Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____Юрій КРАВЧЕНКО
«_____»_____2022 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»
освітньо-професійна програма «Мережеві та інтернет технологій»
на тему:

Розробка системи оперативного оповіщення студентів про розклад занять

Виконав: студент групи МІТ -41

Анатолій Камалетдінов

(ім'я та ПРІЗВИЩЕ) (підпис)

Керівник: завідувач кафедри мережевих та інтернет технологій
(посада)

д.т.н., проф. Юрій КРАВЧЕНКО

(науковий ступень, вчене звання, ім'я та ПРІЗВИЩЕ) (підпис)

Київ - 2022

Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Юрій КРАВЧЕНКО

«_____»_____2021 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Камалетдінову Анатолію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи:
Розробка системи оперативного оповіщення студентів про розклад заняття
затверджена на засіданні кафедри МІТ «24» грудня 2021 р. протокол №8
2. Термін здачі закінченої роботи «30» травня 2022 р.
3. Вихідні дані до проекту (роботи)
Сучасні технології для обміну текстовими повідомленнями
4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
Вступ
1. Дослідження технологій для обміну текстовими повідомленнями. Постановка задачі
1.1 Огляд та аналіз технології для обміну текстовими повідомленнями
1.2 Постановка задачі
2. Розробка системи оперативного оповіщення студентів про розклад заняття на основі менеджерів
3. Дослідження розробленої системи оперативного оповіщення
3.1. Алгоритм дослідження
3.2. Оцінка ефективності системи
3.3. Рекомендації до підвищення ефективності системи
Висновки
5. Перелік графічного матеріалу 8-10 слайдів
Дата видачі завдання _____
Керівник роботи _____ Юрій КРАВЧЕНКО
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання
Анатолій КАМАЛЕТДІНОВ
(підпис)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	02.05.2022	
2	Розділ 1	10.05.2022	
3	Розділ 2	15.05.2022	
4	Розділ 3	20.05.2022	
5	Доповідь та слайди	25.05.2022	
6	Пояснювальна записка	30.05.2022	

Здобувач вищої освіти _____ Анатолій КАМАЛЕТДІНОВ
(підпис)

Керівник _____ Юрій КРАВЧЕНКО
(підпис)

РЕФЕРАТ

Пояснювальна записка: 38 с., 12 рис., 3 табл., 16 джерел .

Мета роботи: розробка системи оперативного оповіщення студентів про розклад занять

Об’єкт дослідження: побудова систем сповіщень за допомогою месенджера.

Предмет дослідження: Java Spring Boot додаток для посилання сповіщень через Telegram-бота використовуючи Web-Hook

Методи дослідження: було використано методи системного підходу, функціонально-структурні методи а також метод узагальнення. Предмет дослідження був представлений у вигляді окремих систем, що спрощує структуризацію функцій кожної із них. Системи було досліджено на здатність синтезу та взаємодії, а також згруповано до монолітного механізму. Після точних вимірювань, проведення експериментів та досліджень було розроблено систему сповіщення за допомогою месенджера.

Протягом виконання дипломної роботи було досліджено технології для обміну текстовими повідомленнями. Було вивчено принцип роботи telegram-бота, взаємодію мессенджера з Java-додатком. За допомогою додатку було перевірено збереження повідомлень, забезпечено їх доступність та планування часу відправлень кожного із них. Використовуючи наявні інструменти було створено повноцінний додаток для встановлення, редагування а також надсилання сповіщень студентам щодо розкладу занять. Додаток було протестовано а також виявлено певні фактори, що можуть підвищити ефективність та стабільність його роботи

Практичне застосування додатку може бути використане не лише для оповіщення студентів про розклад. Він може реалізовувати функції оповіщення про будь-які системні процеси, що прив’язані до фіксованого часу, навіть інтернаціонально, з урахуванням різних часових поясів користувачів.

Оригінальність розробки забезпечена тим, що додаток здатен забезпечувати збереження сповіщень на будь-який термін, здатен автоматично виявляти часову зону користувача а також забезпечувати безперебійність доставки повідомлень, крім того є досить широко функціональним, що дає змогу використовувати його у багатьох індивідуальних випадках.

Галузь використання: розробка Telegram-бота та його управління за допомогою Java-додатку

Ключові слова: MESSENGER, TELEGRAM, BOT, API,JAVA, SPRING BOOT, WEBHOOK, NGROK

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДП – Дипломний Проект

Webhook - метод збільшення або розширення функціональності веб сторінки, або запиту

HTTP - HyperText Transfer Protocol

API - Application Programming Interface

REST - Representational state transfer

Telegram Bot - Telegram accounts operated by software

IDE – Integrated Development Environment

Spring Boot - Java framework for creating application

SQL - Structured Query Language

ВСТУП

Месенджери - системи, що надають можливість обміну текстовими, та іншими типами повідомлень між телефонами, комп'ютерами і деякими пристроями користувачів. Також їх часто називають системами миттєвих повідомлень. Відмінність миттєвих повідомлень від, наприклад, скриньки e-mail, полягає в тому, що користувачі обмінюються повідомленнями в режимі реального часу, на відміну від електронної пошти, де повідомлення зберігаються на сервері та для їх отримання необхідно зайти на свою адресу і прочитати їх. У мессенджерах клієнти під’єднані один до одного постійно, а тому повідомлення передаються миттєво.

Користувачі можуть обмінюватись повідомленнями приватно, один-на-один, або в бесідах, спільних чатах, де приймає участь одразу велика кількість співрозмовників.

Існують певні протоколи для роботи месенджерів. Протоколи поділяються на серверні і безсерверні. Безсерверні протоколи підтримують підключення клієнтів між собою, тобто повідомлення від одного користувача надсилається безпосередньо іншому.

Натомість більшість з месенджерів є саме серверними. У серверних протоколах, системи повідомлень не працюють окремо, а приєднуються до основного сервера, головного комп’ютера що відповідає за обмін повідомлень. А тому такі системи називають клієнтськими.

Системи обміну повідомленнями містять декілька складових: Ідентифікація(облік, або адресація) клієнтів, система перегляду стану клієнта (наявність/відсутність підключення), система надсилання повідомлень (відправка напряму іншому користувачу, або відправка на сервер).

За допомогою сервера вирішується проблема ідентифікації клієнтів. Різні месенджери призначені для різних цілей можуть використовувати будь-які

дані для ідентифікації користувацьких підключень. У більшості сучасних систем обміну повідомленнями це мобільний номер, або e-mail, але в якості ідентифікатора може слугувати будь що, у цьому і полягає гнучкість використання серверної частини. Надалі сервер займатиметься розпізнаванням користувача, та встановленням адресації повідомлень.

Більшість месенджерів надає користувачам змогу переглядати стан співрозмовника. Найбільш поширені стани це online/offline/invisible. Online - користувач активний та під'єднаний до мережі, offline - користувач не підключений, invisible - користувач вимкнув можливість переглянути його статус. Також деякі системи обміну повідомленнями дозволяють встановити певний статус - наприклад busy - зайнятий, або away - відійшов, та інші, що дозволяють інформувати співрозмовника про стан користувача.

Надсилання повідомлень, їх збереження у разі не активності співрозмовника, зберігання адрес клієнтів, їх контактів між собою та встановлення доступу забезпечується серверним додатком.

Найбільш популярними в світі месенджерами 2021 року були WhatsApp, Facebook Messenger, WeChat і QQ Mobile.

В Україні ж, за даними 2020 року, найпопулярніші месенджери - Viber, FaceBook Messenger, Telegram, WhatsApp. Останнім часом популярність WhatsApp спадає, а Telegram - набуває популярності.

