МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Програмний додаток підтримки планування заходів в дитячих колективах»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-81 Печериця Владислав Сергійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2019 р.

Науковий керівник _____ к.т.н., Марченко А.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2019 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Печериця Владислав Сергійович
(прізвище, ім'я, по батькові)

1 Тема проекту *Програмний додаток підтримки планування заходів в дитячих колективах*

затверджена наказом по університету від «19» листопада 2019 р. №2305-III

2 Термін здачі студентом закінченого проекту «_10__» ___грудня___ 2019 р.

3 Вхідні дані до проекту _____

_____

_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) _____

_____

_____

_____

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

_____

_____

_____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка

Магістрант _____ Печериця В.С.

Керівник роботи _____ к.т.н., Марченко А.В.

РЕФЕРАТ

до дипломного проекту

Печериці Владислава Сергійовича на тему: «Програмний додаток підтримки планування заходів в дитячих колективах»

Дипломний проект розроблений для надання можливості користувачеві раціонально розподілити завдання в дитячих колективах, а також надати необхідну мотивацію до роботи, що сприятиме ефективному виконанню тієї чи іншої задачі.

У роботі проведено аналіз області та аналогів, сплановано етапи роботи, ризики та варіанти їх запобігання, проведено моделювання роботи системи та веб-сайту.

Засоби, за допомогою яких було виконано розробку: MySQL, Vue.JS, Node.JS, Bootstrap, CSS, HTML, Git, JavaScript.

Результатом роботи є система підтримки планування заходів в дитячих колективах у вигляді веб-сайту з інтеграцією телеграм-боту.

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списка використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи складає 81 сторінку, в тому числі 62 сторінки основного тексту, 3 сторінки використаних джерел та 21 сторінок додатків.

Загальний обсяг роботи 81 сторінка, 46 рисунків, 5 таблицm, 2 додатки, 30 бібліографічних найменувань.

Ключові слова: інформаційна система, Node, Vue, планування, заходи, дитячі колективи, телеграм-бот.

ЗМІСТ

Вступ 6

1 Аналіз предметної області 7

1.1 Аналіз ресурсів аналогічного функціонального призначення 7

1.2 Дослідження методів розробки 18

2 Постановка задачі та методи дослідження 21

2.1 Мета дослідження 21

2.2 Вибір методу дослідження 22

3 Проектування програмного додатку підтримки планування заходів в дитячих колективах 25

3.1 Архітектура системи 25

3.2 Моделювання процесу роботи телеграм-боту 26

3.3 Моделювання бази даних 34

3.4 Засоби дослідження 39

4 Розробка інформаційної системи 41

4.1 Серверна реалізація 41

4.2 Налаштування проекту 44

4.3 Реалізація бази даних 49

4.4 Розробка API 52

4.5 Створення клієнтської частини додатку 55

Висновки 59

Список використаних джерел 60

Додаток А. Планування робіт 63

Додаток Б. Лістинг програмного продукту 68

ВСТУП

Актуальність: онлайн месенджери стрімко набирають популярність у наш час. Окрім звичайних чатів, існує велика кількість різноманітних публічних сторінок, групових чатів, а також чат-ботів. Тому, зараз як не як актуальна розробка таких ботів.

Об'єкт дослідження: чат-боти та інформаційні системи які були змодельовані для надання користувачеві можливості відслідкування та нагадування про його події чи про дні народження знайомих.

Предмет дослідження: розробка проекту чат-боту.

Мета: розробити інформаційну систему та чат-бот на основі проведеного дослідження предметної області.

Практичне значення: дозволяє користувачеві із легкістю відслідкувати всі необхідні для нього події, нагадування про які будуть приходити в месенджер.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ресурсів аналогічного функціонального призначення

Станом на сьогодні телеграм-боти являються популярним інструментом для зв’язку між користувачами відомого месенджера. Боти не тільки надають можливість швидко дістатися до згрупованих функцій сервіса, а й отримати миттєву автоматичну відповідь на свій запит, що суттєво економить час на відміну від онлайн-сервісів, які часто містить черги користувачів при обробці наприклад текстових запитів. [1]

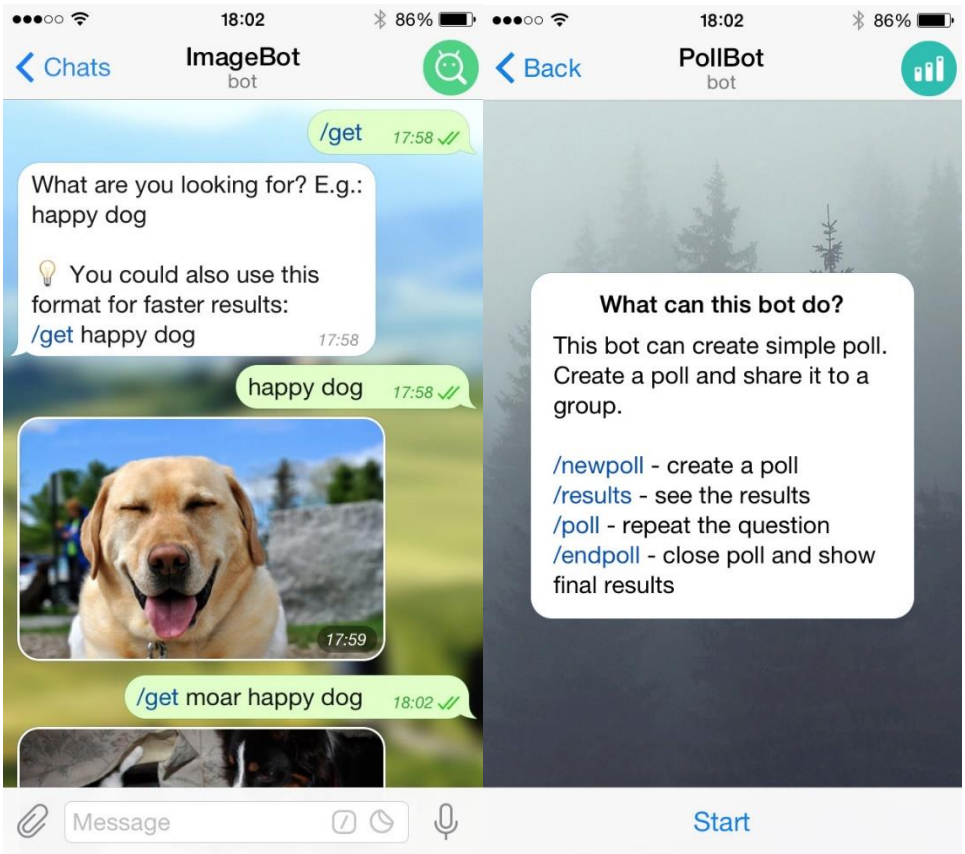

Рисунок 1.1 – Приклади чат-ботів

Боти стають з кожним днем популярнішими через те, що їх почали активно використовувати в соціальних мережах та месенджерах. Функціонал дуже широкий, та може надавати доступ до різноманітних сервісів. Наприклад рекламна розсилка статей та актуальних пропозицій, миттєва обробка даних, онлайн спілкування з автоматичними відповідями.[2,3]

Більшість компаній, спостерігаю позитивне ставлення аудиторії до таких нових сервісів, тому починають їх активно застосовувати для своїх цільових дій. Боти вже починають активно рекламуватися як прості сервіси навіть від великих брендових компаній. Для зовсім малих компаній, які в більшості працюють з оф-лайн аудиторією такі функції не є актуальними. Проте через високий рівень використання їх починають використовувати для розсилання спам-інформації цільовій аудиторії. Це також може впливати на знання аудиторії про такі сервіси.[4]

Активне використання чат-ботів як програм наділених значущими технічними і функціональними параметрами почалося недавно. До основних трендів в інтернет-маркетингу їх прийнято відносити до 2017 року.

Такі сервіси зараз здійснюють заміну процесу спілкування з реальними адміністраторами сервісів, вам можуть швидко та автоматично надати відповіді про курси валют, записи на певні онлайн та оф-лайн події, нагадування, та пошук інформації за категоріями.

Початково такі сервіси створювалися для того, щоб надавати швидкі відповіді користувачам на певний ряд питань, але і зараз вони використовують такий підхід, що суттєво економить час на чекання відповідей від реальних працівників. [5-7]

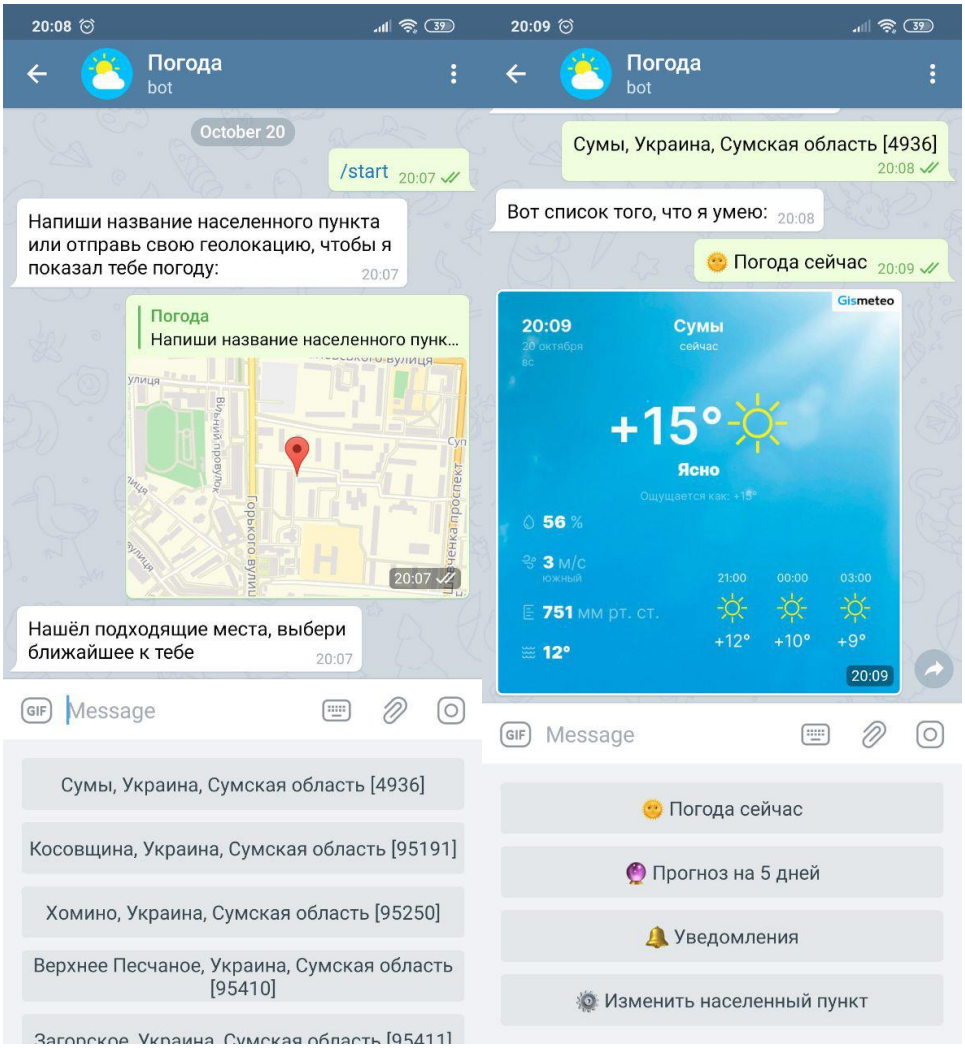

Рисунок 1.2 – Приклад боту який повідомляє погоду

В основі телеграм-ботів лежать сучасні технології, які містять інформаційно технології для комунікації, щоб застосовувати їх для розв’язання комерційних та маркетингових задач.[8,9]

Отже, чат-боти здатні вирішувати багато задачі, пов'язані із здійсненням комунікацій з клієнтами, спрямованих на можливість залучення нової аудиторію, просування самої реклами, надання їм інформаційних послуг, стимулювання продажів, формування у них лояльності до компанії, залучення споживача в рекламну комунікацію, що сприяє підвищенню її ефективності.[10-13]

Було проаналізовано такі інтернет-ресурси, які були створені для того, щоб надавати користувачеві можливість планувати будь-які завдання, або ж встановлювати нагадування для себе, наприклад нагадування про дні народження:

1. Сайт « http://nezabit.ru/ »

Перший ресурс який був проаналізований - це сайт “не забить” (рис 1.3).

