданих, який:

- надає зручний інтерфейс для отримання інформації про сутності, їхні атрибути та відносини між сутностями; а також для навігації за посиланнями
- слугує спеціалізованою і більш зручною у використанні альтернативою Java Reflection API
- регламентує допустимі типи даних і відносин між сутностями
- дає змогу створювати універсальні механізми роботи з даними

Розглянемо основні інтерфейси метаданих.

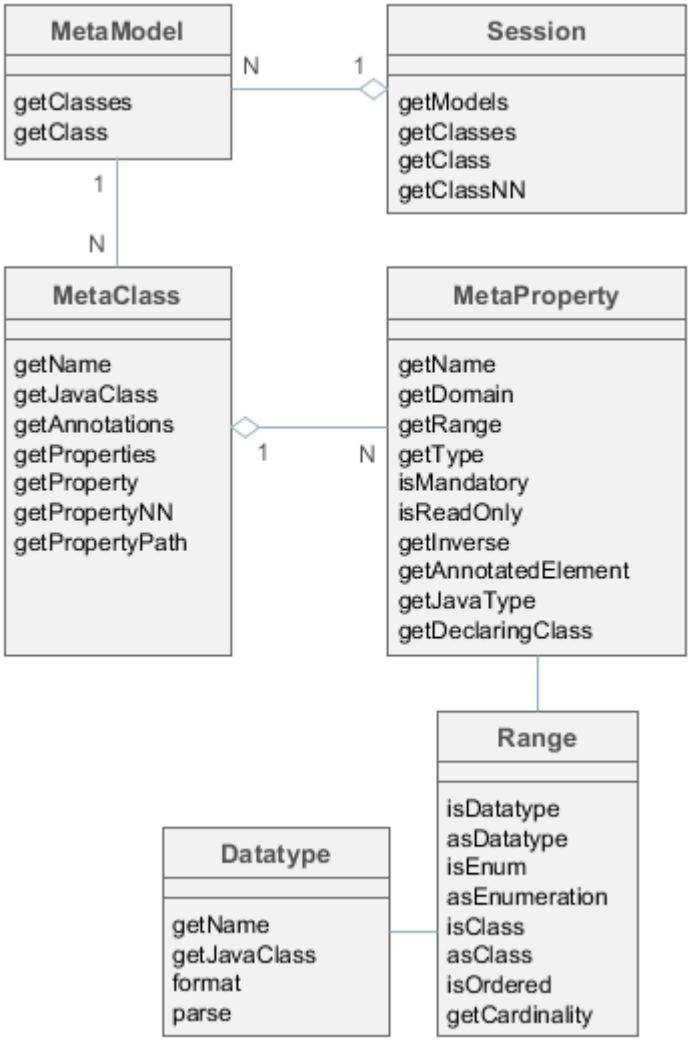


Рисунок 2.4 – Інтерфейси фреймворка метаданих

Session

Точка входу у фреймворк метаданих. Дозволяє отримувати екземпляри MetaClass за ім'ям і за відповідним класом Java. Зверніть увагу на відмінність методів getClass() і getClassNN() - перші можуть повертати null, другі ні (NonNull).

Об'єкт Session може бути отриманий через інтерфейс інфраструктури Metadata.

Приклад:

```
@Inject
protected Metadata metadata;
...
Session session = metadata.getSession();
MetaClass metaClass1 = session.getClassNN("sec$User");
MetaClass metaClass2 = session.getClassNN(User.class);
```

```
assert metaClass1 == metaClass2;
```

MetaModel

Рідко використовуваний інтерфейс, слугує для групування мета-класів.

Групування здійснюється за ім'ям кореневого Java пакета проєкту, що вказується у файлі metadata.xml.

MetaClass

Інтерфейс метаданих класу сутності. MetaClass завжди асоційований з класом Java, якого він представляє.

Основні методи:

getName() - ім'я сутності, за угодою першою частиною імені до знака _ є код простору імен, наприклад, sales_Customer

getProperties() - список мета-властивостей (MetaProperty)

getProperty(), getPropertyNN() - отримання мета-властивості за іменем. Перший метод у разі відсутності атрибута із зазначеним ім'ям повертає null, другий викидає виняток.

Приклад:

```
MetaClass userClass = session.getClassNN(User.class);
MetaProperty groupProperty = userClass.getPropertyNN("group");
```

getPropertyPath() - дає змогу переміщатися за посиланнями. Цей метод приймає строковий параметр - шлях з імен атрибутів, розділених крапкою. Об'єкт MetaPropertyPath, що повертається, дає змогу звернутися до шуканого (останнього в шляху) атрибута викликом getMetaProperty().

Приклад:

```
MetaClass userClass = session.getClassNN(User.class);
MetaProperty groupNameProp =
userClass.getPropertyPath("group.name").getMetaProperty();
assert groupNameProp.getDomain().getName().equals("sec$Group");
```

getJavaClass() - клас сутності, якому відповідає цей MetaClass

getAnnotations() - колекція мета-анотацій

MetaProperty

Інтерфейс метаданих атрибута сутності.

Основні методи:

getName() - ім'я властивості, відповідає імені атрибута сутності

getDomain() - мета-клас, якому належить ця властивість

getType() - тип властивості:

простий тип: DATATYPE

перерахування: ENUM

посилальний тип двох видів:

ASSOCIATION - просте посилання на іншу сутність. Наприклад, відношення замовлення і покупця - асоціація.

COMPOSITION - посилання на сутність, яка не має самостійного значення без сутності, що володіє нею. COMPOSITION можна вважати "тіснішим" відношенням, ніж ASSOCIATION. Наприклад, відношення замовлення і пункту цього замовлення - COMPOSITION, оскільки пункт не може існувати без замовлення, якому він належить.

Вид посилального атрибута ASSOCIATION або COMPOSITION впливає на режим редагування сутності: у першому випадку збереження пов'язаної сутності в базу даних відбувається незалежно, а в другому - пов'язана сутність зберігається в БД тільки разом із сутністю, що володіє.

getRange() - інтерфейс Range, що детально описує тип цього атрибута

isMandatory() - ознака обов'язковості атрибута. Використовується, наприклад, візуальними компонентами для сигналізації користувачеві про необхідність введення значення.

isReadOnly() - ознака незмінності атрибута

getInverse() - для посилального атрибута повертає мета-властивість зі зворотного боку асоціації, якщо така є

getAnnotatedElement() - поле (java.lang.reflect.Field) або метод (java.lang.reflect.Method), відповідні даному атрибуту сутності

getJavaType() - клас Java даного атрибута сутності. Це або тип поля класу, або тип значення методу, що повертається.

getDeclaringClass() - клас Java, що містить даний атрибут

Range

Інтерфейс, що детально описує тип атрибута сутності.

Основні методи:

isDatatype() - повертає true для атрибута простого типу

asDatatype() - повертає Datatype для атрибута простого типу

isEnum() - повертає true для атрибута типу перерахування

asEnumeration() - повертає Enumeration для атрибута типу перерахування

isClass() - повертає true для посилального атрибута типу ASSOCIATION або COMPOSITION

asClass() - повертає мета-клас асоційованої сутності для посилального атрибута

isOrdered() - повертає true якщо атрибут являє собою впорядковану колекцію (наприклад, List)

getCardinality() - вид відношення для посилального атрибута: ONE_TO_ONE, MANY_TO_ONE, ONE_TO_MANY, MANY_TO_MANY

Формування метаданих

Основне джерело формування структури метаданих - анотовані класи сутностей.

- Клас сутності відображається в метаданих у таких випадках:
- Клас персистентної сутності анотований @Entity, @Embeddable, @MappedSuperclass і розташований у межах кореневого пакета, зазначеного в metadata.xml.

Клас неперсистентної сутності анотований @MetaClass і розташований у межах кореневого пакета, зазначеного в metadata.xml.

Усі сутності всередині одного кореневого пакета поміщаються в один екземпляр MetaModel, якому присвоюється ім'я цього пакета. Між сутностями всередині однієї MetaModel можна встановлювати довільні зв'язки, між різними - у порядку оголошення файлів metadata.xml у властивості cuba.metadataConfig.

Атрибут сутності відображається в метаданих, якщо:

- поле класу анотовано @Column, @OneToOne, @OneToMany, @ManyToOne, @ManyToMany, @Embedded
- поле класу або метод доступу на читання (getter) анотований @MetaProperty

Параметри мета-класу і мета-властивостей формуються на основі параметрів перерахованих анотацій, а також типів полів і методів класу. Крім того, якщо в атрибута відсутній метод доступу на запис (setter), атрибут стає незмінним (read only).

Компоненти середнього шару

На наступному малюнку наведено основні компоненти середнього шару CUBA-додатку.