Telegram - кроссплатформенний месенджер, з можливістю обміну текстовими, голосовими та відео повідомленнями, а також із надсиланням файлів та здійсненням дзвінків VoIP - Voice Over IP. Telegram з’явився у 2013-ому році, та вже у 2021 році кількість щомісячних користувачів становить близько 500 млн людей. За допомогою ботів, функціонал цієї системи майже не обмежений.

Telegram bot - спеціальний аккаунт в telegram, який, на відміну від інших акаунтів, керується системою, а не певною людиною. Системи приймають, обробляють повідомлення і здатні надсилати повідомлення у відповідь.

Тобто, телеграм-бота можна використовувати як інтерфейс роботи користувача з будь-якою серверною системою. Пересилати дані від бота до серверного додатку можна використовуючи Webhook. Вебхук - метод, що заключається у реагуванні на певну подію та надсиланні http-запиту на певну адресу. Тобто його можна використати для пересилання повідомлення від користувача до бота, та від бота до серверного додатку, подальшої обробки, та надсилання відповіді назад.

1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ДЛЯ ОБМІНУ ТЕКСТОВИМИ ПОВІДОМЛЕННЯМИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд і аналіз технологій для обміну повідомленнями

Технології обміну повідомленнями є дуже популярними та користуються великим попитом серед користувачів. Найпростішою подібною системою є система SMS - Short Message Service. Така система дозволяє користувачам більшості мобільних телефонів обмінюватися короткими повідомленнями за допомогою зв’язку з мобільною мережею. Перше смс-повідомлення було написане у 1992 році, після чого стало одним із найбільших джерел доходів мобільних операторів.

Смс повідомлення були популярними досить довго, але сьогодні їх виштовхують більш зручні, швидкі та широко функціональні мобільні месенджери.

Переваги їх у тому, що можна надсилати повідомлення будь-яких розмірів, та в необмеженій кількості. Наприклад, в SMS повідомлення є фіксованої довжини, а їх кількість напряму залежить від коштів на рахунку користувача. Також у месенджерах наявні відео-чати, можна відправляти стікери та фотографії. Функціонал месенджерів знатно ширший, а швидкість роботи вища - тому вони набувають такої популярності.

Серед основних переваг месенджерів можна виділити наступні:

- Зручна комунікація і швидкий зв'язок;
- Створення групових чатів;
- Надсилання голосових та відео повідомлень
- Надсилання файлів, без обмежень за типом файлу, з можливістю надсилати великі файли, їх збереження в чаті
- Відеодзвінки, та голосові дзвінки VoIP - voice over IP
- Швидка передача даних;

- Кросплатформенність.

Розробка подібних систем починалася з додатку ICQ. Сервіс і зараз пропонує той самий набір функцій, що у конкурентів, але все одно вже не такий популярний. У часи коли він був розроблений, у 1996 році, школярі старших класів із Ізраїлю заснували компанію Mirabilis, яка створила знаменитий інтернет-пейджер. Згодом, у 1998 році ICQ викупила компанія AOL за 407 мільйонів доларів. У 2004 році користувачі мали змогу користуватися додатком Jimm - мобільною версією ICQ.

Перші месенджери з'явилися на мобільних телефонах із операційними системами SYMBIAN та Windows mobile. На жаль в той час, мобільний інтернет не був досить розвинений та не міг забезпечити безперебійну роботу додатків. Крім того, знаходитись онлайн можна було лише з одного пристрою, тобто заходячи з телефона, додаток автоматично виходив з аккаунту на комп'ютері та навпаки. Серед недоліків ICQ можна спостерігати:

- Централізованість;
- Нестабільність;
- Незахищеність
- Погана робота при слабкому підключенні
- Велика кількість обмежень
- Закритість протоколу та офіційних клієнтів
- Проблеми з підтримкою декількох мов.

Сучасні технології - швидкий мобільний інтернет за доступною вартістю, швидкодіючі смартфони, що можуть виконувати десятки завдань одночасно, а системи обміну повідомлень доступні майже на будь-яких пристроях. На сьогоднішній день користуватися месенджером значно легше. Наступним ступенем еволюції серед месенджерів можна назвати WhatsApp.

Його розробили Ян Кум та Брайан Актон, що раніше працювали в Google та Yahoo. Сьогодні це один з найпопулярніших месенджерів в світі, що налічує майже 500

мільйонів користувачів по всьому світу. Основними нововведеннями WhatsApp були:

- Можливість перегляду статусу своїх контактів (онлайн, спілкуюсь по телефону, зайнятий)
- Прив’язка до телефонного номеру

Крім того, саме у тий час у Apple з’явилася можливість надсилати push-повідомлення, що й стало ключем до успіху WhatsApp.

Для передачі повідомлень у WhatsApp використовується модифікований протокол обміну повідомленнями та інформація про наявність XMPP (раніше відомий як Jabber) із шифруванням SSL та TLS. SSL і TLS – це широко використовувані криптографічні протоколи, що забезпечують захисну передачу даних у мережі Інтернет. Крім того, протокол WhatsApp застосовується у веб-браузерах та при роботі з електронною поштою.