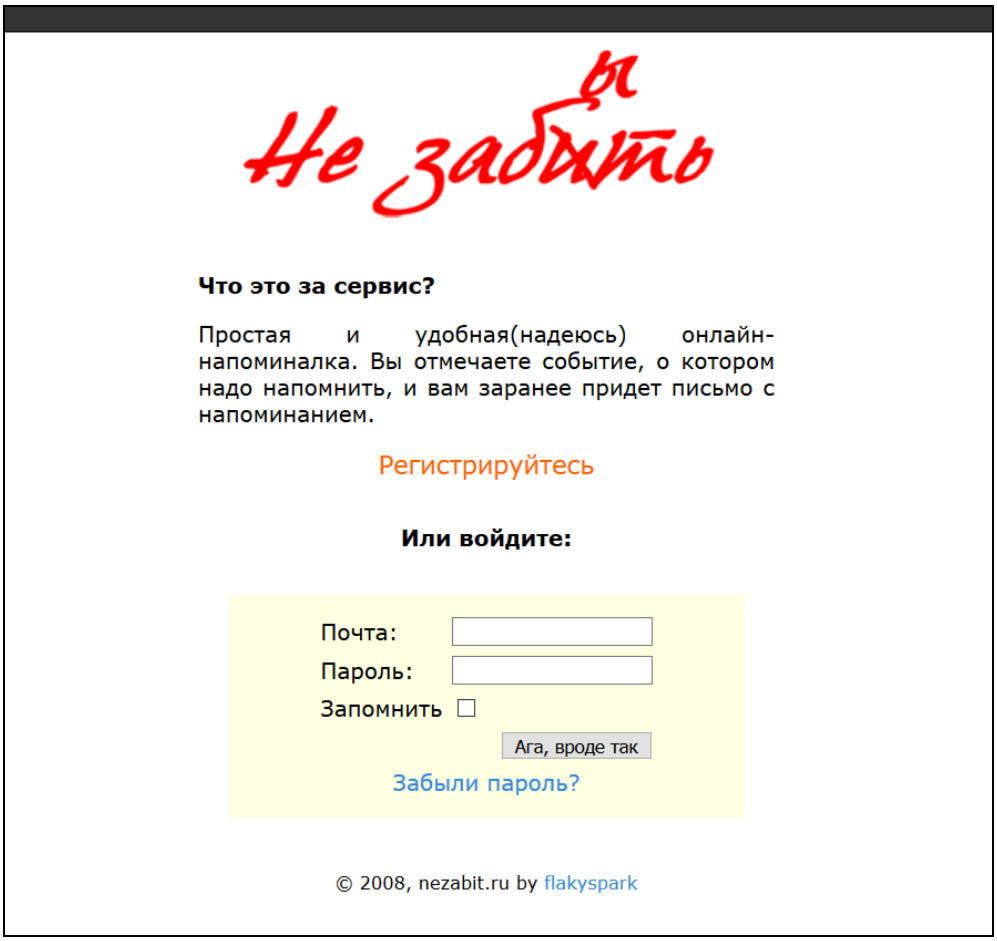

Рисунок 1.3 – Головна сторінка сайту «не забить»

Цей сайт 2008 року який повинен відправляти повідомлення користувача йому ж на електронну пошту. Хочу зауважити те, що на даний момент електронну пошту перевіряють далеко не всі користувачі інтернету. Провівши опитування потенційних користувачів, які являються моєю цільовою аудиторію, вияснилося

те, що майже всім було б не зовсім зручно отримувати нагадування на свою електронну пошту.[15,16]

Даний інтернет-ресурс пропонує користувачеві зареєструватися там і написати собі нагадування (рис 1.4).

Рисунок 1.4 – Сторінка створення нагадування сайту «не забить»

Після реєстрації потенційний користувач вписує нагадування і відповідне поля, після чого очікується надсилання повідомлення на електронну пошту, але в необхідний час, повідомлення так і не надійшло на пошту, що скоріш за все, значить те, що даний сайт не підтримується розробником та має не працюючий функціонал.[17-20]

2. Телеграм-бот DeLorean

Також, під час аналізу конкурентного ринку було виявлено що існує прототип чат-боту «DeLorean», який виконує функцію повідомлення користувачеві про подію яку він додав до боту заздалегідь.

Даний бот має функції додавання нагадування, налаштування, відображення списку нагадувань а також відображення довідки користувача.

Під час дослідження його роботи, було виявлено те, що основний функціонал не коректо працює (рис 1.5).

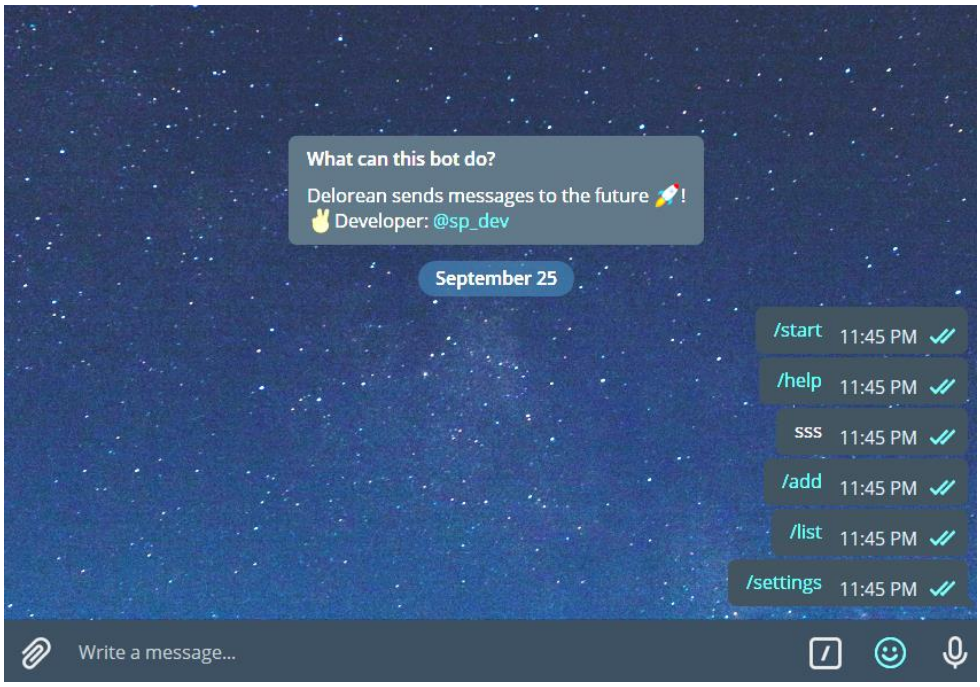

Рисунок 1.5 – Перевірка функціоналу телеграм-боту

3. Google calendar та Business calendar 2

Наступним кроком аналізу предметної області, було виявлено найбільш ефективний інтернет ресурс який відповідає встановленим критеріям: «Google calendar» - онлайн додаток в якому можна додати подію, чи нагадування про день народження і вона буде відображатися в додатку. Але проблема в тому, що по-перше, не всі користувачі гугл-календарю кожного дня заходять в нього і є велика ймовірність того, що користувач пропустить важливу дату чи день народження, по-друге якщо у вас в гугл-контактах немає користувача якого б ви хотіли привітати, то в календарі вона не відобразиться, а можливість додати користувача вручну на даний момент не реалізована.[21]

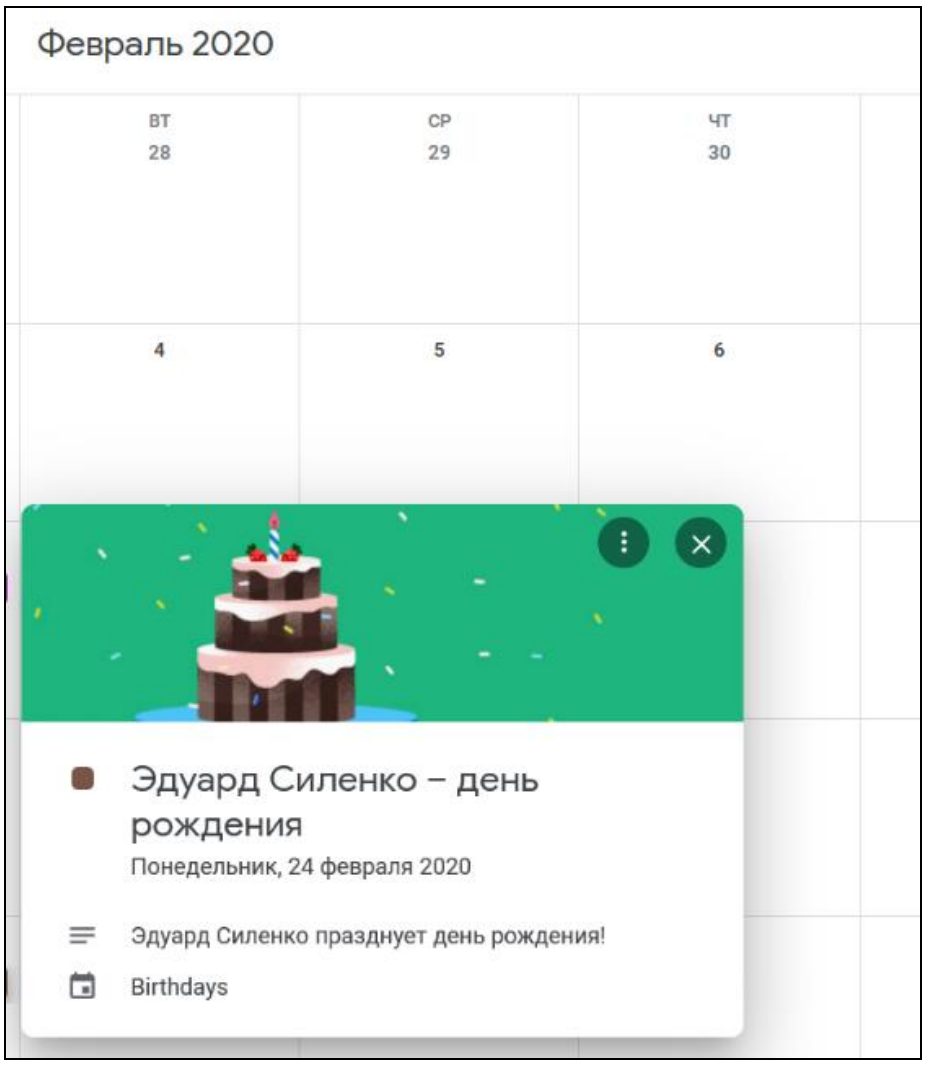

Рисунок 1.6 – Приклад відображення події в Google calendar

Також було проаналізовано те, що існує мобільний додаток «business calendar 2» в якому можна синхронізувати дані із гугл акаунту, а також додавати свої події, календарі і налаштовувати нагадування.