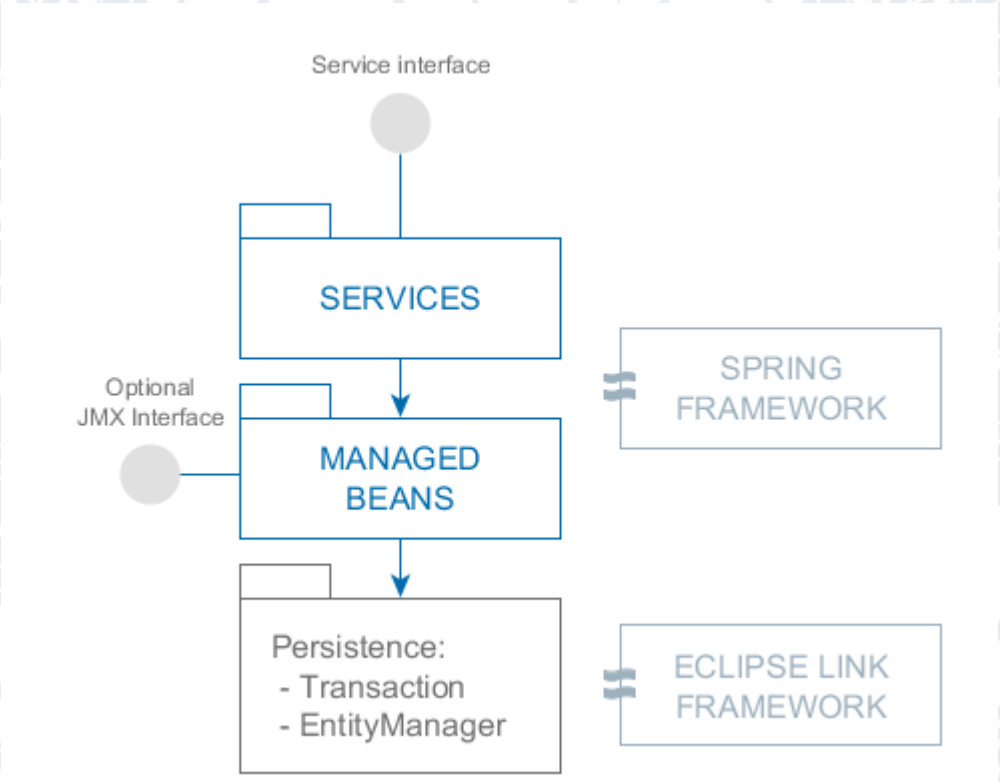


Рисунок 2.5 – Компоненти середнього шару

Services - Spring-біни, що формують межу застосунку і надають інтерфейс клієнтському рівню застосунку. Сервіси можуть містити бізнес-логіку самі, або делегувати виконання managed beans.

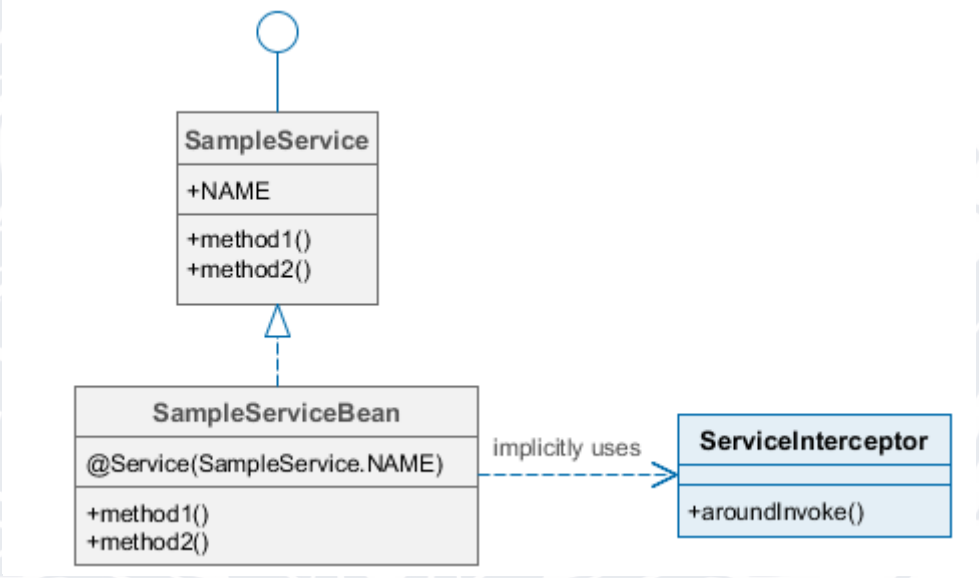
Managed Beans - Spring-біни, що містять бізнес-логіку додатка. Викликаються сервісами, іншими бінами або через опціональний JMX-інтерфейс.

Persistence - інфраструктурний інтерфейс для доступу до функціональності зберігання даних: управління транзакціями та ORM.

Сервіси

Сервіси утворюють шар компонентів, що визначає безліч операцій Middleware, доступних клієнтському рівню програми. Іншими словами, сервіс являє собою точку входу в бізнес-логіку середнього шару. У сервісі можна керувати транзакціями, перевіряти права користувачів, працювати з базою даних або делегувати виконання іншим Spring-бінам середнього шару.

Діаграма класів сервісу, що відображає основні компоненти сервісу:



Діаграма 2.3 – Діаграма класів сервісу

Інтерфейс сервісу розташовується в модулі global і доступний на клієнтському рівні та на Middleware. Під час виконання, на клієнтському рівні для інтерфейсу сервісу створюється проксі-об'єкт, який забезпечує виклики методів біна, що реалізує сервіс, за допомогою механізму Spring HTTP Invoker.

Бін, що реалізує сервіс, розташовується в модулі core і доступний тільки на середньому шарі.

ServiceInterceptor автоматично викликається для кожного методу сервісу через Spring AOP. Він перевіряє наявність у потоці виконання користувацької сесії, а також трансформує та логірує винятки, якщо сервіс викликано з клієнтського рівня.

Кращий спосіб роботи з даними в CUBA-додатках - використання сутностей: або декларативно у пов'язаних із даними візуальних компонентах, або програмно через DataManager чи EntityManager. Сутності відображаються на дані в сховищі, яке зазвичай являє собою реляційну БД. Додаток може працювати з кількома сховищами, тож його модель даних міститиме сутності, що відображаються на дані з різних БД.

Деяка сутність може належати тільки одному сховищу. Сутності з різних сховищ можна відображати на одному екрані UI, водночас DataManager забезпечує їхнє читання і запис у відповідне сховище. Залежно від типу сутності, DataManager вибирає зареєстроване сховище, представлене реалізацією інтерфейсу DataStore, і делегує йому завантаження або збереження. При управлінні транзакціями і роботі з сутностями через EntityManager необхідно явно вказувати, яке сховище використовувати.

Платформа містить єдину реалізацію інтерфейсу DataStore: RdbmsStore, призначену для роботи з реляційними СУБД через шар ORM. Крім того, у проєкті застосунку можна реалізувати DataStore для інтеграції, наприклад, із нереляційною СУБД або зовнішньою системою, що має REST інтерфейс.

Кожен CUBA-додаток має основне сховище, яке містить системні сутності та використовується для входу користувачів у додаток. У цьому посібнику під базою даних завжди мається на увазі основне сховище, якщо явно не обумовлено інше. Основне сховище має являти собою реляційну БД, підключену через джерело даних JDBC. Додаткове сховище може бути будь-якою реалізацією інтерфейсу DataStore.

Універсальний користувацький інтерфейс

Підсистема універсального користувацького інтерфейсу (Generic UI, GUI) дає змогу розробляти екрани UI, використовуючи Java і XML.

Використання XML не обов'язкове, але дає змогу описувати компонування екрана декларативно і знижує обсяг коду, необхідного для створення користувацького інтерфейсу.