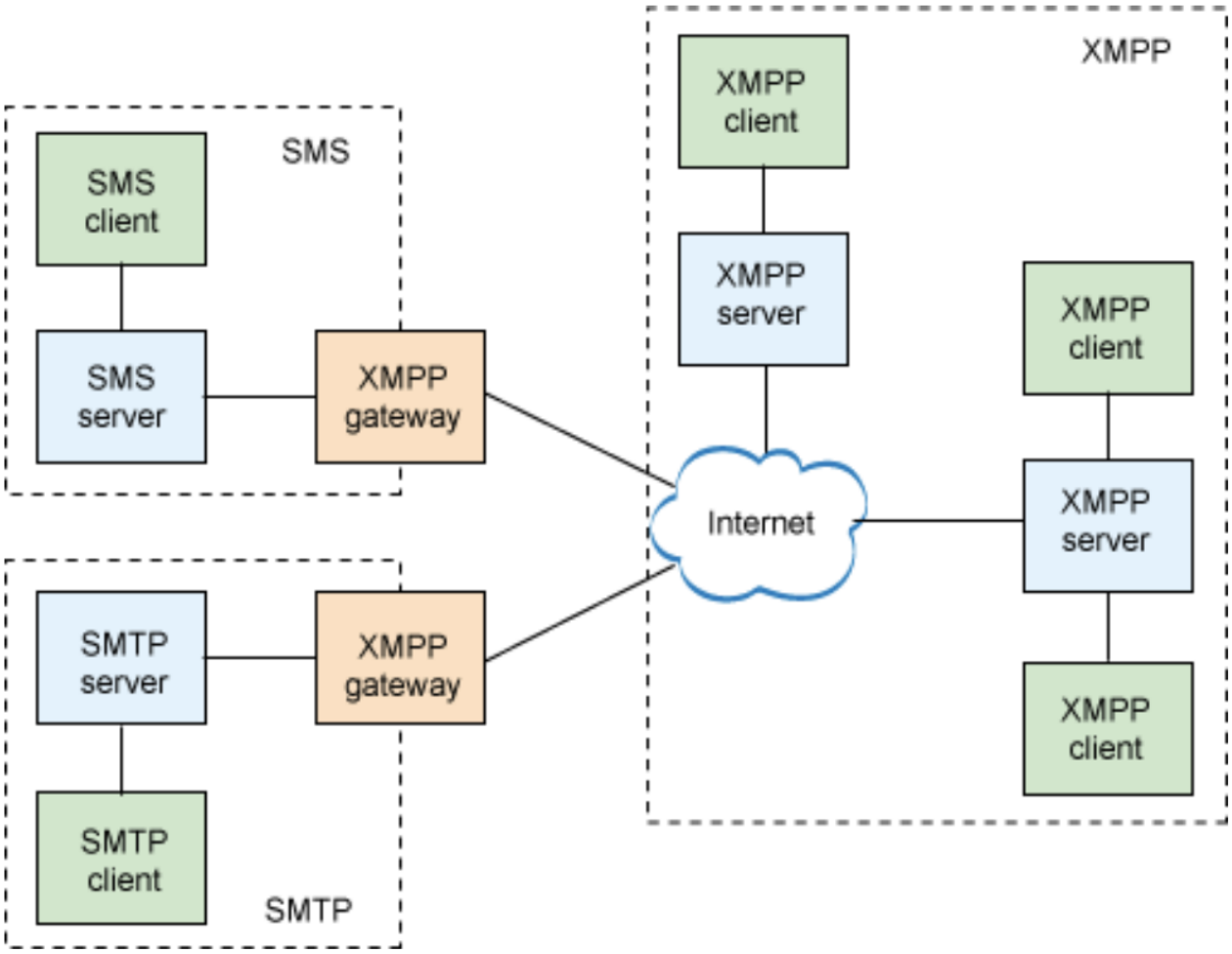

Рис.1.1 - Архітектура XMPP

Поки користувач не запустив додаток, усі вхідні повідомлення вибудовуються в черзі і зберігаються в пам'яті сервера. Як тільки повідомлення потрапляє на пристрій користувача, воно стирається з пам'яті сервера, а відправник отримує підтвердження про його отримання.

У якості СУБД використовується багато користувацька розподілена система Mnesia DB, яка використовує близько 2 ТБ RAM і зберігає порядка 18 мільярдів записів. Вона написана на мові розробки Erlang. Судячи з швидкості обробки запитів виходить досить ефективна інтеграція. Фото, аудіо та відео, прикріплені до

повідомлення, загружаються на веб-сервер YAWS, а посилання на них відправляються отримувачеві.
Використання Erlang принесло неймовірні показники масштабованості. Початкове навантаження WhatsApp складало 200 000 одночасних з'єднань на сервер. Трохи згодом цей показник значно виріс і досяг 2,8 мільйонів з'єднань. Усю цю кількість інформації обробляють приблизно 550 серверів, з яких 150 - це чат-сервери, що підтримують до 1 мільйона смартфонів. Ще близько 250 машин – це мультимедіа сервери. Вузли, відповідальні роботу баз даних, мають 512 ГБ оперативної пам'яті, а стандартні обчислювальні вузли – 64 ГБ.

WhatsApp довів, що розмір компанії не має значення. Команда з кількох десятків розробників змогла створити популярний месенджер, використовуючи недороге обладнання та відкрите ПЗ.

З недоліків WhatsApp можна виділити наступні:

- Більшість повідомлень не шифрується;
- Повідомлення видаляються тільки у вас;
- Недосконалий алгоритм шифрування;
- Ваші дані можуть передаватися на Facebook;
- Особисті дані зберігаються в хмарному сховищі;

Також досить популярним мессенджером є Viber. Viber також прив’язується до номеру телефону, а також сповіщає ваші контакти, якщо ви встановили його пізніше за них. Перша версія була розроблена у грудні 2010 року, у 2014 році японська компанія Rakuten викупила вайбер.
Viber надає можливості:

- вибрати оформлення кожного окремого чата;
- відправляти не тільки фотографії, але і намальовані пальцем зображення;

− здійснювати дзвінки на будь-які мобільні номери, навіть якщо їх власників немає у Вайбері (за ці дзвінки здійснюється окрема плата);

Серед недоліків можна спостерігати:

− обмеження на розмір файлів, що відправляються – 200 Мб;

− тривалість голосових повідомлень – максимум 1 хвилина;

− кількість учасників у групових чатах – не більше 250;

− безліч рекламних повідомлень і спаму;

− Нестабільна робота;

− Погано налагоджена система синхронізації повідомлень на різних платформах;

Багато месенджерів мають проблеми із захищеністю, саме тому доцільно буде згадати telegram. Telegram для захисту повідомлень використовує власний протокол MTProto - це криптографічний протокол, що полягає в комбінуванні симметричного алгоритму AES, протоколу Діффі-Хеллмана для обміну 2048-бітними RSA-ключами між двома пристроями та ряду хеш-функцій.

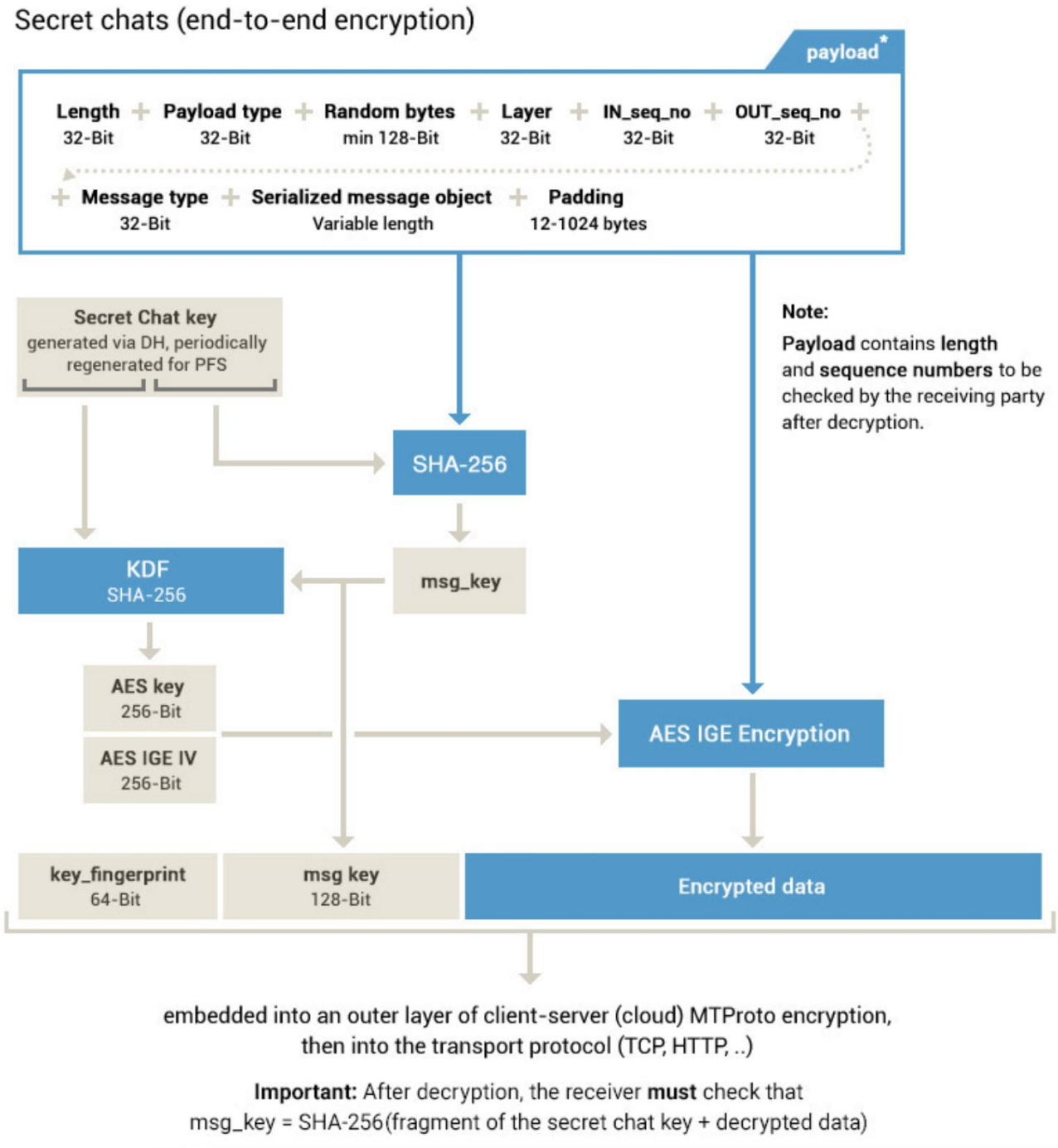

Рисунок 1.2 - Схема шифрування MTProto 2.0

Переваги telegram:

- У Telegram можна редагувати надіслані повідомлення;
- У Telegram можна створити лише декілька облікових записів;
- У Telegram набагато менші обмеження на передачу файлів;

- У Telegram окремі діалоги групуються до папок
- У Telegram вбудовано хмарне сховище даних
- У Telegram є надсилання повідомлень за таймером
- Повідомлення Telegram зручно шукати за хештегами
- У Telegram є можливість створювати опитування та наявні інші соціальні функції;
- Інтерфейс Telegram можна кастомізувати
- У Telegram можна сховати свій номер, своє фото, інші особисті дані, а також розмежовувати доступ до них;
- Усі програми Telegram працюють незалежно
- У Telegram можна закріпити повідомлення у групових чатах
- У Telegram можна обрати якість надісланого фото
- Telegram налаштовано та надійно зберігає чати та діалоги;
- У Telegram є велика кількість віджетів для домашнього екрану;
- У Telegram є велика кількість корисних каналів;
- Telegram постійно покращується та розвивається ;
- Telegram працює з міні-додатками - ботами

1.2 Постановка завдання

Необхідно створити єдиний додаток - систему, що складається з кількох підсистем.

Перша підсистема - telegram-bot, необхідна для взаємодії з клієнтом через месенджер telegram. Система повинна мати змогу налаштовуватися, отримати команди, повідомлення від користувача, а також надсилати ці команди та повідомлення до серверної частини додатку для подальшої обробки.

Друга підсистема - Spring boot додаток, написаний на мові Java, що здатний отримувати повідомлення, обробляти їх, та надсилати відповідь. Сервер має виконувати також функції ідентифікації користувача, має зберігати інформацію про ідентифікатор чату для кожного окремого користувача. Крім того, серверний додаток має визначати часову зону користувача, зберегти цю інформацію. Також, додаток має забезпечувати заплановане надсилання повідомлень, можливість їх редагування, оновлення та видалення.

На основі двох виділених систем має бути побудований повноцінний додаток для роботи з сповіщеннями у вигляді повідомлень у месенджері telegram.

Потрібно встановити чіткий, стабільний зв’язок між двома складовими використовуючи технологію webHook, та забезпечити безперебійну доставку повідомлень між сервісом, що слугує інтерфейсом для користувача, та сервісом, що обробляє нагадування, та надсилає їх у необхідний момент часу.

Необхідно налагодити здобуття інформації про користувача - його ідентифікатора, часового поясу, налагодити зберігання та запам’ятовування системою цієї інформації для подальшого використовування при плануванні часу відправки повідомлення а також адреси отримувача. Крім того, в разі збою системи, додаток повинен мати функцію відновити вже встановлені користувачем нагадування, та відновити запланування їх відправки.

2. РОЗРОБКА СИСТЕМИ ОПЕРАТИВНОГО СПОВІЩЕННЯ СТУДЕНТІВ ПРО РОЗКЛАД ЗАНЯТЬ.

2.1 Вибір технології програмування

Java Spring Boot - фреймворк мови Java для створення серверних (бекенд) додатків широкого спектру використання. Spring Boot додаток може бути використаний як бекенд для сайту, так і в ролі обробника запитів з інших джерел, в тому числі і запитів із telegram-bot-ів.

Spring Boot має великий функціонал, але його найбільш значущими особливостями є: управління залежностями, автоматична конфігурація та вбудовані контейнери сервлетів.

Сервлет - стандартизований API для створення динамічного запиту до веб сервера, використовуючи платформу Java, використовує технологію запит-відповідь.

Щоб прискорити процес управління залежностями, Spring Boot неявно пакує необхідні сторонні залежності для кожного типу програми на основі Spring і надає їх розробнику за допомогою так званих starter-пакетів (spring-boot-starter-web, spring-boot-starter-data-jpa і т.д.). Starter-пакети є набором зручних дескрипторів залежностей, які можна включити у свою програму. Це дозволить отримати універсальне рішення для всіх, пов'язаних зі Spring технологій, позбавляючи програміста від зайвого пошуку прикладів коду та завантаження з них необхідних дескрипторів залежностей.

Наприклад, якщо необхідно почати використовувати Spring Data JPA для доступу до бази даних, потрібно включити у свій проект залежність spring-boot-starter-data-jpa і робота із доступом до бази даних буде майже готова. Якщо стоїть задача створити Spring web-додаток, можна просто додати залежність spring-boot-starter-web, яка підтягне в проект усі бібліотеки, необхідні для розробки Spring MVC-додатків, таких як spring-webmvc, jackson-json, validation-api та Tomcat.

Іншими словами, Spring Boot збирає всі загальні залежності та визначає їх в одному місці, що дозволяє розробникам просто використовувати їх.

Другою чудовою можливістю Spring Boot є автоматична конфігурація програми. Після вибору відповідного starter-пакету, Spring Boot спробує автоматично налаштувати Spring-додаток на основі доданих вами jar-залежностей.

Кожен Spring Boot web-додаток включає в себе вбудований web-сервер. Розробникам не потрібно турбуватися про налаштування контейнера сервлетів та розгортання програми на ньому. Тепер програма може запускатися сама, як виконуваний jar-файл з використанням вбудованого сервера.

Якщо потрібно використовувати окремий HTTP-сервер, для цього достатньо виключити залежності за замовчуванням. Spring Boot надає окремі starter-пакети для різних HTTP-серверів.

Створення автономних web-додатків з вбудованими серверами не тільки зручно для розробки, але і є допустимим рішенням для програм корпоративного рівня і стає все більш корисним у світі мікросервісів. Можливість швидко спакувати весь сервіс (наприклад, автентифікацію користувача) в автономному артефакті, що повністю розгортається, який також надає API — робить установку і розгортання програми значно простіше.Spring MVC – це веб-фреймворк Spring. Він дозволяє створювати веб-сайти або RESTful сервіси (наприклад, JSON/XML) і добре інтегрується в екосистему Spring, наприклад він підтримує контролери та REST контролери у ваших Spring Boot додатках.

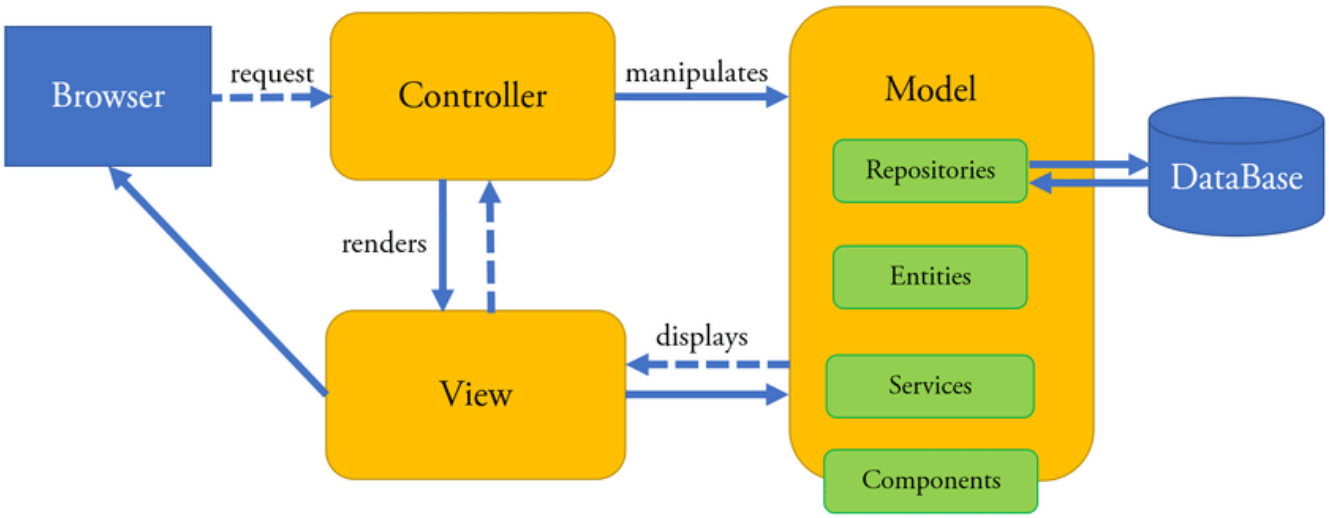

Рисунок 2.1 - Структура додатку у Spring Boot MVC

Саме Controller відповідає за отримання запиту, його обробку, а також надсилання відповіді на нього.