Крім цього було звернуто увагу на вибір категорій. Даний додаток дозволяє користувачеві створювати не тільки нагадування про день народження, а й про річниці, одруження, тощо (рис 1.7). Можливість поділяти події чи завдання на категорії – зменшує час пошуку інформації в подібних додатках.

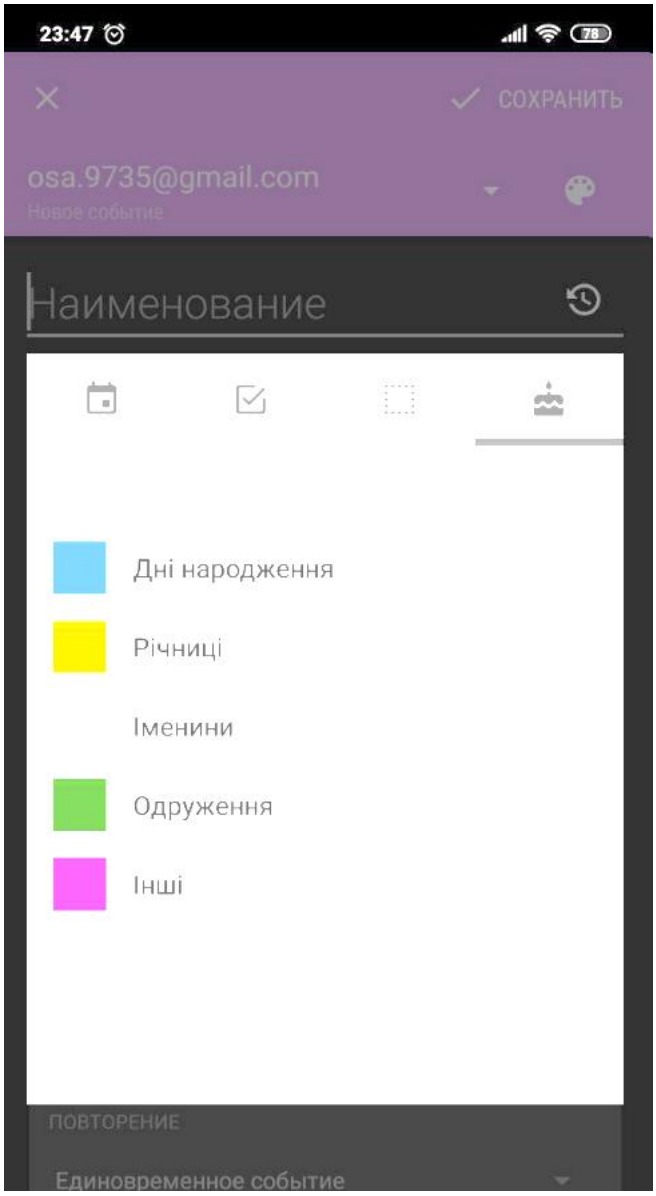

Рисунок 1.7 – Категорії подій в додатку business calendar

4. PlayMarket

Мобільний ресурс Play Market призначений для того щоб користувачі мобільних телефонів із операційною системою «Android» могли завантажити той чи інший додаток на свій пристрій. Ці додатки поділяються на платні та безкоштовні.

В Play Market снує багато мобільних додатків які виконують функцію нагадування користувачеві про важливі події або ж про задачі які потрібно виконати (рис 1.8).

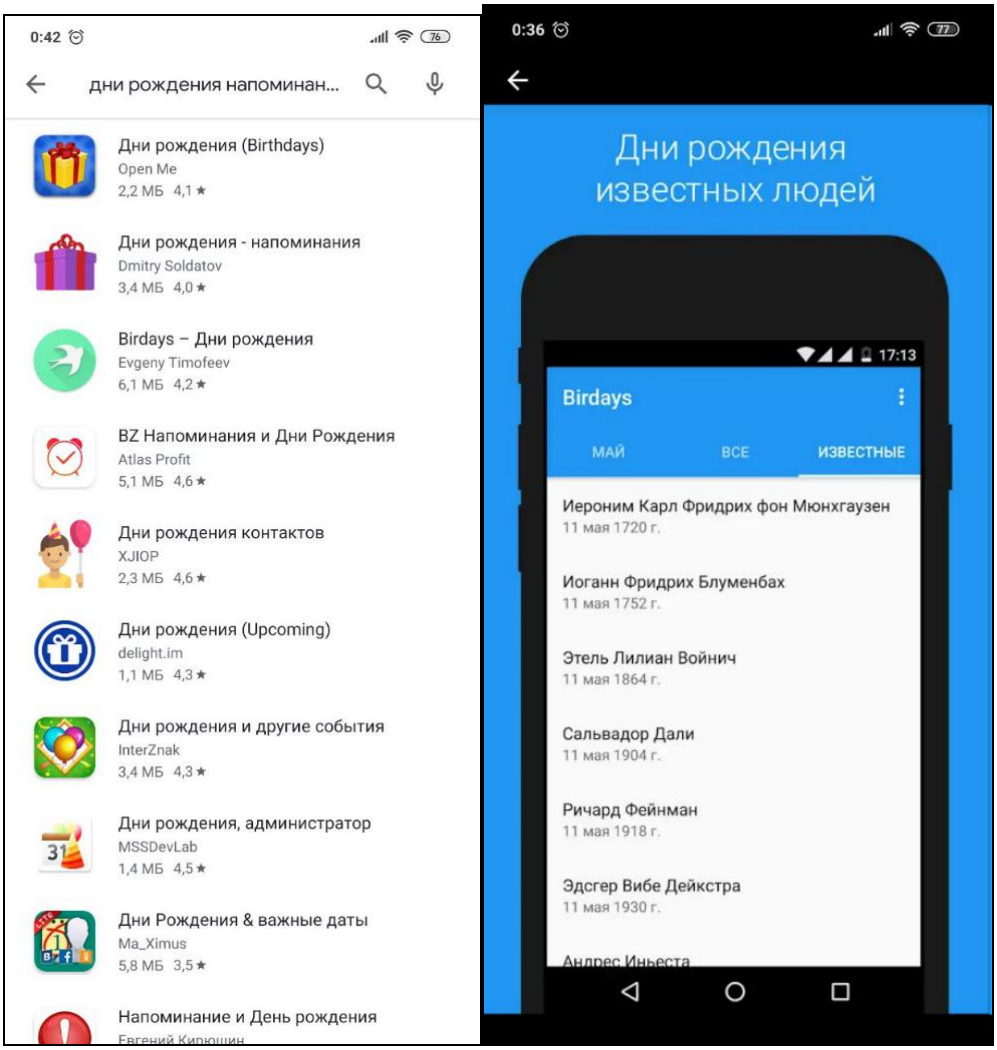

Рисунок 1.8 – Приклади мобільних додатків

Недолік мобільних додатків полягає в тому, що для таких простих речей необхідно ставити окрему програму на телефон. Тому, якщо б всі дані функції знаходилося в одному місці, наприклад, в популярному месенджері Telegram, то було б набагато зручніше та рентабельніше.[22,23]

Мобільний додаток «Facebook», також має функцію надсилання пуш-повідомлення (пуш-повідомлення, це те яке використовує так звану push-технологію, яка являє собою інструмент розсилки від постачальника послуг до споживача) про події чи дні народження, але події можуть бути не актуальні чи не цікаві для користувача (рис 1.9). І навпаки: не всі необхідні для користувача події будуть надсилатися.

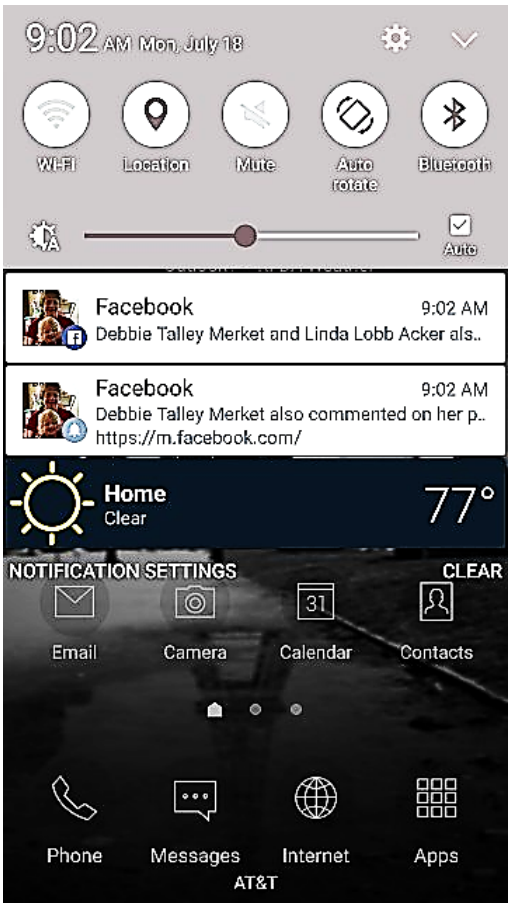

Рисунок 1.9 – Приклад push-повідомлення мобільного додатку

Отже, в результаті проведеної роботи було виявлено те, що додаток повинен сповіщувати користувача шляхом пуш-повідомлень в додатку Telegram, на відміну від веб-сервісу «nezabit.ru». Також додатку необхідно мати зрозумілий інтерфейс, та наявність робочих функцій на відміну від телеграм-боту DeLorean. Крім цього

потрібно реалізувати додавання нових подій просто та зручно як у додатку Google Calendar.

Фінальний продукт має виконувати розсилки користувачам, мати можливість надсилати нагадування, в тому числі через push-повідомлення. Крім цього необхідно реалізувати об’єднання користувачів у групи і головний критерій – щоб додаток був працездатним.

Було створено порівняльну таблицю для того щоб визначитися із специфікою роботи розроблюваного додатку.