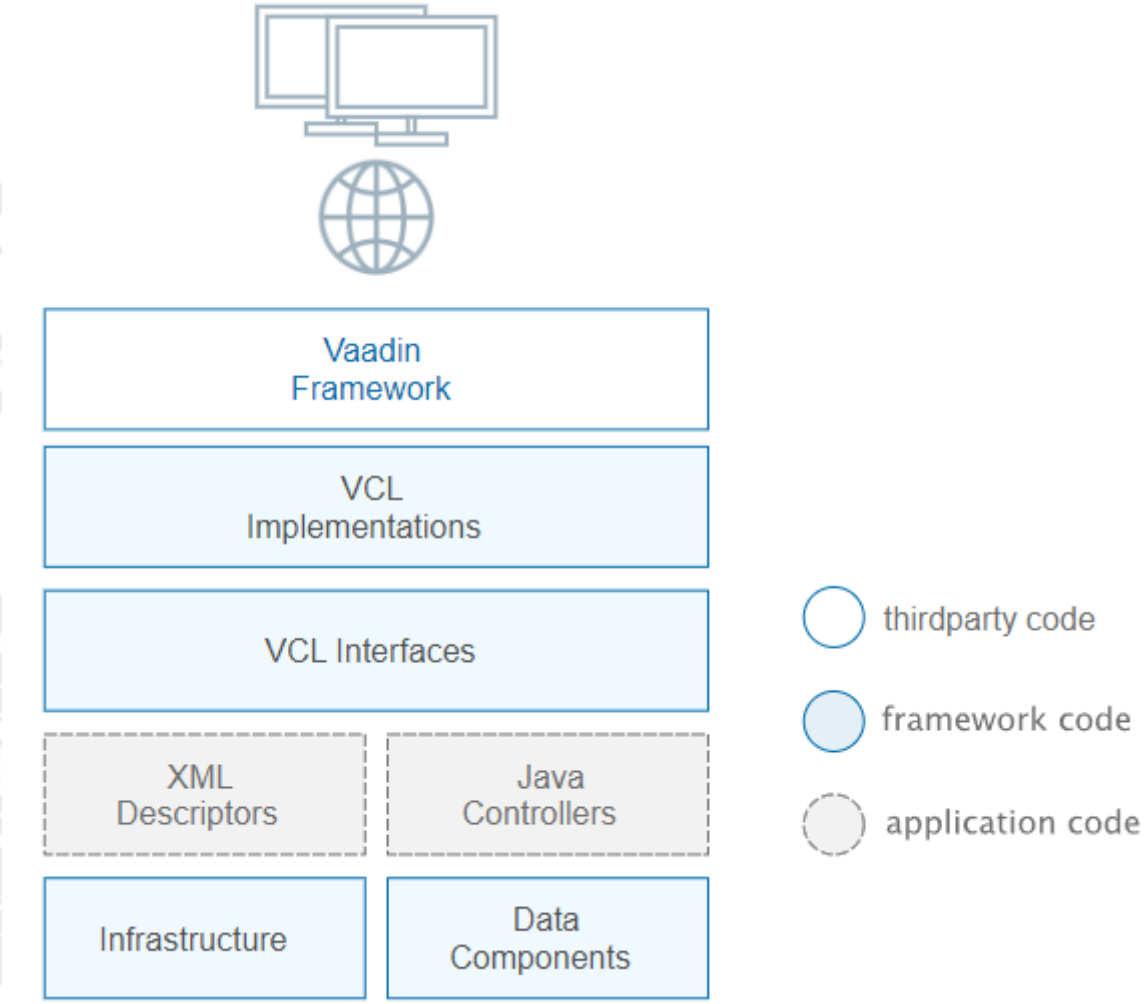


Рисунок 2.6 – Структура універсального користувацького інтерфейсу

Екрани програми складаються з таких частин:

- Дескриптори - XML-файли, що містять інформацію про компонування екрана і компоненти даних.
- Контролери - класи Java, що містять логіку ініціалізації та обробки подій від екрана і його компонентів.

Код екранів додатка взаємодіє з інтерфейсами візуальних компонентів (VCL Interfaces). Ці інтерфейси реалізовані з використанням компонентів фреймворку Vaadin.

Бібліотека візуальних компонентів (Visual Components Library, VCL) містить великий набір готових компонентів.

Компоненти даних (Data components) надають уніфікований інтерфейс для зв'язування візуальних компонентів із сутностями моделі даних і для роботи з сутностями в контролерах екранів.

Інфраструктура (Infrastructure) містить головне вікно програми та інші загальні клієнтські механізми.

Екрани

Екран - основний елемент користувацького інтерфейсу. Екран складається з візуальних компонентів, контейнерів даних та інших невізуальних компонентів. Екрани відображаються всередині головного вікна програми, в окремій його вкладці або у вигляді модального вікна.

Основу екрана становить Java- або Groovy-клас, який називається контролером. Компонування екрана зазвичай визначається в XML-файлі, іменованому дескриптором екрана.

Щоб відобразити екран, платформа створює новий екземпляр візуального компонента Window, під'єднує його до контролера екрана і завантажує компоненти, що входять до компонування екрана, як дочірні компоненти вікна. Потім вікно цього екрана додається до головного екрана програми.

Фрагмент - це ще один структурний елемент UI, який можна використовувати як компонент екранів або інших фрагментів. Внутрішній устрій фрагмента аналогічний екрану, але фрагмент має специфічний життєвий цикл, а в корені ієрархії візуальних компонентів знаходиться Fragment замість Window. Фрагменти, так само як екрани, складаються з контролерів і XML-дескрипторів.

Контролер екрану

Контролер екрана - це Java або Groovy клас, який містить у собі логіку ініціалізації екрана та обробки подій. Найчастіше контролер пов'язаний з XML-дескриптором екрана, який декларативно описує компонування екрана

та контейнери даних, однак і в контролері можна програмно створювати візуальні та невізуальні компоненти.

Усі контролери екранів реалізують інтерфейс-маркер FrameOwner. Назва цього інтерфейсу означає, що в ньому міститься посилання на фрейм, тобто візуальний компонент, що являє собою екран під час його відображення в головному вікні програми. Існує два типи фреймів:

- Window - самостійне вікно, яке можна відобразити всередині головного вікна програми у вкладці або у вигляді модального діалогового вікна.
- Fragment - легкий компонент, який можна додавати до вікон або до інших фрагментів.

Контролери також поділяються на дві окремі категорії за типами використовуваних фреймвів:

- Screen - базовий клас контролерів вікон.
- ScreenFragment - базовий клас контролерів фрагментів.

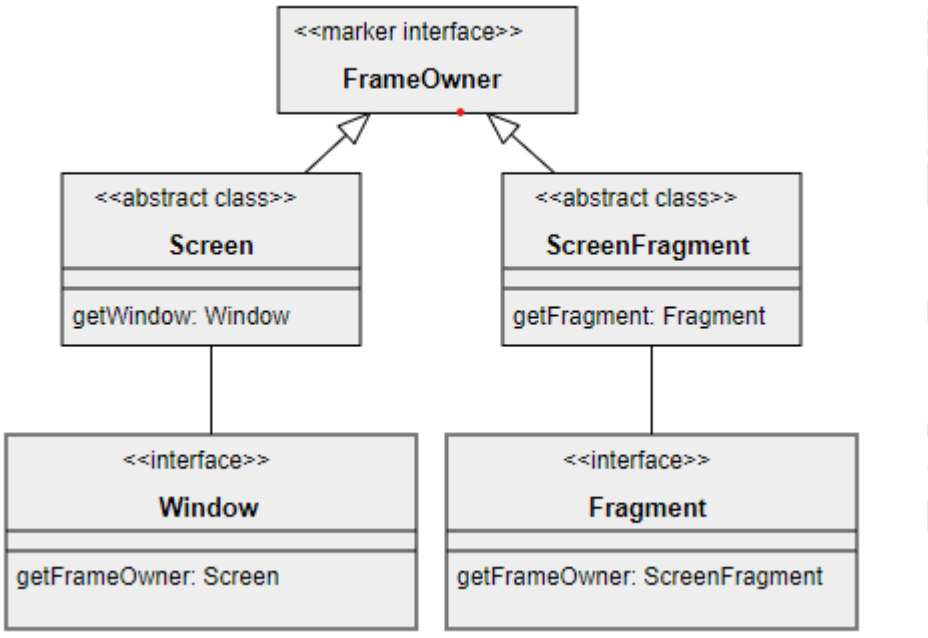


Рисунок 2.7 – Контролери та фрейми

Клас Screen надає базову функціональність для будь-якого типу самостійних екранів. Для екранів, призначених для роботи із сутностями, існують окремі, більш специфічні класи:

- StandardEditor - базовий клас контролерів екранів редагування.
- StandardLookup - базовий клас контролерів екранів перегляду і вибору.
- MasterDetailScreen - комбінований екран, що відображає список екземплярів сутностей зліва і деталі обраної сутності праворуч.

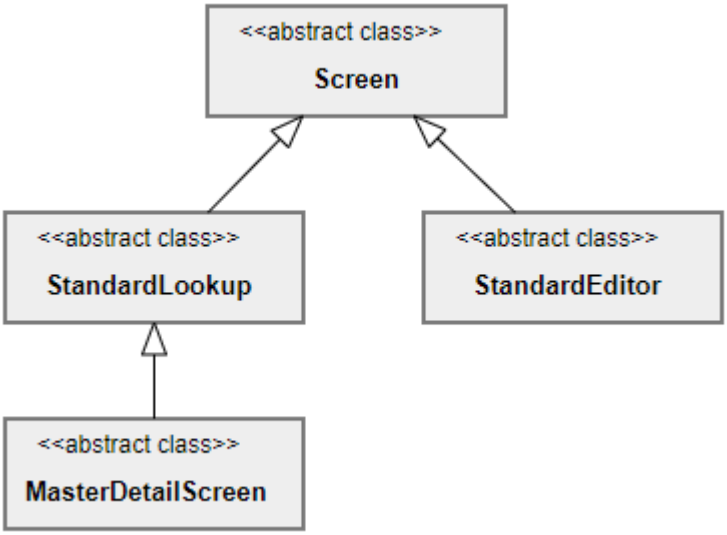


Рисунок 2.8 – Базові класи контролерів

2.4 Розробка архітектури додатку

Монолітний додаток - це додаток, що поставляється за допомогою єдиного розгортання. Завдяки цьому такий додаток буде поставлятися з виглядом єдиного WAR. Він також може поставлятися як Node-додаток з єдиною точкою входу.

Переваги монолітної архітектури додатків:

Великою перевагою моноліту є те, що його легше реалізувати. У монолітній архітектурі ви можете швидко почати реалізовувати свою бізнес-логіку замість того, щоб витрачати час на роздуми про взаємодію між процесами.

Інша справа - наскрізні (E2E) тести. Їх простіше виконувати в монолітній архітектурі.