Для пересилання повідомлень з месенджера до додатку, буде використане перенаправлення за допомогою webHook. Telegram сервер відправлятиме на зареєстровану адресу webhook повідомлення у форматі JSON методом POST контролер їх прийматиме та передаватиме бібліотеці telegram у вигляді об'єкта Update.

Об’єкти, використовувані для взаємодії telegram месенджера із додатком Spring Boot через бота, будуть представлені у вигляді JSON - JavaScript Object Notation - стандартний текстовий формат для представлення структурованих даних.

Об’єкти JSON мають наступний вигляд:


```
{
   "update_id": 12672073,
   "message": {
      "message_id": 928,
      "from": {
         "id": 998832143,
         "is_bot": false,
         "first_name": "Marcin",
         "last_name": "Dudek",
         "language_code": "en"
      },
      "chat": {
         "id": -499910330,
         "title": "New Bot Test",
         "type": "group",
         "all_members_are_administrators": true
      },
      "date": 1583398919,
      "text": "\/getinfo",
      "entities": [
         {
            "offset": 0,
            "length": 8,
            "type": "bot_command"
         }
      ]
   }
}
```


Рисунок 2.2 - JSON Object повідомлення Telegram

Отже, для розробки додатку було використано Spring Boot, Telegram API та WebHook.

2.2 Розробка telegram-bot

Для початку необхідно створити самого бота, через якого буде проходити взаємодія користувача та серверного додатку. Розробники telegram дають можливість легко створювати ботів за допомогою офіційного джерела - боту @BotFather.

Цей засіб є найлегшим способом створення свого власного боту. Потрібно лише мати аккаунт telegram, далі за допомогою бота @BotFather, потрібно зареєструвати свого бота, дати йому ім’я та нікнейм-адресу у telegram, пройти усі налаштування, у відповідь отримати токен.

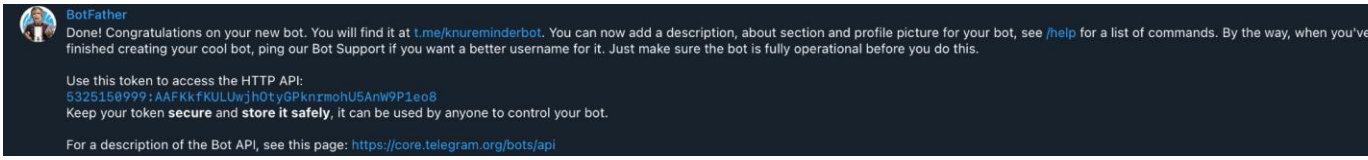

Рисунок 2.3 - Необхідна інформація про бота.

Після реєстрації та встановлення ім’я для бота, він одразу стає доступним для всіх і йому навіть можна написати повідомлення.

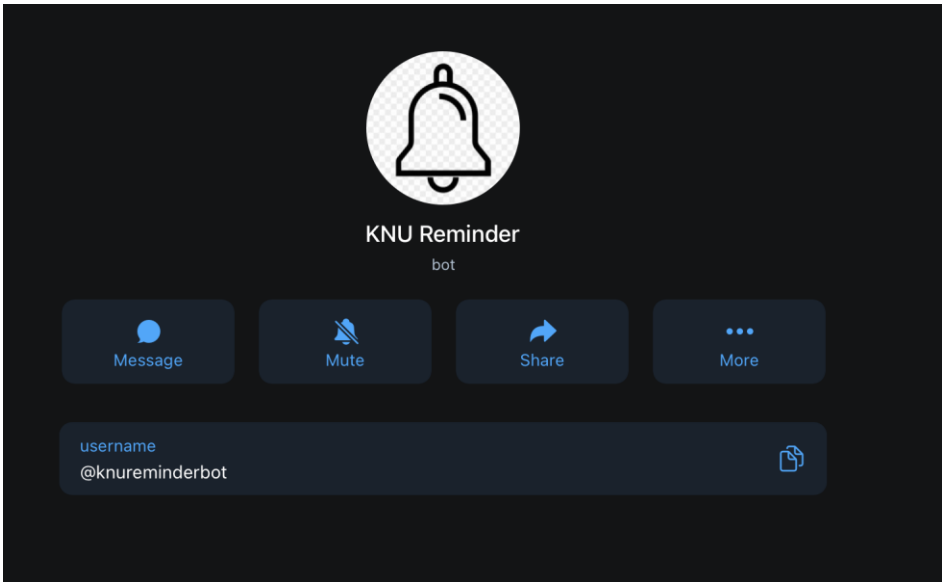

Рисунок 2.4 - Вигляд бота після створення

Без логіки опрацювання повідомлень вони просто будуть збережені в чаті з ботом.

Надані ім’я та токен бота будуть використані пізніше для конфігурації серверного додатку, та для встановлення взаємодії між ботом, telegram та додатком.

2.3 Розробка Spring Boot додатку

За допомогою будь-якої IDE для роботи з Java можна створити проект Spring Boot Application, або ініціалізувати його через офіційний сайт - https://start.spring.io. Після створення проекту, необхідно налаштувати його залежності:

Для роботи з telegram необхідна наступна залежність:

```xml
<dependency>
    <groupId>org.telegram</groupId>
    <artifactId>telegrambots-spring-boot-starter</artifactId>
    <version>5.2.0</version>
</dependency>
```