Таблиця 1.1 – Функціонал аналогів

	nezabit.ru	Телеграм-бот DeLorean	Google calendar	PlayMarket
Створення нагадувань	+			
Об'єднання користувачів у групи			+	
Push-повідомлення		+	+	+
Інтеграція з месенджерами		+		
Виконання розсилок користувачам			+	+
Зрозумілий інтерфейс користувача			+	+

Продовження таблиці 1.1

Працездатність			+	+
Робота в оф-лайн режимі			+	
Повністю безкоштовний	+	+	+	

1.2 Дослідження методів розробки

Для того щоб розпочати розробку телеграм-бота, необхідно зареєструвати його в системі телеграм. Щоб це зробити, необхідно в додатку «Telegram» знайти системного бота «@BotFather» і надалі слідувати його інструкціям (рис 1.10).

Єдине обмеження при створенні нового бота це на ім'я - воно повинно закінчуватися на «bot». У разі успіху BotFather повертає токен бота і посилання для швидкого додавання бота в контакти.[24]

Наступний крок: обрати мову програмування для бота.

Всі боти в додатку Telegram написані на власному API (програмний інтерфейс програми який описує спосіб яким один додаток взаємодіє із іншим. В ньому міститься набір різних класів, констанкт, функцій, тощо), який може взаємодіяти із майже, будь-якими, серверними мовами програмування які підтримуються на процесорі типу «Corezoid», такі як C#, node.js, PHP.[25]

В зв’язку з тим що node.js та PHP відносно просто реалізувати на хостинг сервісах, то обирати прийшлося між цими двома варіантами. Так як Node.js має прекрасний інструмент, npm (менеджер пакетів, с його допомогою можна керувати модулями і залежностями), то остаточний вибір лишився на ньому.

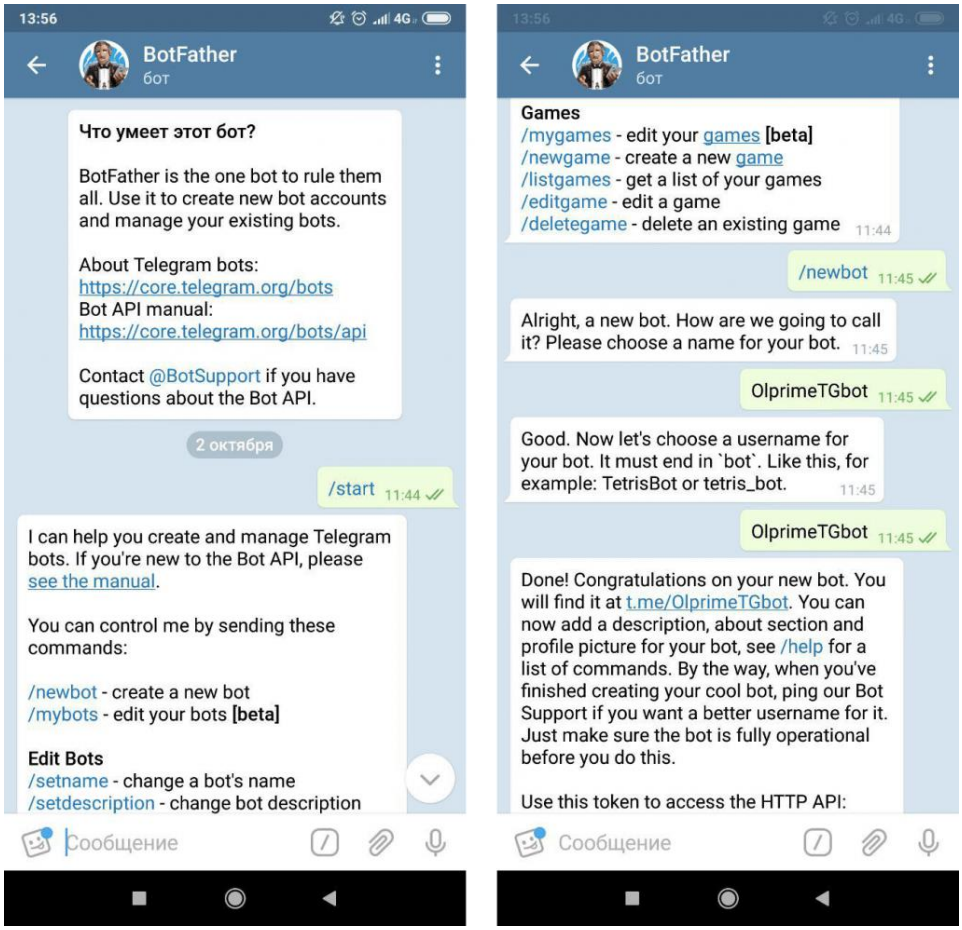

Рисунок 1.10 – Чат із ботом «BotFather»

Так як в якості серверного додатку був обраний node.js, то клієнтську частину можна обрати серед Angular, Vue.js, React (рис 1.11).

AngularJS - це структура JavaScript, розроблена Google. Вона спрямована на розробку веб-додатків, надає великого значення структурі та якості. Це був перший каркас, який також був придатний для великих корпоративних додатків через фокус на архітектурі, тестуванні та ізольованих компонентах в області JavaScript. Використовуючи такі методи, як ін'єкція залежностей та складний інструментарій, це дозволяє ефективно та бездоганно розробляти програмне забезпечення на основі JavaScript.

Vue.js є більш гнучким, менш виразним рішенням (ніж Angular). Це дозволяє структурувати додаток таким, яким ви його хочете бачити, а не змушувати робити все шляхом Angular. З іншого боку, Vue.js набагато простіший, ніж Angular.[26]

Фреймворки Vue.js та React пропонує кращі показники продуктивності та гнучкості, ніж Angular.

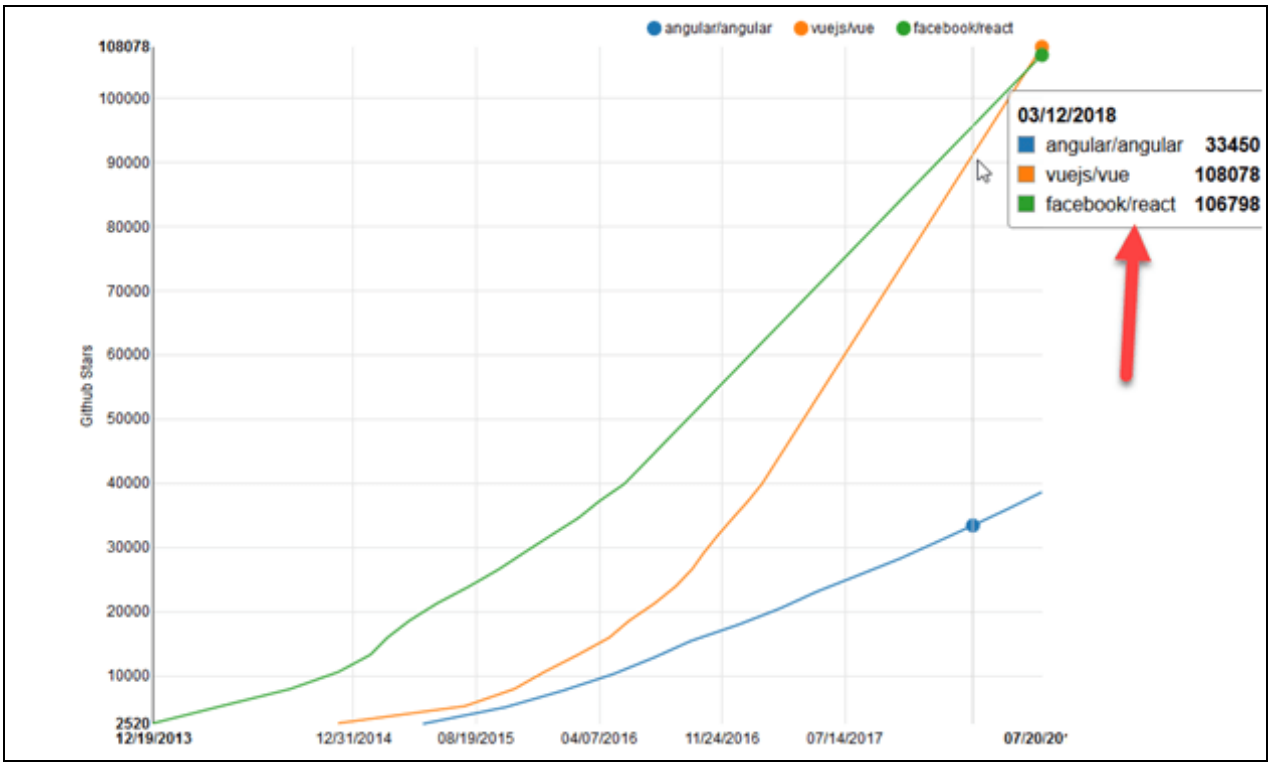

Рисунок 1.11 – Графік популярності фреймворків

Вони більше підходять для додатків з простим функціоналом, а Angular - найкращий для програм з складними функціями для більш досвідчених розробників. Має розширений інтерфейс від маршрутизації, шаблонів до тестування утиліт у своєму пакеті. Vue – найпопулярніший і зростаючий фреймворк на мові програмування Javascript. Для реалізації мого додатку ми будемо використовувати фреймворк Vue.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета дослідження

Метою виконання цієї роботи є розробка інформаційної системи та чат-боту для підтримки планування заходів в дитячих колективах, яка буде містити функціонал для виконання таких задач:

- реєструвати користувача;
- надавати можливість користувачеві редагувати персональні дані;
- надавати користувачеві функції налаштування акаунту;
- створення власних подій чи нагадувань;
- редагування та видалення створених подій чи нагадувань;
- надавати користувачеві можливість об’єднання інших користувачів у групи;
- додавання завдання у користувацькі групи;
- перегляд подій, які запланує користувач та можливість їх подальшого редагування;

Для реалізації виділених завдань ми перед собою ставимо такі задачі:

- Проаналізувати аналоги;
- Створити додаток;
- Розробити інтерфейс для взаємодії з користувачем;
- Провести розробку основних модулів системи;
- Налаштування зв’язку з базою даних;
- Реалізація класу створення запитів в базу даних;

- Виконати розробку ядра додатку;
- Інтегрувати додаток із телеграм-ботом;
- Створити необхідні маршрути додатку;
- Розробити сесійний модуль додатку;
- Протестувати створений додаток.

2.2 Вибір методу дослідження

Через тотальне заповнення технологіями нашого повсякденного життя, та постійним удосконаленням технологій все ще існує проблема ку правилах для розробки інформаційної системи (ІС)

ІС – це вимогливі програмні додатки, в яких містяться певні групи з прикладними програмами, вони надають можливості в системі для надання розподілених прав та переваг, їх часто використовують задля створення унікального доступу до не публічної інформації користувацьких груп.

Метою їх реалізації є надання впевненості в безпеці персональних даних, безпомилковою роботою з даними та легкість в навчанні для користування.