Говорячи про виконувані операції, важливо сказати, що Monolith легко розгортається і легко масштабується. Для розгортання ви можете

використовувати скрипт, який завантажить ваш модуль і здійснить запуск додатку. Масштабування досягається шляхом розміщення Loadbalancer перед декількома екземплярами вашого додатку. Як бачите, моноліт досить простий в експлуатації.

Недоліки монолітної архітектури:

Моноліти мають тенденцію вироджуватися зі свого чистого стану в так званий "великий клубок бруду". Коротко це описується як стан, який виник через те, що були порушені архітектурні правила і компоненти з часом зрослися між собою.

Така дегенерація уповільнює процес розробки: кожну наступну функцію буде складніше розробити. Оскільки компоненти ростуть разом, їх також потрібно змінювати разом. Створення нової функції може означати дотик до 5 різних місць: 5 місць, де потрібно писати тести; 5 місць, які можуть мати небажані побічні ефекти на існуючі функції.

В моноліті все легко масштабується. Це відбувається до тих пір, поки він не перетвориться на "велику грудку бруду", як згадувалося раніше. Масштабування стає проблематичним, коли тільки для однієї частини системи потрібно додаткові ресурси, оскільки в монолітній архітектурі ви не можете масштабувати окремі частини вашої системи.

На жаль, монолітна архітектура майже не має ізоляції. Весь додаток може бути зруйнований або сповільнений однією проблемою або помилкою в модулі.

Побудова моноліту часто відбувається через вибір фреймворку. Складним може бути відключення або оновлення початкового вибору, оскільки це потрібно робити відразу і для всіх частин системи.

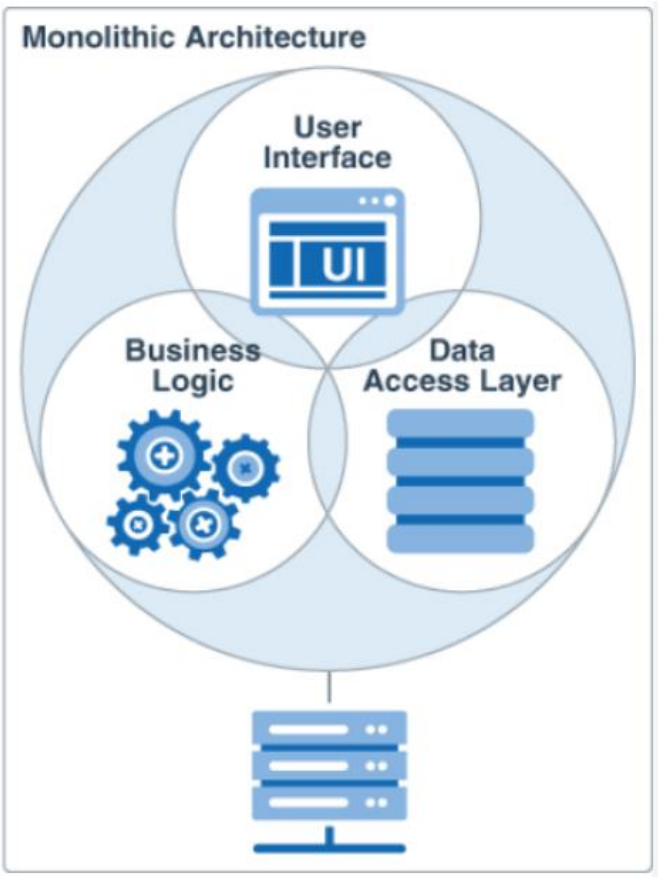


Рисунок 3.1– приклад монолітної архітектури.

Висновки до розділу 2.

У другому розділі були розглянуті переваги правильної архітектури. Більш детально була описана база даних PostgreSQL, а також те, що було використано при написанні бота. Була розглянута та обрана монолітна архітектура для створення бота.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ПІДБОРУ ЛІКАРСЬКИХ ПРЕПАРАТІВ НА ОСНОВІ TELEGRAM - БОТУ

3.1 Огляд Telegram API

Telegram API - це простір для розробки пошукових ботів. Це середовище містить вбудовані інструменти та сервер, на якому ці боти зберігаються. Кожен бот має свою адресу, де він знаходиться і звідти взаємодіє з людьми. Телеграм-бот - це програма, яка може виконувати певні дії за людину. Насправді існує багато способів використання ботів, але здебільшого це пошук якоїсь інформації або, наприклад, імітація живого спілкування.

Telegram Bot API дозволяє створювати ботів. Боти - це спеціальні акаунти, які не потребують телефонного номера для реєстрації. Код бота має бути бути запущеним на стороні людини (ПК або сервері) - Telegram служить лише інтерфейсом для отримання і відправки повідомлень.

Для використання API не потрібно володіти якимись специфічними знаннями низькорівневих протоколів Telegram - всі запити здійснюються за протоколом HTTP. Його можна відправити навіть вручну, за допомогою браузера.

Bot API - це HTTP-інтерфейс для взаємодії з ботами в додатку Telegram. Кожен бот - це створений людиною акаунт, який автоматично обробляє та надсилає повідомлення.

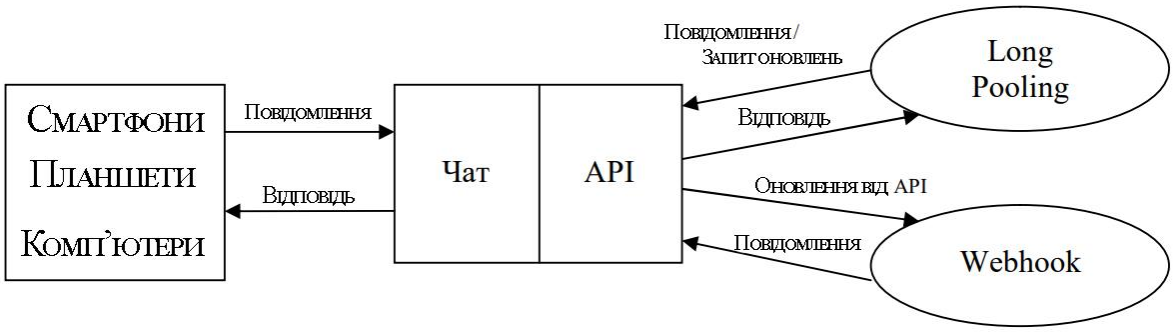


Рисунок 2.2 – Принцип роботи чат-бота на платформі Telegram

При створенні бота йому присвоюється унікальний токен виду 234567: ABD-EFG1434ghIkl-zyx57S2v1u123ew11. Замість бота в документації буде використовуватися <token> для більш легкого сприйняття.

Допускаються такі GET і POST запити. Існує 4 доступних способів передачі параметрів до Bot API [40]:

- Запит в URL
- application/x-www-form-urlencoded
- application/json (не підходить для завантаження файлів)
- multipart / form-data (для завантаження файлів)

Відповідь буде мати вигляд JSON-об'єкта, який буде містити логічне boolean поле "ok" та необов'язкове строкове поле description, яке містить зрозумілий для людини опис результату. Запит пройшов успішно і результат можна побачити в полі result тільки в тому випадку, якщо поле "ok" має значення true. Якщо допущена помилка, то поле "ok" буде мати позначку «false», а помилка буде описана в полі descriptrion. Крім того, відповідь буде містити цілочисельне поле error_code, але в подальшому коди помилок будуть змінюватися.

Якщо бот працює через веб-хуки, можна зробити запит до API бота одночасно з відправкою відповіді на веб-хук. Для передачі параметрів потрібно використовувати один з типів:

- application / json
- application / x-www-form-urlencoded
- multipart / form-data

Метод, який буде буде застосований для виклику, повинен бути визначений в запиті в полі method. Однак дізнатися статус і результат виконання такого запиту неможливо.

Існує два способи отримання оновлень від бота, які є діаметрально протилежними за логікою: getUpdates та webhooks. Максимальний час зберігання вхідних оновлень на сервері буде до моменту їх обробки, але не більше 24 годин.

Незалежно від того, в який спосіб буде отримане оновлення, відповіддю буде об'єкт Update, серіалізований у форматі JSON.

Об'єкт Update являє собою вхідне оновлення. Під оновленням ми маємо на увазі дію, яка виконується над ботом – це може бути, наприклад, отримання повідомлення від користувача.

У кожному оновленні може бути присутній тільки один з необов'язкових параметрів.

Щоб отримати оновлення за допомогою long polling використайте метод GetUpdates. Відповідь повернеться у вигляді масиву Update.

Метод setWebhook необхідний для встановлення URL-адреси веб-хука, на який бот буде відправляти оновлення. Кожного разу при отриманні оновлення на цю адресу буде відправлятися HTTPS POST з серіалізованим в JSON об'єктом Update. Якщо запит до вашого сервера виявиться невдалим, спроба буде повторена помірну кількість разів [24].

Таблиця 3.1 – Параметри setWebhook [24]

Параметри	Тип
url	String
Sertificate	InputFile

- Url - HTTPS url для відправки запитів;
- Sertificate – завантаження публічного ключа для того щоб перевірити кореневий сертифікат.

Метод getWebhookInfo містить інформацію про поточний стан веб-хука. Нижче, в таблиці 3.2, наведені параметри цього метода[24].