Рисунок 2.5 - Залежність для роботи з telegram

Для роботи з базою даних та моделювання Data Access Objects необхідні наступні залежності:

```xml
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

Рисунок 2.6 - Залежності для роботи з базою даних

Також для зв'язування Spring Boot додатку із telegram необхідно створити класс TelegramBotConfig, який зберігатиме у собі інформацію, отриману при створенні telegram - бота (username, bot token, webhook path), а у файлі properties вказати актуальні значення:

telegrambot.userName=@knureminderbot

telegrambot.botToken=5325150999:AAFKkfKULUwjhOtyGPknrmohU5AnW9P1eo8

Після налаштування конфігурації для майбутнього бота, можна переходити до створення безспосередньо класу TelegramBot, що розширює класс SpringWebhookBot, із бібліотеки telegram. Необхідно реалізувати метод onWebhookUpdateReceived() - що відповідатиме за приймання повідомлень у вигляді JSON об’єктів а також у поверненні відповіді у том форматі, який буде зрозумілий для telegram.

Далі, для запуску Spring Boot додатку, необхідно встановити утиліту ngrok. Дана утиліта дозволяє створювати тимчасову адресу, та перенаправляти надіслані на неї запити у вказаний порт локального сервера. Під час запуску утиліти, необхідно викликати команду *ngrok http 5000*, або із використанням зручного порту. Після виконаних дій результат матиме вигляд:

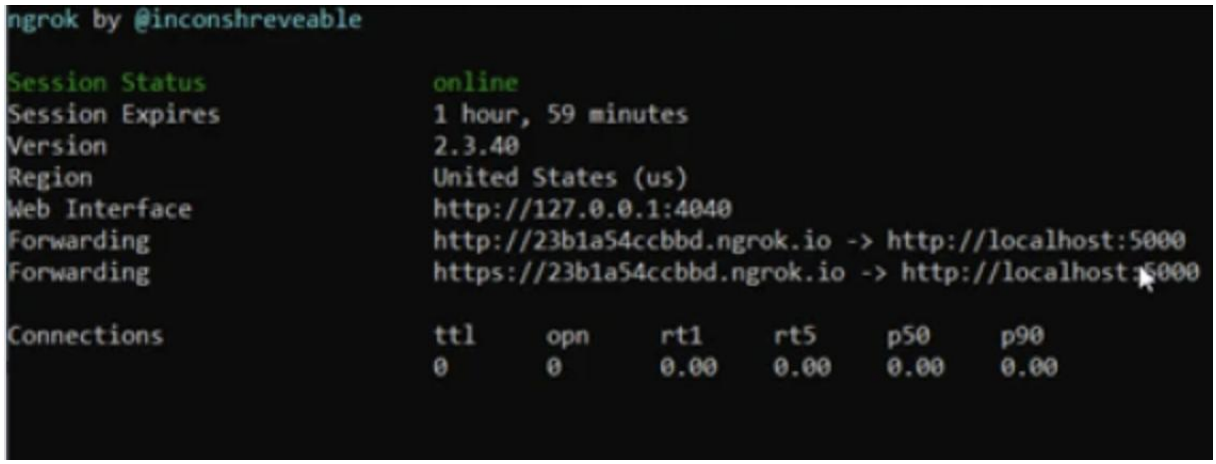

Рисунок 2.7 - Адреса NGROK та перенаправлення на локальний порт

Отриману адресу необхідно внести у файл properties: telegrambot.webHookPath=https://cd5f-128-124-133-45.eu.ngrok.io

Після запуску додатку, всі повідомлення направлені від користувача до бота будуть перенаправлені на розроблений Spring Boot додаток для подальшої обробки.

Наступним кроком буде забезпечення взаємодії роботи Spring Boot додатку із базою даних, а також збереження необхідних даних у таблиці БД. У файлі properties необхідно зробити певні зміни відповідно до даних використовуваної

бази даних:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/telegramusers
spring.datasource.username=postgres
spring.datasource.password=password
```

У самій базі даних необхідно створити таблиці, що будуть використані для роботи з користувачами і повідомленнями:

```
CREATE TABLE users
(
    id              INTEGER PRIMARY KEY UNIQUE NOT NULL,
    name            VARCHAR,
    time_zone       INTEGER DEFAULT 0,
    on_off          BOOLEAN DEFAULT true
);

CREATE TABLE user_events
(
    user_id INTEGER ,
    time timestamp ,
    description varchar ,
    event_id serial,
    event_freq varchar default 'TIME',
    FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
);

CREATE TABLE event_cash
(
    time timestamp ,
    description varchar ,
    user_id INTEGER ,
    id serial
);
```

У таблиці users ми будемо записувати ідентифікатор облікового запису користувача telegram, його ім'я, яке може і не бути, буде розглянутий часовий пояс

користувача, для коректної відправки повідомлень, а також стан увімкнення/вимкнення розсилок повідомлень.

У таблиці user_events у нас буде записаний id користувача, час уведомления, опис, буде автоматично генеруватися id для подій та встановити частоту відомих.

У таблиці event_cash буде записуватися повідомлення перед відправкою, і, якщо воно відправлено, то видалятися з таблиць.

Потім, необхідно створити відповідні сутності у Java коді -

```
@Entity
@Table(name = "users")
@Getter
@Setter
public class User {

@Entity
@Table(name = "user_events")
@Getter
@Setter
public class Event {

@Entity
@Table(name = "event_cash")
@Getter
@Setter
//serves to save unhandled events after rebooting heroku
public class EventCashEntity {
```

А також створити репозиторії та сервіси, для використання CRUD операцій із базою даних.

Після забезпечення доступу до збереження/редагування/видалення даних з таблиць можна починати роботу над обробкою повідомлень, та відправкою відповіді. Для цього потрібно створити клас TelegramFacade, та розробити в ньому логіку обробки повідомлень в методі handleInputMessage():

```
private BotApiMethod<?> handleInputMessage(Message message) {
    BotState botState;
    String inputMsg = message.getText();
    //we process messages of the main menu and any other messages
    //set state
    switch (inputMsg) {
        case "/start":
            botState = BotState.START;
            break;
        case "Мої нагадування":
            botState = BotState.MYEVENTS;
            break;
        case "Створити нагадування":
            botState = BotState.CREATE;
            break;
        case "Вимкнути нагадування":
        case "Увімкнути нагадування":
            botState = BotState.ONEVENT;
            break;
        case "All users":
            if (message.getFrom().getId() == adminId)
                botState = BotState.ALLUSERS;
            else botState = BotState.START;
            break;
        case "All events":
            if (message.getFrom().getId() == adminId)
                botState = BotState.ALLEVENTS;
            else botState = BotState.START;
            break;
        default:
            botState = botStateCash.getBotStateMap().get(message.getFrom().getId()) == null?
                    BotState.START: botStateCash.getBotStateMap().get(message.getFrom().getId());
    }
    //we pass the corresponding state to the handler
    //the corresponding method will be called
    return messageHandler.handle(message, botState);

}
```

Рисунок 2.8 - Приклад обробки прийнятих від користувача повідомлень

Для того, щоб бот розумів, що від нього очікують у певний момент часу, наприклад, для цього необхідно ввести классBotState, що зберігатиме стани бота.

public enum BotState {

```
    ENTERDESCRIPTION,//the bot will wait for the description to be entered.
    START,
    MYEVENTS, //the bot show to user list events.
    ENTERNUMBEREVENT,//the bot will wait for the number of event to be entered.
    ENTERDATE, //the bot will wait for the date to be entered
    CREATE, //the bot run created event
    ENTERNUMBERFOREDIT, //the bot will wait for the number of event to be entered
    EDITDATE, //the bot will wait for the date to be entered
    EDITDESCRIPTION,//the bot will wait for the description to be entered
    EDITFREQ,//the bot will wait callbackquery
    ALLUSERS, // show all users
    ALLEVENTS, //show all events
    ENTERNUMBERUSER,//the bot will wait for the number of user to be entered.
    ENTERTIME,//the bot will wait for the hour to be entered.
    ONEVENT // state toggle
}
```

Залежно від станів бота реалізується логіка обробки повідомлень:

```
switch (botState.name()) {
        case ("START"):
            return menuService.getMainMenuMessage(message.getChatId(),
                    "Воспользуйтесь главным меню", userId);
        case ("ENTERTIME"):
            //set time zone user. for correct sent event
            return eventHandler.enterLocalTimeUser(message);
```

```
        case ("MYEVENTS"):
            //list events of user
            return eventHandler.myEventHandler(userId);
        case ("ENTERNUMBEREVENT"):
            //remove event
            return eventHandler.removeEventHandler(message, userId);
        case ("ENTERDESCRIPTION"):
            //enter description for create event
            return eventHandler.enterDescriptionHandler(message, userId);
        case ("ENTERDATE"):
            //enter date for create event
            return eventHandler.enterDateHandler(message, userId);
        case ("CREATE"):
            //start create event, set state to next step
            botStateCash.saveBotState(userId, BotState.ENTERDESCRIPTION);
            //set new event to cache
            eventCash.saveEventCash(userId, new Event());
            sendMessage.setText("Введите описание события");
            return sendMessage;
        case ("ENTERNUMBERFOREDIT"):
            //show to user selected event
            return eventHandler.editHandler(message, userId);
        case ("EDITDESCRIPTION"):
            //save new description in database
            return eventHandler.editDescription(message);
        case ("EDITDATE"):
            //save new date in database
```

```
            return eventHandler.editDate(message);
        case ("ALLEVENTS"):
            //only admin
            return eventHandler.allEvents(userId);
        case ("ALLUSERS"):
            //only admin
            return eventHandler.allUsers(userId);
        case ("ONEVENT"):
            // on/off notification
            return eventHandler.onEvent(message);
        case ("ENTERNUMBERUSER"):
            //only admin
            return eventHandler.removeUserHandler(message, userId);
        default:
            throw new IllegalStateException("Unexpected value: " + botState);
```

Для того, щоб користувач міг викликати вже готові методи, а не власноруч вгадувати команди бота, необхідно ввести клас MenuService, що залежно від станів бота повертатиме користувачеві певний набір кнопок з командами, за допомогою яких відбувається взаємодія бота і додатку.