Сьогодні все частіше розробники використовують зручний фреймворк Bootstrap. Ця система надає можливість для взаємодії розробників та дизайнерів для створення якісних та привабливих інтерфейсів користувача. Розробники використовують шаблони CSS і HTML для створення різноманітних структурних елементів веб сайтів. Це можуть бути кнопки, вспливаючі вікна, форми, лічильники, таблиці. І саме використання Bootstrap уніфікує роботу дизайнерів з усього світу та дозволяє створювати інтерфейси які враховують рівень usability для користувача.[27-29]

З допомогою модулів JavaScript є можливість створення анімаційної складової інтерфейсу. Їх використання за дослідженнями [12] викликає затримання зацікавленості користувача, що сприяє збільшенню часу, який користувач знаходиться на сайті.

Також Bootstrap дозволяє створювати інтерфейси, які з часом можуть бути легко адаптовані під різні розміри екранів, для безпомилкового відображення.

Наш проект буде реалізовано у вигляді веб-додат0ку та телеграм-бота, який створюватиметься на Node JS.[30]

Для реалізації запланованого проекта ми використаємо:

- MySQL(мову для створення запиту до БД);
- JavaScript;
- Мова гіпертекстової розмітки HTML;
- Мова опису зовнішнього виду веб-сторінки CSS;
- Інтегроване середовище розробки Web Storm 2018.1.4;

Бібліотеки які використовуватимуться в проекті:

1) «mysql»

Бібліотека для платформи Node.js, створена для того, щоб виконувати операції із базами даних, які написані на мові MySQL.

2) «node-telegram-bot-api»

API месенджера телеграм. Дозволяє виконати інтеграцію мі ботом та створюваним додатком

3) «vue»

Бібліотека, призначена для інтеграції фреймворка Vue.js із платфромою Node.

4) «vue-resource»

Розширення яке позволяє використовувати Node.js модулі на сторінках клієнтської частини додатку на Vue.js.

5) «vue-router»

Розширення, що дозволяє динамічний перехід між веб-сторінками додатку.

6) «express»

Бібліотека, що надає змогу серверу створювати маршрути для додатку, а таж використовувати їх в якості запитів на API.

7) «express-session»

Бібліотека, яка надає можливість створювати сесії, та додавати cookie-файли для розроблюваного веб-додатку.

8) «debug»

Бібліотека допомагає користувачеві ефективно робити відладку коду.

9) «cookie-parser»

Бібліотека створена для того, щоб зчитувати cookie-файли на стороні API.

10) «body-parser»

Бібліотека створена для того, щоб зчитувати POST/GET/PUT запити на стороні API.

11) «bootstrap»

Розширення для взаємодії з фреймворком «bootstrap»

12) «node-schedule»

Біблотека яка реалізовує можливість створення відкладених подій без великого навантаження на сервер.

13) «generate-password»

Дана бібліотека призначена для генерації паролів, обраної довжини.

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ПІДТРИМКИ ПЛАНУВАННЯ ЗАХОДІВ В ДИТЯЧИХ КОЛЕКТИВАХ

3.1 Архітектура системи

Фізичний сервер, де ми розмістимо групу програмних модулів нашої ІС, а також база даних буде розміщуватися на безкоштовних хостингових сервісах.

Зображення архітектури ІС.(рис. 3.1) Для взаємодії зі створеною ІС користувачеві потрібно перейти у веб-інтерфейс або відкрити спеціального телеграм-бота та надіслати йому запит по натисканню на кнопку. Наступним кроком веб-інтерфейс або телеграм бот зв’язується з API частиною додатку, після цього надсилається з API запит на сервер, а потім з сервера до БД. У відповідь сервер отримує результат та повертає його до API. І потім іде відображення згенерованої інформації на екран користувача.

Склад архітектури системи:

• Користувачі

• Веб-інтерфейс

• Бот

• API додатку

• База даних

Дана архітектура являється трьохрівневою, вона передбачає в собі три компоненти: клієнт, сервер додатків (до якого підключена клієнтська програма) та сервер бази даних (використовується сервером додатків). А у нашому випадку, до серверу додатків підключений ще телеграм-бот.[10]

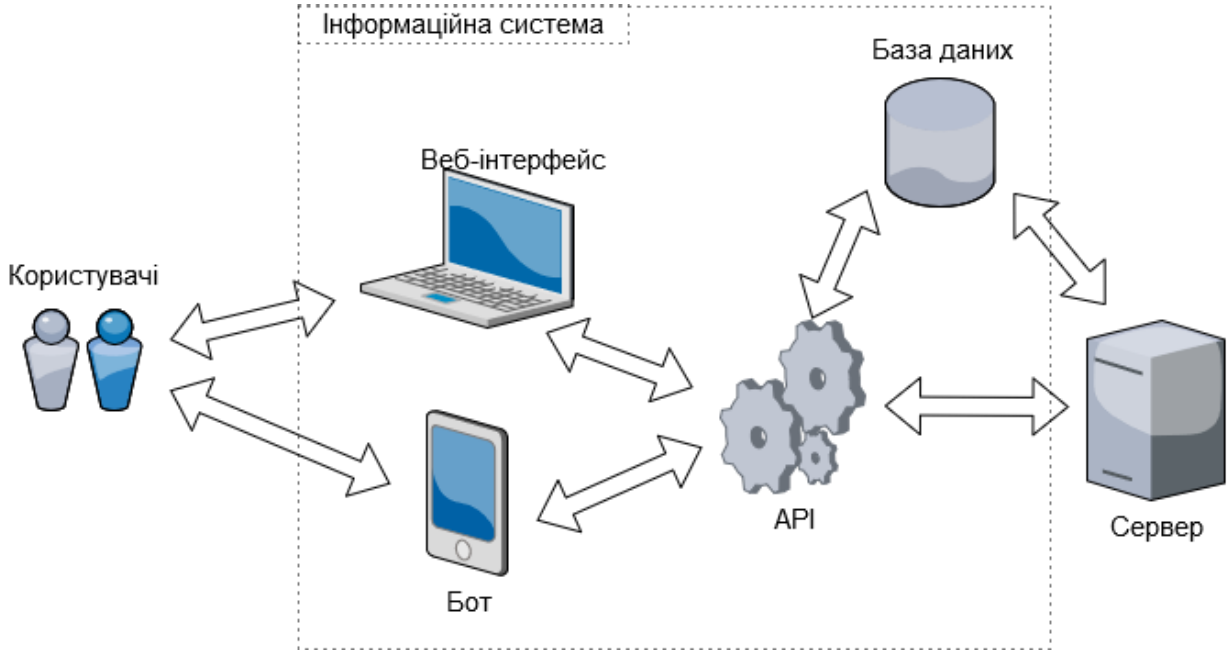

Рисунок 3.1 – Архітектура системи

3.2 Моделювання процесу роботи телеграм-боту

Для того щоб зобразити основні функції мого додатку було змодельовано UML-діаграму (Рис. 3.2).

UML - Уніфікована мова моделювання - це система позначень, яка може використовуватися для аналізу та проектування.[11]

Ці діаграми можна використовувати для візуалізації, специфікації, проектування та документації програмних систем.

Було виділено 15 основних функцій додатку та акторів які будуть їх виконувати.

Дана діаграма має трьох акторів:

1) Користувач.
2) База даних.
3) Телеграм-бот

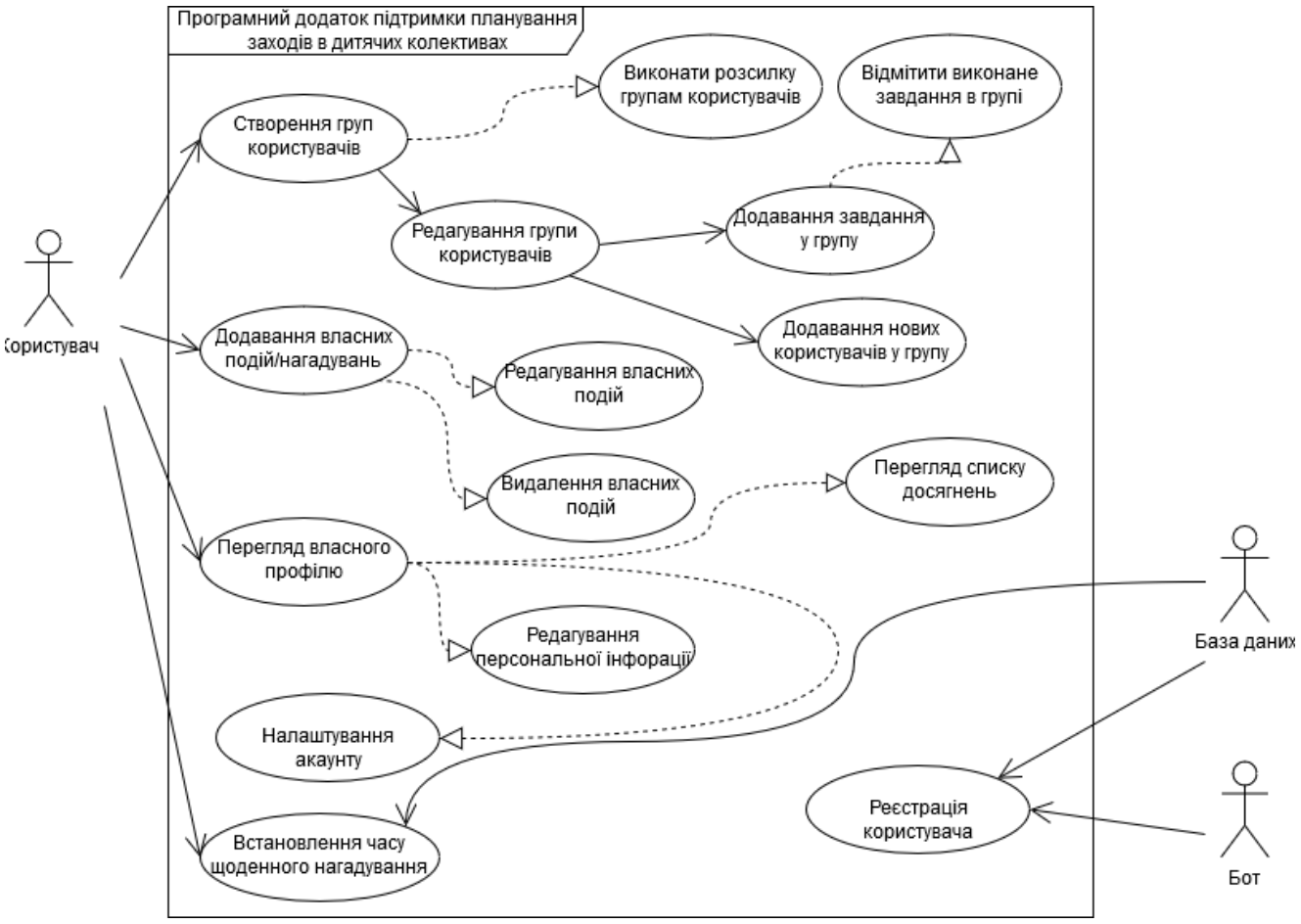

Рисунок 3.2 – UML-діаграма системи

Актор «База даних» та актор «Бот» виконують реєстрацію користувача автоматично. Відбувається це при першому запуску бота, а саме коли користувач натисне на стандартну кнопку «/start», що і активує бота.