Таблиця 3.2 – Параметри які повертає getWebhookInfo [24]

Параметри	Тип
url	String

Продовження таблиці 3.2

Параметри	Тип
has_custom_certificate	Boolean
pending_update_count	Integer
last_error_date	Integer
last_error_mess age	String

- url - url веб-хука, може бути порожнім;
- has_custom_cer tificate – якщо поле “True”. Тільки в тому випадку, коли веб-хук використовує свій самозавірений сертифікат;
- pending_update _count – у цьому полі продемонстрована кількість оновлень, що очікують своєї черги;
- last_error_date – це поле необов’язковим поле. Unix-час останньої помилки доставки оновлення на вказаний веб-хук;
- last_error_mess age - це поле необов’язкове поле. В ньому відбувається опис останньої помилки доставки оновлення на вказаний веб-хук.

Метод sendMessage застосовується для відправки повідомлень. В таблиці 3.3 наведені параметри методу [24].

Таблиця 3.3 – Параметри sendMessage [24]

Параметри	Тип	Обов'язковий
chat_id	Integer або String	Так
Text	String	Так
parse_mode	String	Ні
disable_web_pag e_ preview	Boolean	Ні

Продовження таблиці 3.3

Параметри	Тип	Обов'язковий
disable_notificati on	Boolean	Hi
replay_to message_id	nteger	Hi
replay_markup	InlineKeybo ard Markup або ReplyKeybo ard Markup або ReplyKe yboard Hide	Hi

- chat_id – це унікальний ідентифікатор основного чату або імені основного каналу;
- Text - текст повідомлення, яке нам потрібно відправити;
- parse_mode - потрібно надіслати Markdown або HTML, щоб додатки Telegram відображали різний за стилем текст, такий як: напівжирний, курсивний, текст з фіксованою шириною або вбудовані URL-адреси в повідомленні бота;
- parse_mode - потрібно відправити Markdown або HTML, щоб додатки Telegram відображали в повідомленні бота різні стилі тексту, наприклад, напівжирний, курсив, текст фіксованої ширини або вбудовані URL-адреси;
- disable_web_page_ preview – поле, відключає попередній перегляд посилань е повідомленні;
- disable_notification – це поле відправляє повідомлення у тихому режимі.
- Користувачі iOS не зможуть отримати повідомлення, тим часом як користувачі Android отримають повідомлення, але воно буде без звуку;
- replay_to message_id – це поле містить інформацію про ідентифікатор вихідного повідомлення;

- replay_markup – це поле дозволяє розширити можливості пошуку інтерфейсу та являє собою JSON-серіалізований об'єкт для вбудованої клавіатури, яка призначена для користувача.

Метод sendPhoto використовується для відправлення фотографій. У таблиці 3.4 наведені параметри методу [24].

Таблиця 3.4 – Параметри sendPhoto [24]

Параметри	Тип	Обов'язковий
chat_id	Integer або String	Так
Photo	InputFile або String	Так
Caption	String	Ні
Replay_markup	InlineKeyboard Markup or ReplyKeyboard Markup or ReplyKeyboard Hide or ForcекReply	Ні
replay_to message_id	Integer	Ні

- chat_id – це поле містить інформацію про унікальний ідентифікатор основного чату або назви основного каналу;
- Photo – це поле, що містить інформацію про фото, яке ми маємо відправити. Можна передати file_id у вигляді рядку, щоб відправити фотографію яка вже знаходиться на серверах Telegram, або завантажити нову фотографію;
- Caption - в даному полі міститься інформація про тему фотографії, функція дозволяє написати від 0 до 200 символів;
- • Replay_markup – це поле містить інформацію про функції пошуку інтерфейсу розширення. Це серіалізований об’єкт JSON, що

представляє вбудовану клавіатуру користувача, який використовується, щоб приховати клавіатуру користувача або змусити користувача відповідати на певні команди;

- - replay_to message_id - це поле містить інформацію про ідентифікатор вихідного повідомлення.

Метод editMessageText використовується з метою редагування текстових повідомлень, які були відправлені ботом або через бота. Параметри методу наведені в таблиці 3.5. [24].

Таблиця 3.5 – Параметри методу editMessageText [24]

Параметри	Тип	Обов'язковий
chat_id	Integer або String	Ні
inline_message_id	String	Ні
Text	String	Так
parse_mode	String	Ні
disable_web_page_ preview	Boolean	Ні
replay_markup	InlineKeyboard Markup або ReplyKeyboard Markup або ReplyKeyboard Hide	Ні

- chat_id – є необхідним параметром, при умові, якщо inline_message_id не вказано. Він являється унікальним ідентифікатором цільового чату або імені каналу;
- inline_message_id – це поле є обов'язковим за умови, якщо chat_id і message_id не вказуються. Ідентифікатор вбудованого повідомлення;
- Text – у цьому полі міститься інформація про новий текст повідомлення;

- parse_mode – потрібно надіслати Markdown або HTML, щоб додатки Telegram змогли відображати різний за стилем текст, такий як: напівжирний, курсивний, текст з фіксованою шириною або вбудовані URL-адреси в повідомленні нашого бота;
- disable_web_page_preview - відключає попередній перегляд посилань для посилань , які знаходяться у даному повідомленні;
- replay_markup – у цьому полі міститься інформація про розширення можливостей пошуку інтерфейсу;

Об’єкт типу User надає інформацію про користувача Telegram. Поля типу наведені в таблиці 3.6 [24].

Таблиця 3.6 – Поля об’єкта User [24]

Поле	Тип
Id	Integer
first_name	String
last_name	String
Username	String

- Id – це поле містить інформацію про унікальний ідентифікатор бота або користувача;
- first_name – у цьому полі міститься інформаці про ім'я бота або користувача;
- last_name – це поле необов’язкове, воно містить інформацію про прізвище бота чи користувача;
- Username – це поле також є не обов’язковим, воно містить інформацію про Username бота або користувача;

Chat – це об’єкт, що являє собою інформацію про чат. Поля цього типу наведені нижче в таблиці 3.7 [24].

Таблиця 3.7 – Поля об’єкта Chat [24]

Поле	Тип
Id	Integer
Type	Enum
Title	String
Username	String
first_name	String
last_name	String
all_members_are_adminis trators	Boolean

- Id – у цьому полі міститься інформація про унікальний ідентифікатор чату. Абсолютне значення не перевищує 1e13;
- Type - у цьому полі міститься інформація про тип чату: а) private; б) group; в) supergroup; г) channel;
- Title - це поле необов’язкове, воно містить інформацію про назву для каналів або груп;
- Username - це поле необов’язкове, воно містить інформацію про Username, для чатів і деяких каналів;
- first_name - це поле необов’язкове, воно містить інформацію про Username, для чатів і деяких каналів;
- last_name - це поле необов’язкове, воно містить інформацію про Прізвище співрозмовника в чаті;
- all_members_are_adminis trators - це поле необов’язкове. Поле «True», якщо адміністраторами є всі учасники чату.

Об’єкт типу Message представляє собою інформацію про дане повідомлення. Поля даного типу наведені в таблиці 3.8 [24].

Таблиця 3.8

Поле	Тип
message_id	Integer
From	User
Date	Integer
Chat	Chat
forward_from	User
forward_date	Integer
reply_to_message	Message
Text	String
Entities	Масив з MessageEntity
Document	Document
Photo	масив з PhotoSize
Sticker	Sticker
Video	Video
Voice	Voice
Caption	String
Contact	Contact
Location	Location
Venue	Venue
new_chat_member	User
left_chat_member	User
new_chat_title	String
new_chat_photo	масив з PhotoSize
group_chat_created	True

Продовження таблиці 3.8

Поле	Тип
supergroup_chat_created	True
channel_chat_created	True
migrate_to_chat_id	Integer
migrate_from_chat_id	Integer
pinned_message	Message

- message_id – у цьому полі міститься інформація про унікальний ідентифікатор повідомлення;
- From - це поле необов’язкове, воно містить інформацію про відправника. Може бути порожнім в каналах;
- Date - у цьому полі міститься інформація про дату відправлення повідомлення (Unix time);
- Chat - у цьому полі міститься інформація про діалог, в якому було відправлено повідомлення;
- forward_from - це поле не є обов’язкове. Для пересланих повідомлень: відправник оригінального повідомлення;
- forward_date – це поле не є обов’язкове. Для пересланих повідомлень: дата відправки повідомлення;
- reply_to_message – це поле не є обов’язкове. Використовується для відповідей -оригінальне повідомлення. Слід звернути увагу на те, що об'єкт Message не буде містити в цьому полі додаткові поля reply_to_message, навіть якщо він сам являється відповіддю;
- Text - це поле не є обов’язкове, воно використовується для текстових повідомлень-текст повідомлення, 0- 4096 символів;
- Entities - це поле не є обов’язкове, використовується для текстових повідомлень-особливі суті в тексті повідомлення;
- Document - це поле необов’язкове, воно містить інформацію про фото;