Останнім кроком створення додатку є розробка системи запланованих сповіщень. Клас має назву EventService, працює за розкладом, використовуючи механізм Spring Boot під назвою *@EnableScheduling* та @Scheduled(cron = "0 0 0 * * *"). Отже, метод сервісу буде викликаний за певним розкладом. Під час виконання, метод читатиме усі сповіщення, встановлені користувачем, а також буде формулювати запланований розклад надсилання сповіщень. Тобто, сервіс

періодично перевірятиме наявність нових сповіщень, і робитиме це окремо від надсилання вже запланованих користувачем сповіщень.

3. ДОСЛІДЖЕННЯ СИСТЕМИ СПОВІЩЕННЯ СТУДЕНТІВ

3.1 Алгоритм дослідження додатку

Для дослідження працездатності додатку, необхідно переконатися у коректній роботі його окремих частин, а також у правильній їх взаємодії. За структурою, повідомлення від користувача, має надходити до бота, бот має перенаправляти ці повідомлення до певної адреси, що за допомогою webHook доставить його у Spring Boot додаток. Крім того, повідомлення мають бути правильної структури, а отже меню взаємодії з ботом правильно налагоджене.

Додаток у свою чергу за допомогою заданих команд має забезпечувати збереження нагадування, можливість їх редагування, запланування їх відправки, а також надавати коректні читабельні користувачеві відповіді.

Отже для початку необхідно переконатися у надходженні повідомлень від користувача ботові, а від бота - серверному додатку. Для цього необхідно знайти бота за адресою (@knureminderbot), та натиснути одну з його команд.

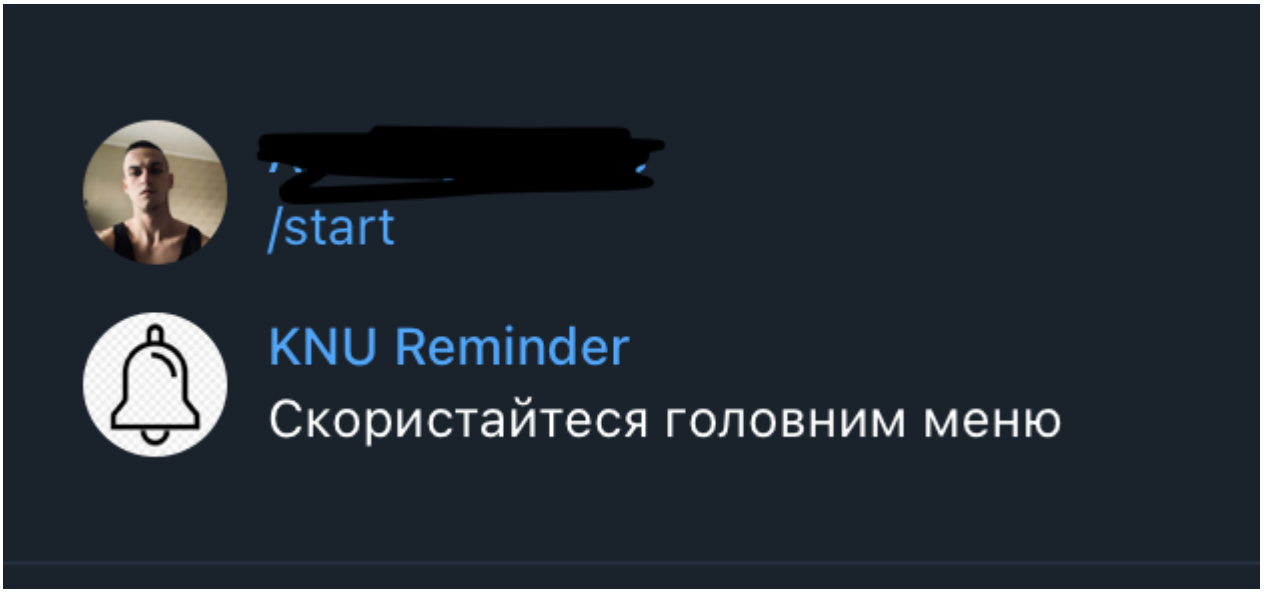

Рисунок 3.1 - Надсилання команди боту та отримання відповіді

Повідомлення “Скористайтеся головним меню” надходить від сервера, тобто можна з упевненістю сказати, що повідомлення від користувача надходять до бота, від бота успішно пересилаються серверному додатку, додаток коректно обробляє повідомлення, і надсилає відповідь.

Крім того, необхідно перевірити встановлення нагадувань, а також надсилання їх від сервера користувачеві у встановлену дату або встановлений час і з відповідним повідомленням. Для цього потрібно скористатися інтерфейсом (меню) та вибрати функцію “створити нагадування”, ввести опис нагадування, ввести його дату, обрати періодичність повторень, або разове нагадування. Після збереження, переконатися що нагадування є на сервері, за допомогою кнопки “Мої нагадування”

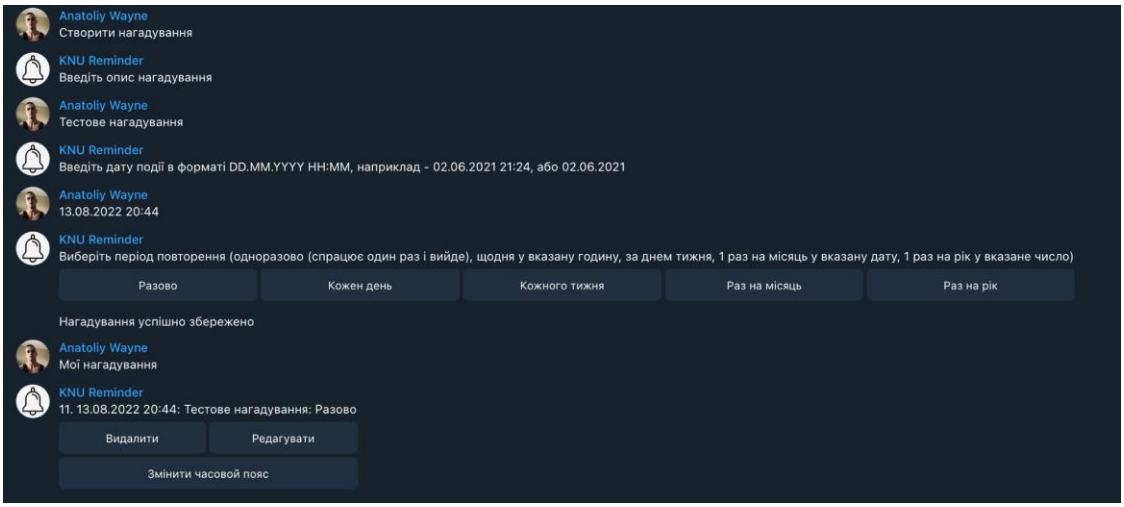

Рисунок 3.2 - Перевірка збереження нагадувань на сервері.