Також, актор «База даних» встановить час нагадування на значення за замовчуванням, але користувач зможе в майбутньому змінити це значення.

У актора «Користувач» будуть доступні такі функції:

Таблиця 3.1 – функції актора «Користувач»

Функція	**Опис функції**
Перегляд власного профілю	Надається можливість переглянути інформацію про профіль користувача, а саме його ім’я, прізвище чи нікнейм в телеграмі.
Редагування персональної інформації	Надається можливіть користувачеві змінити свої персональні дані, такі як прізвище чи ім’я.
Налаштування акаунту	У кожного користувача є свій логін та пароль, дана функція дозволяє редагувати ці дані.
Перегляд списку досягнень	Функція повертає список досягнень користувача.
Встановлення часу щоденного нагадування	Кожного ранку користувачеві надсилатиметься повідомлення в телеграм, і дана функція дозволяє відредагувати час отримання повідомлення.
Додавання власних подій/нагадувань	Функція дозволяє користувачеві створювати нагадування із конкретним типом.

Продовження таблиці 3.1

Редагування власних подій	Внести зміни в, раніше створені користувачем, події.
Видалення власних подій	Видалити з бази даних, раніше створені користувачем, події.
Створення груп користувачів	Об’єднання користувачів в групи, в яких можна створювати задачі та назначити їх користувачам.
Виконати розсилку групам користувачів	Відправити повідомлення в телеграм користувачам групи.
Додавання завдання у групу	Створити завдання для користувачів певної групи.
Додавання нових користувачів у групу	Після створення групи, у її власника є можливість запросити до неї користувачів.
Відмітити виконане завдання в групі	Якщо користувач, який належить групі, виконав завдання, то власник групи може зазначити завдання як виконане, після чого користувачеві який виконав завдання більше не будуть надходити нагадування про виконання того завдання.

Для того щоб показати процеси роботи програми було створено 3 рівні діаграми IDEF0.[12]

На рисунку 3.3 зображений перший рівень діаграми. На вході ми отримуємо необхідність користувача у плануванні подій. Вказівки користувача, а також

технічна документація слугує як управління програмним додатком підтримки планування заходів. Механізмом являється: апаратне забезпечення сервера, сам телеграм-бот та інтернет. А на виході ми отримуємо персональні нагадування про події.

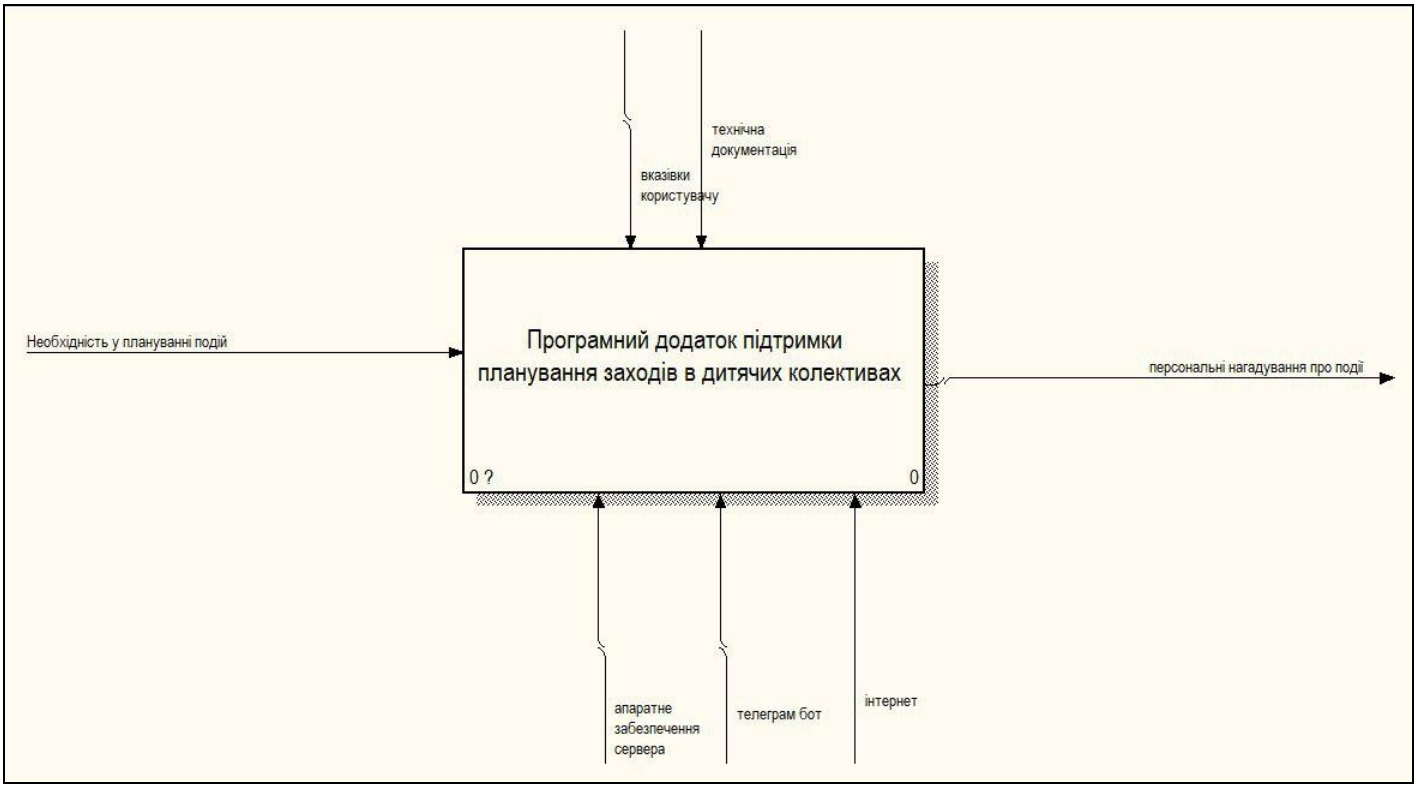

Рисунок 3.3 – Перший рівень діаграми IDEF0

Після побудови першого рівня діаграми, було проведено його декомпозицію, яка була розбита на три блоки:

1) Реєстрація користувача у телеграм-боті
2) Додавання події
3) Налаштування сповіщень

Стрілка управління «вказівки користувача» направляються у всі три блоки декомпозиції, а стрілка «технічна документація» входить лише у блок «додавання події» та в «налаштування сповіщень».

Телеграм-бот, апаратне забезпечення та інтернет приймає участь у всіх трьох блоках.

На вході ми маємо необхідність у планування подій, яка входить у блок «Реєстрація користувача у телеграм-боті» із якого на виході ми отримуємо персональний номер користувача, який в свою чергу входить у блок «Додавання події», з якого на виході отримуємо дані в базі даних, які входять у блок «Налаштування сповіщення» із якого на виході отримуємо персональні нагадування про події. Дана декомпозиція першого блоку зображена на (рис. 3.4).

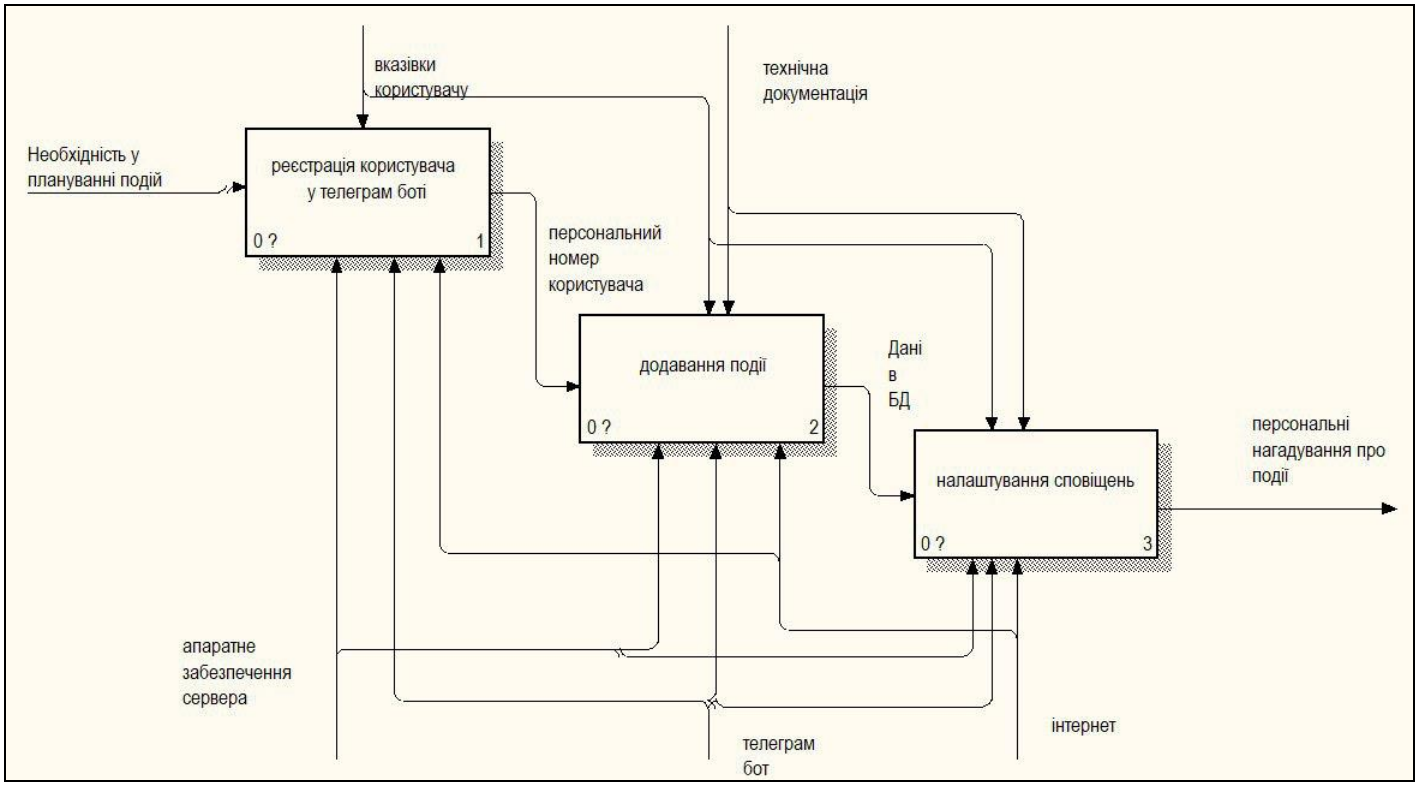

Рисунок 3.4 – Декомпозиція першого рівня діаграми IDEF0

Далі проводиться декомпозиція блоку «Реєстрація користувача» (рис. 3.5). Розділяється на 2 блоки «Вхід в телеграм» та «Початок роботи через кнопку /start». На вході ми маємо необхідність у плануванні подій, а після входу в телеграм у нас появляється можливість запустити бота. Після чого входимо у блок

«Початок роботи через кнопку /start», на виході якого отримуємо персональний номер користувача.