- Photo - це поле необов’язкове, воно містить інформацію про доступні розміри фото;
- Sticker - це поле необов’язкове, воно містить інформацію про стікери;
- Video - це поле необов’язкове, воно містить інформацію про відеозапис;
- Voice - це поле необов’язкове, воно містить інформацію про голосове повідомлення;
- Caption - це поле необов’язкове, воно містить інформацію про підпис до файлу, фото або відео, не має перевищувати 200 символів;
- Contact - це поле необов’язкове, воно містить інформацію про відправлений контакт;
- Location - це поле необов’язкове, воно містить інформацію про місцезнаходження;
- Venue - це поле необов’язкове, воно містить інформацію про місце на карті;
- new_chat_member - це поле необов’язкове, воно містить інформацію про користувача, доданого до групи;
- left_chat_member - це поле необов’язкове, воно містить інформацію про користувача, якого видалили;
- new_chat_title - це поле необов’язкове, воно містить інформацію на це поле;
- new_chat_photo - це поле необов’язкове. Фото групи було змінено на це поле;
- group_chat_created - це поле необов’язкове. Сервісне повідомлення наступне: створена група;
- supergroup_chat_created - це поле необов’язкове. Сервісне повідомлення наступне: створена супергрупа;
- channel_chat_created - це поле необов’язкове. Сервісне повідомлення наступне: створений канал;

- migrate_to_chat_id - це поле необов’язкове. Група була перетворена в супергрупу із зазначеним ідентифікатором. Не перевищує 1e13;
- migrate_from_chat_id - це поле необов’язкове. Супергрупа була створена з групи із зазначеним ідентифікатором. Не перевищує 1e13;
- pinned_message - це поле необов’язкове. Зазначене повідомлення було прикріплене. Об'єкт Message в цьому полі не буде містити додаткові поля reply_to_message, навіть якщо він сам є відповіддю;

Ще однією кнопкою відповіді є об'єкт типу KeyboardButton на клавіатурі**.** Цей об'єкт може бути змінений на рядок, що містить текст на кнопці для звичайних текстових кнопок. Поля цього типу наведено у таблиці 3.9 [24].

Таблиця 3.9 – Поля об’єкта KeyboardButton [24]

Поле	Тип
Text	String
request_contact	Boolean
request_location	Boolean

- Text – містить інформацію про текст, який має набирати користувач, на кнопці. При натисканні на кнопку цей текст буде відправлений боту як просте повідомлення при умові, якщо жодне з опціональних полів не використано;
- request_contact – це поле необов’язкове. Якщо значення «True», то , коли натискати на кнопку, боту відправиться контакт користувача та його номер телефону. Ця функція доступна тільки в діалогах з ботом;
- request_location - це поле необов’язкове. Якщо значення «True», то, коли натискати на кнопку, боту відправиться місце локації користувача. Ця функція доступна тільки в діалогах з ботом;

Об’єкт типу InlineKeyboardMarkup – це є вбудована клавіатура, що з’являється під певним повідомленням. Поля цього типу наведені в таблиці 2.10 [24].

Таблиця 3.10 – Поля об’єкта InlineKeyboardMarkup [24]

Поле	Тип
inline_keyboard	Масив масивів з InlineKeyboardButton

- inline_keyboard - Масив рядків, кожна з яких є масивом об'єктів InlineKeyboardButton.

Об’єкт типу InlineKeyboardMarkup це ще одна кнопка вбудованої клавіатури. Ми обов’язково повинні задіяти рівно одне опціональне поле. Поля цього типу наведені в таблиці 2.11 [24].

Таблиця 3.11 – Поля об’єкта InlineKeyboardButton [24]

Поле	Тип
Text	String
url	String
callback_data	String
switch_inline_query	String
switch_inline_query_current_chat	String
callback_game	CallbackGame

- Text – містить інформацію про текст на кнопці;
- url – це поле не є обов'язковим для заповнення, воно містить інформацію про URL, який буде відкриватися при натисканні на кнопку

- callback_data - це поле необов’язкове. Дані, які відправляються в callback_query при умові натискання на кнопку
- switch_inline_query - це поле необов’язкове. Якщо задати цей параметр, то при натисненні на кнопку додаток буде пропонувати користувачеві вибрати якийсь чат, відкрити його і в поле введення вставити Ім'я користувача (Username) бота і конкретний запит для вбудованого режиму. Якщо ж відправити порожнє поле, то буде вставлений тільки Username бота.
- switch_inline_query_current_chat - це поле необов’язкове. При умові , коли воно встановлене, натисканням кнопки введемо ім'я бота і вказаний вбудований запит в поле вводу поточного чату. Воно може бути пустим.У цьому випадку вставиться тільки ім'я користувача бота
- callback_game - це поле необов’язкове, воно містить інформацію про опис гри, яка буде запускатися під час натискання кнопки користувачем.

3.2 Розробка бази даних

Проектування бази даних реалізується на основі реляційної моделі. В результаті цієї реалізації була отримана наступна база даних (див. додаток А)

В таблиці 3.12 наведені основні поля таблиці бази даних, що призначена для зберігання інформації щодо вибору користувача.

Таблиця 3.12- tabletbot_choise_user

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису

Продовження таблиці 3.12

deleted_by	character varying(2048)	Той хто видалив запис
choise	character varying(50)	Вибір користувача

В таблиці 3.13 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про виробника препаратів.

Таблиця 3.13– tabletbot_manufacture

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(2048)	Назва

В таблиці 3.14 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про форму випуску препаратів.

Таблиця 3.14– tabletbot_release_form

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє

Продовження таблиці 3.14

delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(1024)	Назва

В таблиці 3.15 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про країну виробника препаратів.

Таблиця 3.15– tabletbot_country

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(2048)	Назва

В таблиці 3.16 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про фаракотерапевтичну групу.

Таблиця 3.16– tabletbot_pharmacotherapeutic_group

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис

update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(5096)	Назва

В таблиці 3.17 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про тип препаратів.

Таблиця 3.17– tabletbot_tablet

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис

В таблиці 3.18 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про виготовлення препаратів.

Таблиця 3.18– tabletbot_drug_type

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису

updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(2048)	Назва

В таблиці 3.19 наведені основні поля таблиці бази даних, що призначена для зберігання інформації про країну виробника препаратів.

Таблиця 3.19– tabletbot_expiration_date_type

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
name	character varying(2048)	Назва

В таблиці 3.20 наведені основні поля таблиці бази даних, що призначена для збереження реляційного відношення.

Таблиця 3.20– tabletbot_preparation_prepareation_link

Назва	Тип	Опис
preparation_1_id	uuid	Унікальний ідентифікатор
preparation_2_id	uuid	Унікальний ідентифікатор

В таблиці 3.21 наведені основні поля таблиці бази даних, що призначена для зберігання інформації щодо повного описання препарату.

Таблиця 3.21– tabletbot_preparation

Назва	Тип	Опис
id	uuid	Унікальний ідентифікатор
version	integer	Версія
Назва	Тип	Опис
create_ts	timestamp	Початок створення запису
created_by	character varying(50)	Той хто створив запис
update_ts	timestamp	Початок оновлення запису
updated_by	character varying(50)	Той хто редагував востаннє
delete_ts	timestamp	Початок видалення запису
deleted_by	character varying(50)	Той хто видалив запис
id_driz	character varying(64)	Унікальний ідентифікатор у державному реєстрі
trade_name	character varying(2048)	Торгова назва
vacation_conditions	character varying(2048)	Умови відпустки
drug_composition	character varying(5096)	Лікарський склад
manufacture_id	uuid	Унікальний ідентифікатор

country_id	uuid	Унікальний ідентифікатор
address_manufacture	character varying(2048)	Адреса виробництва
Назва	Тип	Опис
registration_certificate_number	character varying(1024)	Номер свідоцтва про реєстрацію
drug_type_id	uuid	Унікальний ідентифікатор
biological_origin	boolean	Чи являється біологічного походження
plant_origin	boolean	Чи являється рослинного походження
orphan	boolean	Чи це препарат сирота
homeopathic	boolean	Чи це гомеопатичний препарат
instruction_date	character varying(2048)	Дата інструкції
expiration_date	integer	Термін придатності
expiration_date_type_id	uuid	Унікальний ідентифікатор
realese_form_id	uuid	Унікальний ідентифікатор
pharmacotherapeutic_group_id	uuid	Унікальний ідентифікатор

3.3 Результат роботи бота

В даному розділі показано результат роботи бота. Більш детально буде показано на скріншотах:

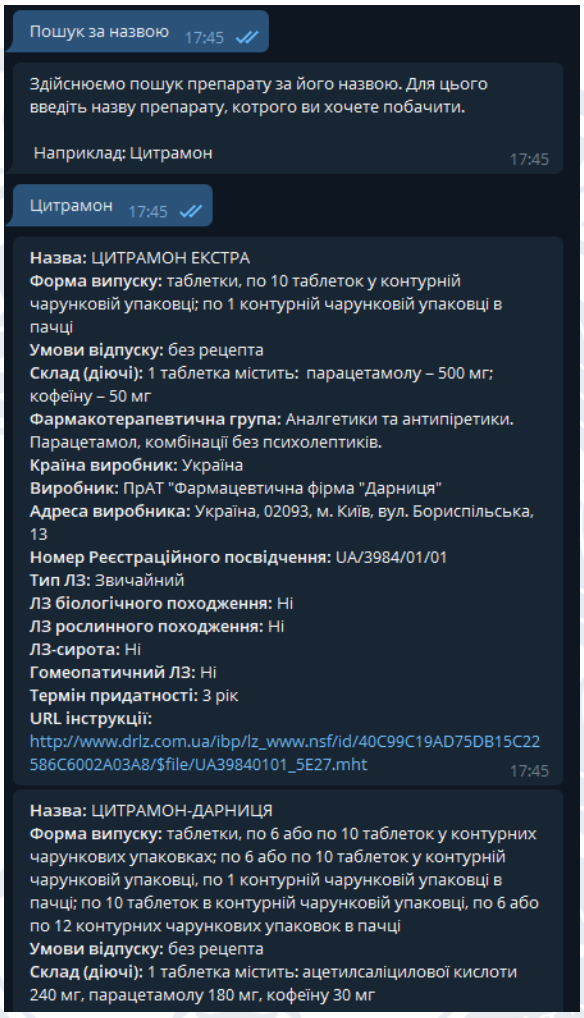


Рисунок 3.2 – Принцип пошуку за назвою препарату

Скріншот (рис. 3.2) показує, як реалізується принцип пошуку препарату саме за його назвою. На рисунку видно всі необхідні дані про препарат , а саме: назва препарату, форма випуску, умови відпуску. склад, фармакотерапевтична група, країна виробник, виробник, адреса виробника, номер реєстраційного посвідчення, тип лікарського засобу, інформація про те, чи є даний лікарський засіб біологічного чи рослинного походження, чи являється сиротою, чи є він гомеопатичним препаратом. Також вказується термін придатності. Тут же є в наявності і інструкція.

Бот показує 5 прикладів препаратів, які мають однакову назву, але різний склад, якщо кількість таких препаратів більше 5, є можливість натиснути кнопку "Наступні 5 препаратів" (рис.3.3)

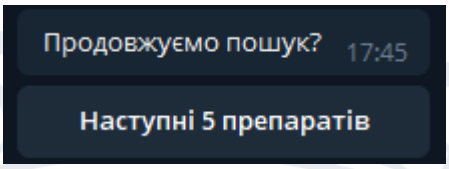


Рисунок 3.3 – кнопка «Наступні 5 препаратів»

Telegram бот може підбирати лікарський засіб за діючою речовиною. Цей спосіб пошуку продемонстрований на рисунку 3.4.

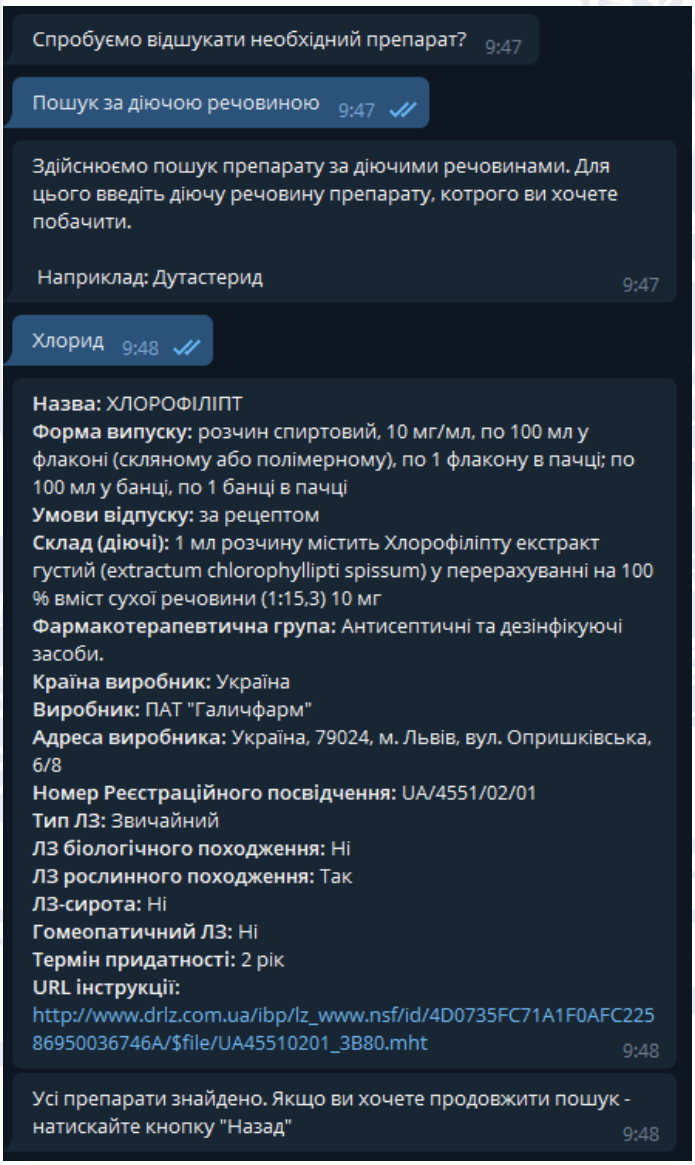


Рисунок 3.4 – Процес пошуку препарату на основі діючих речовин

В даному боті реалізовано метод пошуку аналогів лікарських препаратів. Якщо користувач хоче придбати певні ліки, він має можливість побачити, які препарати схожі за складом, але відрізняються за своєю назвою. Приклад наведено на рисунку 3.6.

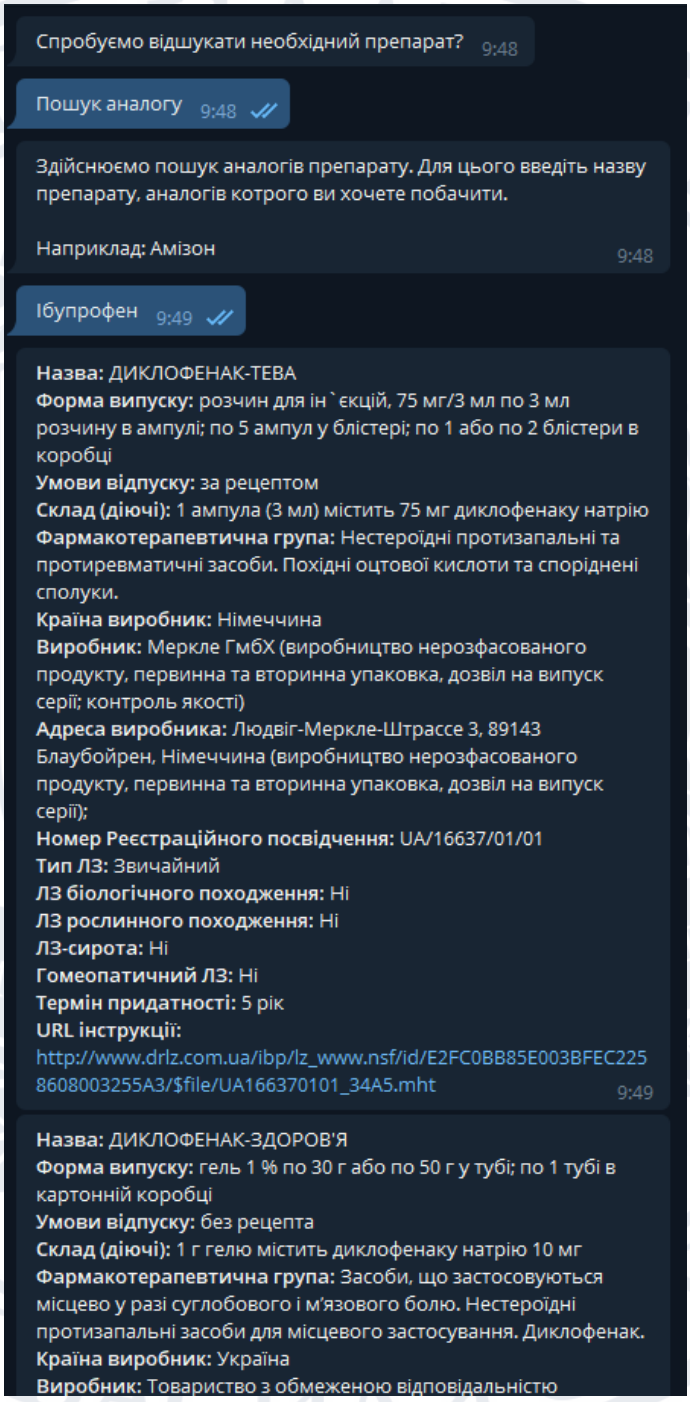


Рисунок 3.6 – Пошук аналога препарата

Якщо користувач не розуміє, як користуватися ботом, то потрібно ввести команду /help, і йому буде продемонстровано, як шукати препарат за його назвою.

Всі інші функції побудовані за таким же принципом, натискаємо кнопку "пошук" і вводимо те, що потрібно. Це продемонстровано на рисунку 3.7.

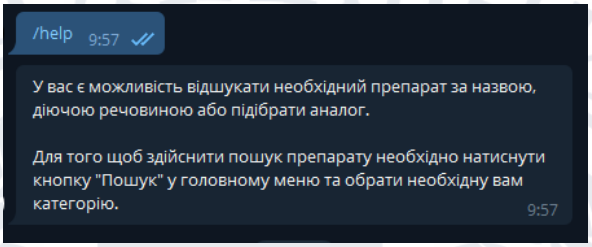


Рисунок 3.7– Команда допомоги

Висновки до розділу 3.