У відповідний час, на який заплановано нагадування, воно буде вислане ботом у вигляді звичайного повідомлення, разом із вказаним описом нагадування.

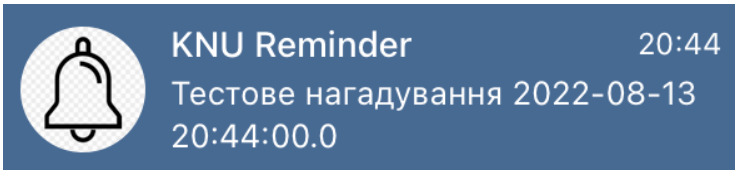

Рисунок 3.3 - Нагадування надійшло у встановлений час

3.2 Оцінка ефективності додатку

Додаток знаходиться у стані альфа-тестування, тому дослідити його ефективність саме зараз досить важко. Додаток стабільно та безперебійно виконує задані функції. Задля оцінки його ефективності можна сформулювати наступні метрики.

Кількість реєстрацій. Додаток містить необхідні таблиці для обліку кількості нових користувачів, за цим показником не можна визначити саме ефективність, але можна визначити швидкість розповсюдження а також розрахувати кількість надходження нових клієнтів протягом певного часу.

Активність у додатку. DAU, або щоденні активні користувачі — щоденне відвідування. Метрика відражає дні, коли клієнти частіше запускають додаток. WAU, або щотижневі активні користувачі — щотижнева активність. Сегмент показує поведінку користувачів через сім днів після встановлення програми. MAU, або щомісячні активні користувачі — щомісячна активність користувачів. Показник дає значення про періодичність запусків додатків.

Розглянуті вище метрики допомагають розраховувати залученість користувачів, або липкість (Sticky Factor) — показник, який відображає корисність та інтерес:

Тижневу ступінь залученості: DAU / WAU * 100%.

Місячна ступінь залученості: DAU / MAU * 100%.

В даному додатку необхідно розглядати ефективність роботи додатку, як ефективність роботи декількох функцій.

По-перше, додаток повинен безперебійно надсилати повідомлення, а також у разі сбою відновлювати надіслані повідомлення.

По-друге, додаток повинен слідкувати за надсиланням нагадувань.

По-третє, додаток повинен мати зручний та простий інтерфейс, а також коректну його роботу.

Функція	Наявність у додатку
Відправлення повідомлень	✓
Обробка повідомлень сервером	✓
Відновлення повідомлень у разі збою	✓
Повідомлення-відповіді від сервера	✓

Табл. 3.1 - Функції додатку як доставника повідомлень

Функція	Наявність у додатку
Заплановане надсилання нагадувань	✓
Редагування нагадувань	✓

Урахування часового поясу	✓

Табл 3.2 - Функції додатку як система нагадувань, прив’язана до часу

Елемент Інтерфейсу	Максимальний показник	Показник додатку
Меню	10	8
Повідомлення бота	10	9
Оформлення нагадувань	10	6

Табл 3.3 - Оцінка інтерфейсу додатку

Отже, функціонально додаток представляє із себе повноцінну систему сповіщень, що забезпечує взаємодію користувача із серверним додатком через telegram bot інтерфейс. Але естетична оцінка інтерфейсу не є досить високою, оформлення меню, нагадувань та повідомлень від бота представлені у досить неопрацьованому вигляді.

3.3 Рекомендації до підвищення ефективності додатку

Із попередніх спостережень можна сформулювати декілька факторів, що покращили б роботу додатку в цілому. Для початку можна забезпечити більшу автономність та безперебійність роботи додатку. Для цього необхідно перенести базу даних на віддалений сервер, а також задеплоїти додаток до певного сервісу, на якому слід розгорнути роботу загальнодоступного Spring Boot додатку.

Наприклад, можна використати сервіс Heroku app. Для цього необхідно зайти на сайт Heroku.com, натистнути Create New app, після чого зробити Deploy додатку, для його доступності 24/7 і коректної роботи на віддаленому сервері, а не на локальній машині.

Крім того, покращити UX - user experience можна, попрацювавши над естетичним сприйняттям інтерфейсу додатку. Необхідно зробити меню більш зручним та читабельным, повідомлення від бота більш помітними. Також корисним буде забезпечення коректного формату вводу дати сповіщення (а не у вигляді 13.08.2022 20:44). Вид самих сповіщень можна також покращити, за допомогою структурування, а також використання emoji.

Для більш широкого спектру використання, також можна забезпечити мультинаціональність додатку, та внести повідомлення на різних мовах, попередньо структурувавши команди на сервері.

ВИСНОВКИ

Месенджери набувають все більшої популярності, а години користування ними все збільшуються і збільшуються. На відміну від інших засобів, месенджери доставляють повідомлення миттєво і безперебійно.

Саме тому досить зручною і помітною є система сповіщень саме через програми миттєвого надсилання повідомлень. Продукт зможе забезпечити помітність користувачем його нагадувань, так як будь-який користувач месенджера відкриває його багато разів на день, а обов’язково помітить нагадування, що було вислане даним ботом.

Система протестована і вже може бути використана в якості системи оповіщення студентів про розклад занять. Для цього кожному студенту необхідно лише створити необхідні нагадування через меню та інтерфейс бота, і вони будуть автоматично збережені на сайті, а також надіслані у необхідний момент часу.

Крім того, через широкий функціонал, программа може бути використана для інших галузей, може бути використана індивідуальними юзерами, або як систематична система сповіщень для великої кількості людей. Система здатна підтримувати велику кількість користувачів та їхніх нагадувань, що підтверджує широкий спектр можливості її використання.

Додаток розділено на декілька частин: телеграм бота та серверного додатку. Забезпечено комунікації між підсистемами, та їх успішне виконання поставлених цілей у рамках задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kathy Sierra. Head first java – USA, c.110-200.
2. Nicolas Modrzyk, Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API, c. 200 - 250
3. Joshua Bloch, Effective java 2001 , c.50-125
4. Joshua March , Message Me: The Future of Customer Service in the Era of Social Messaging and Artificial Intelligence 2018, c. 80-155
5. Joshua Bloch, Thinking in Java 1998 c.120-170
6. Кей С. Хорстманн, Гари Корнелл (2013). Java. Библиотека профессионала, том 1. Основы. 9-е издание. «Вільямс».
7. Барри Берд (2013). Программирование на Java для чайников, 3-е издание. «Діалектика».
8. Bruce Eckel (2006). Thinking in Java (4th Edition). Prentice Hall PTR. ISBN 978-0131872486.
9. Craig Walls, Spring Boot in Action 2015, c.200-264
10. Greg L. Turnquist, Learning Spring Boot, 2017 c. 280-344
11. Matthias Biehl, Webhooks – Events for RESTful APIs, 2017
12. Кей С. Хорстманн (2014). Java SE 8. Вводный курс. «Вільямс». ISBN
13. Фрэд Лонг та ін. (2014). Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищенных программ. «Вільямс».
14. Кларенс Хо, Роб Харроп. Spring 3 для профессионалов. — М. : «Вильямс», 2012. — 880 с. .
15. Бот (або інтернет-бот) // Термінологічний словник з питань запобігання та протидії легалізації (відмиванню) доходів, одержаних злочинним шляхом, фінансуванню тероризму, фінансуванню розповсюдження зброї масового

знищення та корупції / А. Г. Чубенко, М. В. Лошицький, Д. М. Павлов, С. С. Бичкова, О. С. Юнін. — Київ : Ваіте, 2018. — С. 118.

16. Крейг Уоллс. Spring в действии. — Третє. — М. : «Manning», 2014. — 624 с.