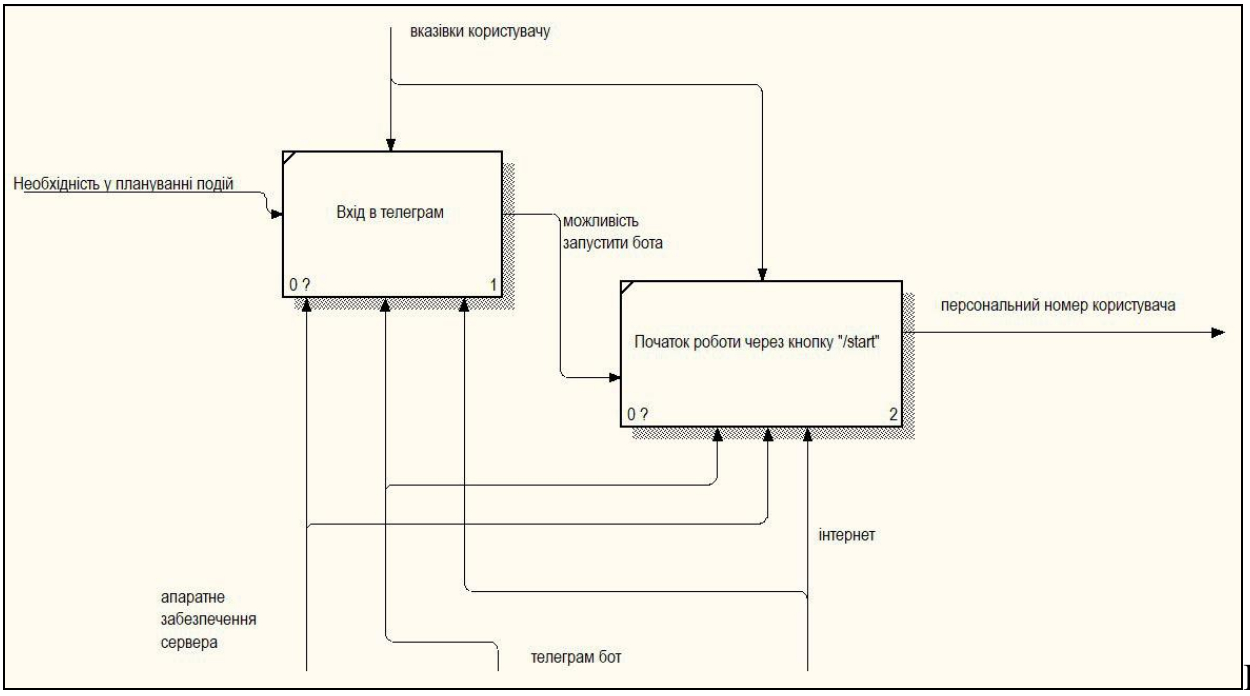

Рисунок 3.5 – Декомпозиція блоку «Реєстрація користувача»

Наступним кроком декомпозуємо блок «Додавання події» (рис. 3.6).

Декомпозується на 3 блоки:

1) Вхід в чат бота
2) Натиск кнопки /add
3) Введення даних події

На вході ми маємо персональний номер користувача, а після входу в чат бот у нас появляється можливість створювати нові події. Після чого входимо у блок «Натиск кнопки /add», на виході якого можна отримати інтерфейс створення події. Потім вводимо дані події і на виході блока «Введення даних події» ми отримуємо дані в базі даних.

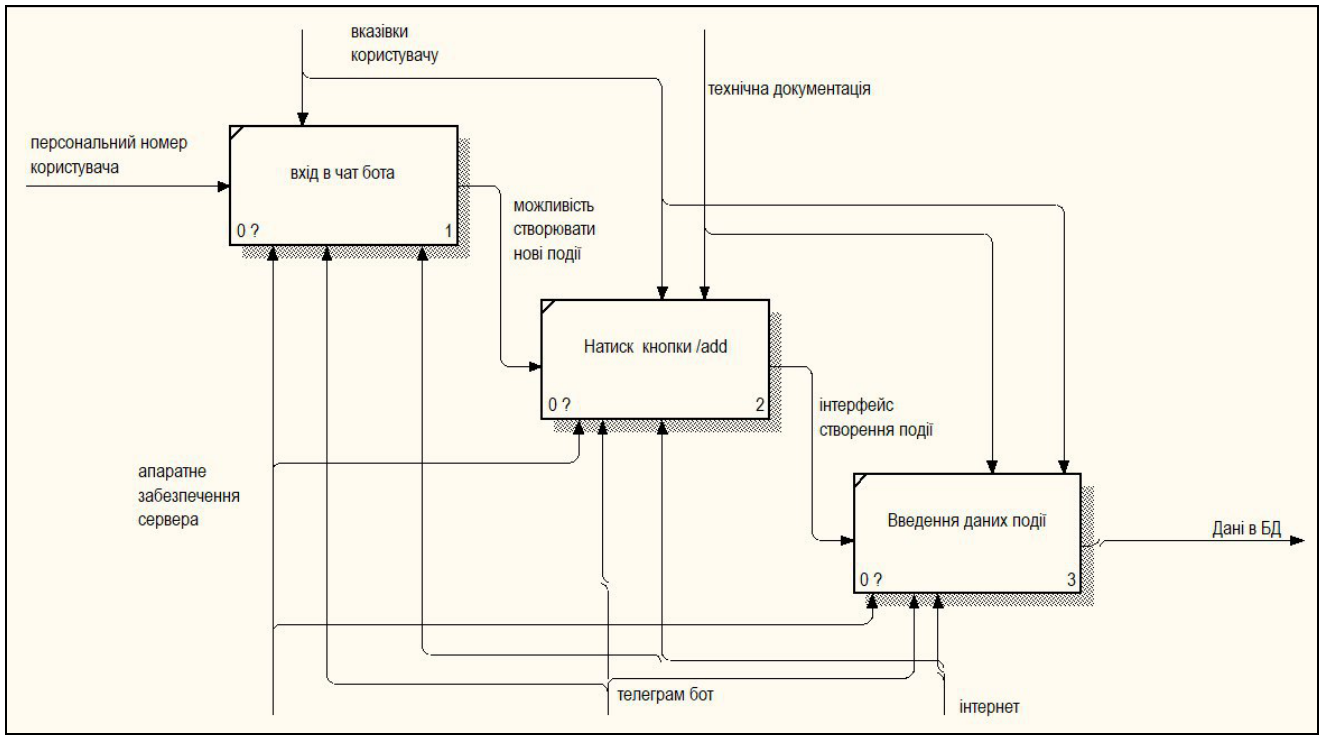

Рисунок 3.6 – Декомпозиція блоку «Додавання події»

Наступним кроком декомпозуємо блок «Налаштування сповіщень» (рис. 3.7). Декомпозується в блок «Вибір існуючого сповіщення», а також в «Редагування обраного сповіщення».

На вході ми маємо данні з бази даних, а після входу в блок «Вибір існуючого сповіщення» ми отримуємо унікальний номер сповіщення, після чого входимо в блок «Редагування обраного сповіщення», а на виході з цього блоку отримуємо персональні нагадування про подію.

Завдяки побудованим моделям, ми чітко визначили процеси роботи які плануються у розроблюваному додатку, а UML діаграми дозволила визначити всі функції додатку які плануються реалізовувати в проекті.

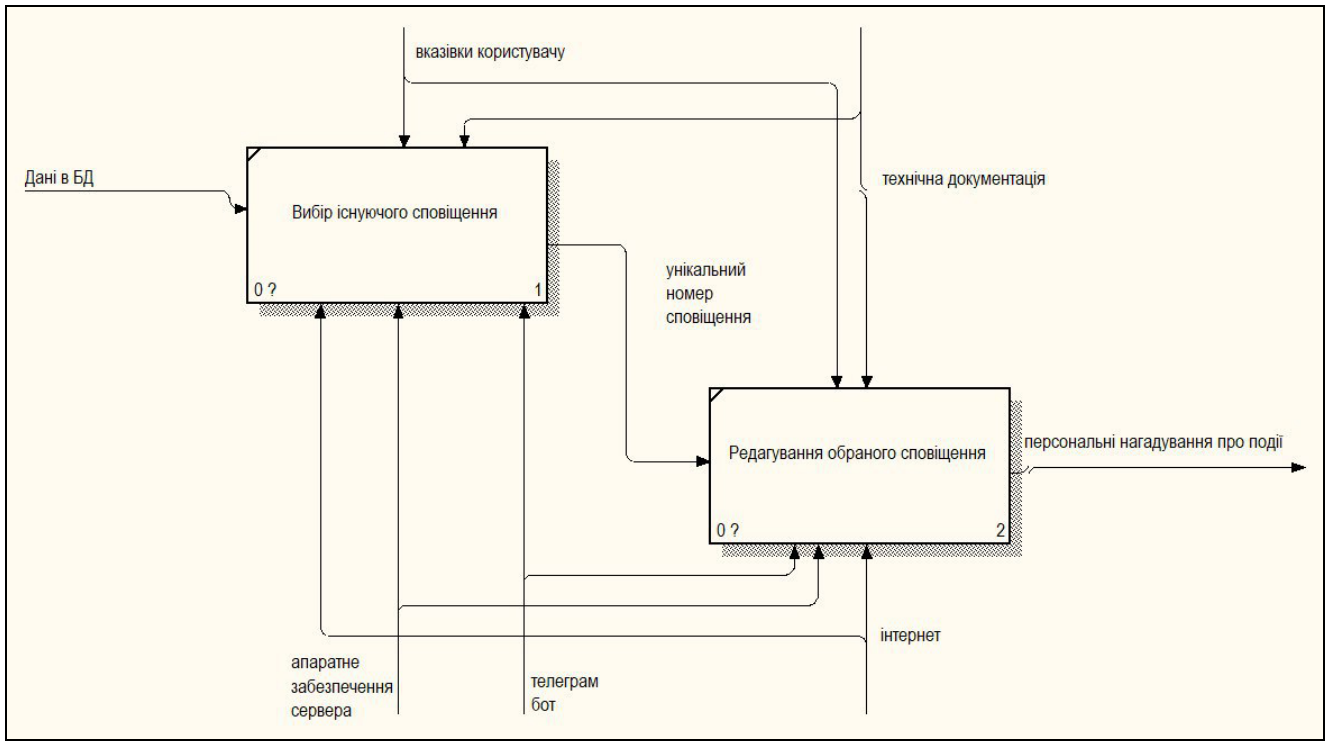

Рисунок 3.7 – Декомпозиція блоку «Налаштування сповіщень»

3.3 Моделювання бази даних

В рамках даної робити необхідно спроектувати реляційну базу даних, що являється сукупністю таблиць які зв’язані між собою. І ці зв’язки встановлюються по атрибутам зв’язаних між собою таблиць. Таблиці реляційних баз даних мають такі властивості:

· кожен табличний елемент являється одним елементом даних;

· у всіх рядків стовпця присутні однакові властивості, такі як тип даних, довжина чи значення по замовчуванню;

· у кожного стовпця – своє ім’я;

· в одній таблиці не може бути однакового стовпчика;

· стовпчики та рядки в таблиці можна розміщувати у довільному порядку.[13]

До вашої уваги на рисунку 3.8 представлена змодельована база даних для даного проекту. При розробці цієї моделі використовувався онлайн сервіс моделювання діаграм Draw.io.