У третьому розділі була розглянута і детально описана монолітна архітектура. Більш детально була описана база даних. Продемонстровано результат та принцип роботи бота. Також було детально описано принцип роботи з Telegram API та використання його для створення бота.

ВИСНОВОК

В ході виконання магістерської роботи було досліджено особливості пошуку медичних препаратів та їх аналогів. Проведено аналізування переваг та недолік аналогів телеграм боту. Здійснено огляд інструментів які використовуються при розробці телеграм ботів.

Використовуючи отримані результати здійснено побудову комп'ютерно математичної моделі. Для вирішення задачі пошуку препаратів було сформовано локальну базу даних, яка базується на відкритих джерелах.

Вивчено інфраструктура Cuba Platform, бібліотека EclipseLink, база даних PostgreSQL, REST архітектура.

Реалізовано веб-додаток на основі REST архітектури, що здійснює взаємодію із Telegram API. Також сам Telegram бот має зручний користувацький інтерфейс.

Завдання було виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Важливість Telegram бота [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/WnqAu5B - Дата звернення 20.05.2021
2. Сфери використання Telegram бота [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/TnqASfc - Дата звернення 20.05.2021
3. Особливості Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/wnt2kt3 - Дата звернення 25.05.2021
4. Python [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/dnqY2c0 - Дата звернення 24.05.2021
5. Про мову Python [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/8nqUO21 - Дата звернення 24.05.2021
6. Короткі відомості про мову C# [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/6nqPI0z - Дата звернення 24.05.2021
7. Рейтинг мов програмування [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/xnqDBPt - Дата звернення 24.05.2021
8. Мова програмування Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/bnqDQc1 - Дата звернення 24.05.2021
9. Відомості про Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/xnqGnpk - Дата звернення 24.05.2021
10. Відомості про фреймворк Cuba Platform [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/6nqKJpx - Дата звернення 24.05.2021
11. Екосистема Cuba Platform [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/enqZRY3 - Дата звернення 24.05.2021
12. Засоби розробки Cuba Platform [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/8nqZz5M - Дата звернення 24.05.2021
13. Набір функцій Cuba Platform [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/lnqCxra - Дата звернення 24.05.2021

14. База даних PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/snqMw1A - Дата звернення 24.05.2021
15. Функції PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/fnqM2OW - Дата звернення 24.05.2021
16. Детально про Telegram API [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/nnwjYCS - Дата звернення 24.05.2021
17. Плюси Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/XnwFhNb - Дата звернення 25.05.2021
18. Плюси PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/IneE9AJ - Дата звернення 25.05.2021
19. Монолітна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/ZnabWja - Дата звернення 27.05.2021
20. Платформа CUBA [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/LnaIoZz - Дата звернення 27.05.2021
21. Роберт Мартин. Чистый код. Создание, анализ и рефакторинг. Перевод на русский язык ООО Издательство «Питер», 2015. 464 с.
22. Саймон Ригс, Ханну Кросинг. Администрирование PostgreSQL 9. Книга рецептов. Оформление, перевод на русский язык ДМК Пресс, 2013. 366 с.
23. Ханс-Юрген Шониг. Mastering PostgreSQL 12: Advanced Techniques to Build and Administer Scalable and Reliable PostgreSQL Database Applications, 3rd Edition. Published by Packt Publishing Ltd. 2018. 443 с.
24. Telegram API [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/tnwfkGh - Дата звернення 24.05.2021
25. Nicolas Modrzyk. Building Telegram Bots. Издатель Apress, 2018. 277 с.
26. Офіційний сайт Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/9ndylKf - Дата звернення 27.05.2021
27. Типи даних Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/2ndyQDF - Дата звернення 27.05.2021

28. Плюси Cuba Platform (Jmix) [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/fndyVDo - Дата звернення 27.05.2021
29. Telegram API [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/undy18R - Дата звернення 27.05.2021
30. Telegram APIs [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/ondy5JE - Дата звернення 27.05.2021
31. Telegram API Methods [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/gndueIg - Дата звернення 27.05.2021
32. Telegram API Bots [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/HnduHJE - Дата звернення 27.05.2021
33. Інструкція по Bot API [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/9ndylKf - Дата звернення 27.05.2021
34. Telegram бот на Java [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/8nduX8j - Дата звернення 27.05.2021
35. Створення Telegram бота [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/HnduBYU - Дата звернення 27.05.2021
36. Герберт Шилдт. Java: Полное руководство. Издатель Вильямс, 2012. 1101 с.
37. Джошуа Блох. Effective Java. Издатель Pearson Education, 2016. 265 с.
38. Васильев Алексей Николаевич. Java для всех. Издательский дом «Питер», 2019. 512 с.
39. Валерий Романчик, Игорь Блинов. Java. Методы программирования. Издатель Litres, 2017. 895 с.
40. Хабибулин Ильдар Шаукатович. Java. Издатель БХВ-Петербург, 2012. 768 с.

Додаток А

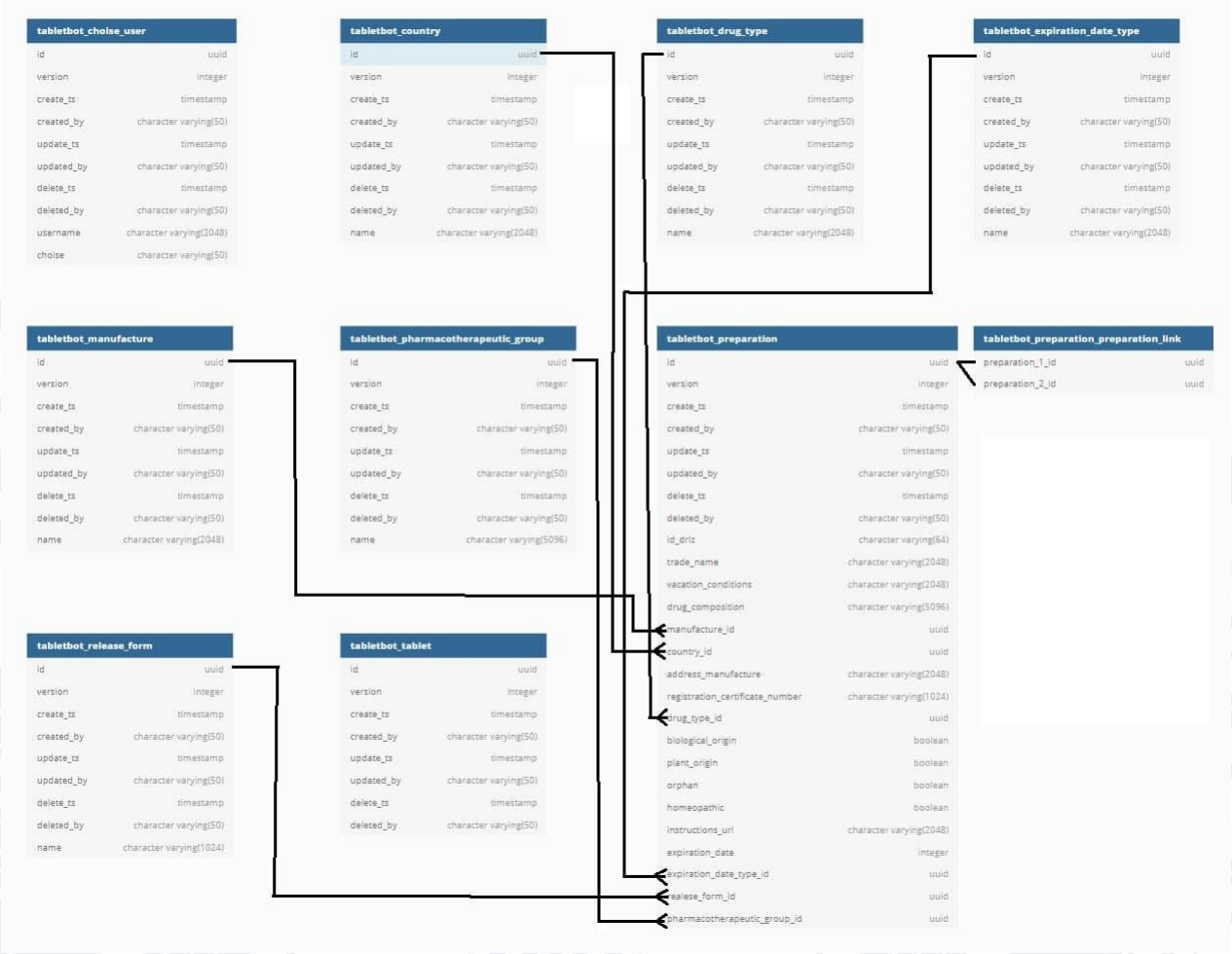


Рисунок – реляційна модель БД

Медецький Сергій Юрійович
Прізвище, ім'я по батькові

Факультет інформаційних і прикладних технологій
Факультет

122 Комп'ютерні науки
Шифр і назва спеціальності

Комп'ютерні технології обробки даних
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему « Рекомендаційна система підбору лікарських препаратів на основі Telegram-боту» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

(дата) (підпис здобувача освіти)