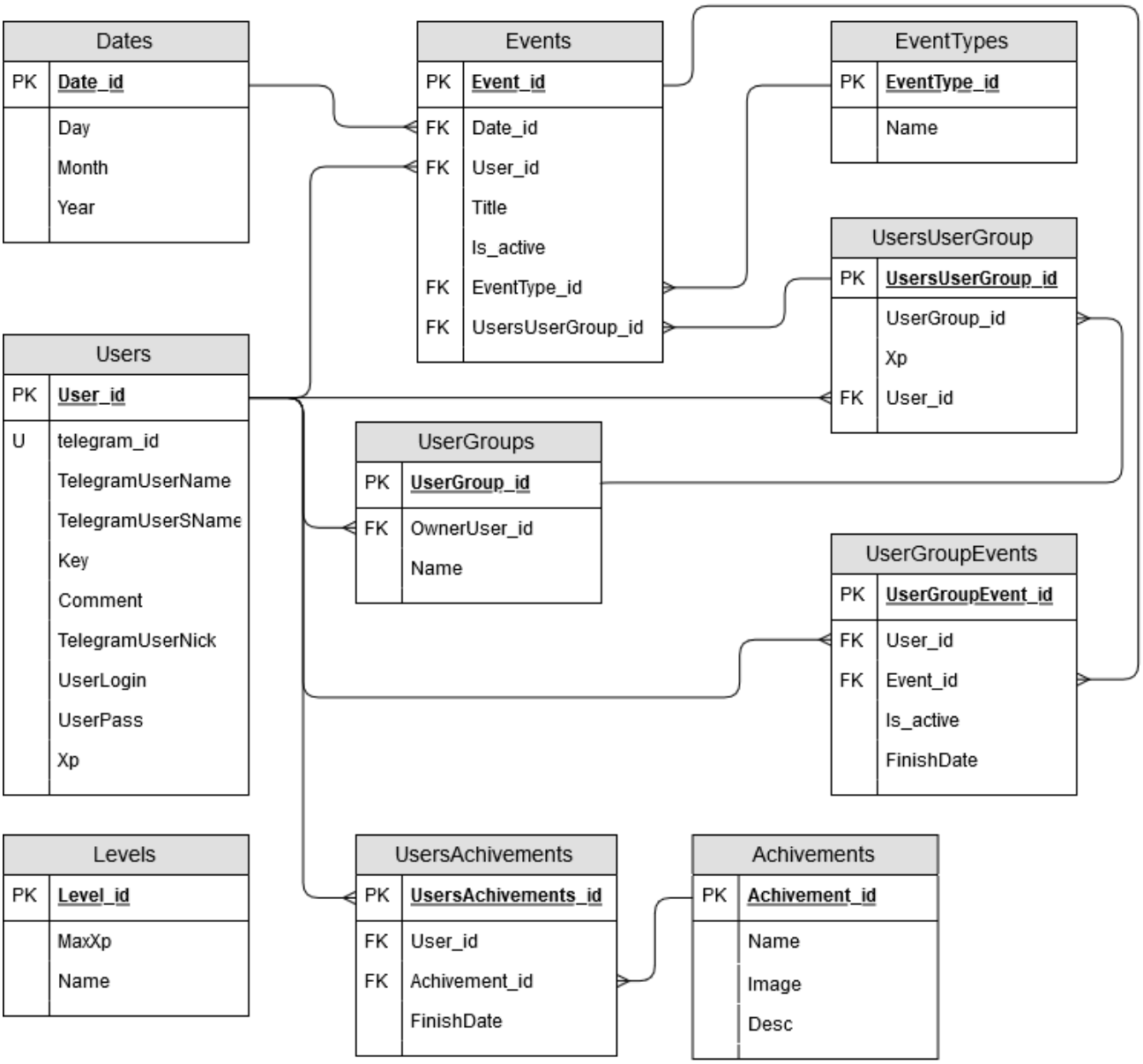

Рисунок 3.8 – Модель бази даних

Для роботи додатку необхідна база даниз яка складається із дев’яти таблиць зв’язаних між собою, а також однієї не зв’язаної таблиці яка виконує функцію зберігання інформації про рівні.

Таблиця 3.1 - Сутності, ключі та атрибути бази даних

Сутність	**Опис сутності**
Dates.	Дана сутність являється сховищем всіх дат для подій системи. Містить наступні атрибути: day тип: int(2), month тип: int(2), year тип: int(4), ключ: Date_id типу int(11)
Events.	Дана сутність містить в собі дані про події системи, їх назву, їх тип, їх дата, а також зв’язок між користувачем чи групою користувачів. Містить наступні атрибути: title тип: varchar(100), is_active тип: bit(1), eventType_id тип int(11) зовнішній ключ до таблиці EventTypes, Date_id тип int(11) зовнішній ключ до таблиці Dates, EventType_id тип int(11) зовнішній ключ до таблиці EventTypes, UsersUserGroup_id тип int(11) зовнішній ключ до таблиці UsersUserGroup, первинни ключ: Date_id типу int(11)
EventTypes.	Дана сутність являється словником типу подій в проекті, наприклад: «День народження». Містить в собі наступні атрибути: Name тип: varchar (25), первинний ключ: EventType_id типу int(11)

Продовження таблиці 3.1

Users.	Дана сутність зберігає дані користувача, а саме його ім’я та прізвича яке він зазначив у месенджері Telegram, його логін та пароль для веб-додатку, його персональний телеграм-ідентифікатор, коментарій до запису та кількість досвіду. Сутність містить наступні атрибути: telegramUserName тип: varchar (50), telegramUserSname тип: varchar (50), Key тип: varchar (25), Comment тип: varchar (100), telegramUserNick тип: varchar (25), UserLogin тип: varchar (25), UserPass тип: varchar (25), Xp тип: int (11), унікальний ключ telegram_id типу int(11), первинний ключ: User_id типу int(11)
UserGroups.	Сутність зберігає в собі дані про користувацькі групи, та їх власників. Містить наступні атрибути: Name тип: varchar(50), OwnerUser_id тип int(11) зовнішній ключ до таблиці Users, первинний ключ: UserGroup_id типу int(11)
UsersUserGroup.	Сутність визначає відповідність користувачів до груп користувачів, а також кількість досвіду користувача. Містить наступні атрибути: Xp тип: int (11), User_id тип int(11) зовнішній ключ до таблиці Users, UserGroup_id тип int(11) зовнішній ключ до таблиці UserGroups, первинний ключ: UsersUserGroup _id типу int(11)

Продовження таблиці 3.1

UserGroupEvents.	Сутність містить у собі список задач для конкретного користувача групи, їх стан та дату завершення(якщо є), дана сутність має наступні атрибути: is_active тип: bit(1), finishDate тип: datetime, User_id тип int(11) зовнішній ключ до таблиці Users, Event_id тип int(11) зовнішній ключ до таблиці Events, первинний ключ: UserGroupEvent_id типу int(11)
UsersAchievements.	Дана сутність зберігає в собі інформаці. Про отримані користувачем досягнення, а також про час їх отримання. Містить наступні атрибути: FinishDate типу datetime, Achievement_id тип int(11) зовнішній ключ до таблиці Achievements, User_id тип int(11) зовнішній ключ до таблиці Users, первинний ключ: UsersAchievements_id типу int(11).
Achievements	Дана сутність являється словником досягнень які доступні для користувача. Сутність зберігає інформацію про назву досягнення, його опис, а також назву картинки яка буде відображатися на клієнтській частині додатку. Містить наступні атрибути: Name тип: varchar(25), Image тип: varchar(22), Desc тип: varchar(100), первинний ключ: Achievement_id типу int(11).

Продовження таблиці 3.1

Levels	Сутність містить у собі інформацію про кількість досвіду необхідний для отримання того чи іншого рівня. Містить наступні атрибути: maxXp тип: int(11), Name тип: varchar (25), первинний ключ: Level_id типу int(11)

3.4 Засоби дослідження

В даній роботі будуть використовуватися такі засоби дослідження:

- Front-End фреймворк Vue.js
- Back-End фреймворк Node.js
- Мова структурованих запитів MySQL
- Ядро телеграм боту для Node.js – «node-telegram-bot-api»

Vue.js це фреймворк призначений для того щоб надати можливість розробнику створювати веб-додатки, а саме інтерфейси для користувачів. В порівнянні з більш складними фреймворками-монолітами, Vue.js достатньо придатний щоб впровадити в продукт поступово. Основа даного фреймворка по перше, може вирішувати задачі різного типу, що достатньо полегшує роботу з різними сторонніми додатками (бібліотеками), а також із уже існуючими розробками. Також, цей фреймворк можна використовувати для створення простих та ускладнених лендінгах. Крім цього можна використовувати Vue.js разом із сучасними інструментами розробки а також із додатковими бібліотеками.

Node.js - це програмний засіб що являється платформою для запуску та роботи серверу з підтримкою JavaScript, що виконує дії та функції на клієнтській

стороні, а Node - на серверній його частині. За допомогою Node можна писати повноцінні програми. Node.js може працювати з зовнішніми бібліотеками, запускати JavaScript команди та працювати в якості сервера для веб-додатків.

MySQL - це мова структурованих запитів, що надає змогу користувачеві управляти базою даних, що являє собою сукупність структурованих даних, об’єднаних між собою зв’язком. Дані можуть мати будь-який вміст - від переліку покупок до списку днів народжень студентів.

Node-telegram-bot-api – це спеціальна бібліотека яка встановлюється на Node.js сервер, що надає змогу розробнику використовувати всі доступні функції додатку телеграм бота.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Серверна реалізація

Перше, що необхідно виконати перед розробкою даного додатку – це обрати та налаштувати сервер. На даний момент існує достатня кількість сервісів для реалізації даної задачі. Існують як безкоштовні так і платні ресурси, надання хостигових послуг. В рамках дипломного проекту будуть використовуватися тільки безкоштовні.

Для реалізації програмного продукту необхідно використовувати такі хостингові сервіси:

1) Сервер для бази даних.
2) Веб-сервер для клієнтської частини додатку.
3) Сервер для реалізації API частини додатку.

Щоб реалізувати базу даних було обрано ресурс «RemoteMySQL» зображений на рисунку 4.1, який являється безкоштовним та надає користувачеві можливості роботи із базою даних, достатніх для реалізації даної дипломної роботи.[17]

Даний сервіс дозволяє безкоштовно створювати до трьох баз даних на одному акаунті. Доступні майже всі SQL команди, є можливість налаштування привілегій, а також перегляд статистики бази даних.

Мінус даного ресурсу полягає в тому, що якщо в базі даних не виконуються запити протягом одного місяця – база автоматично видаляється, але у нашому

випадку запити до бази даних будуть виконуватися кожного дня як мінімум один раз.

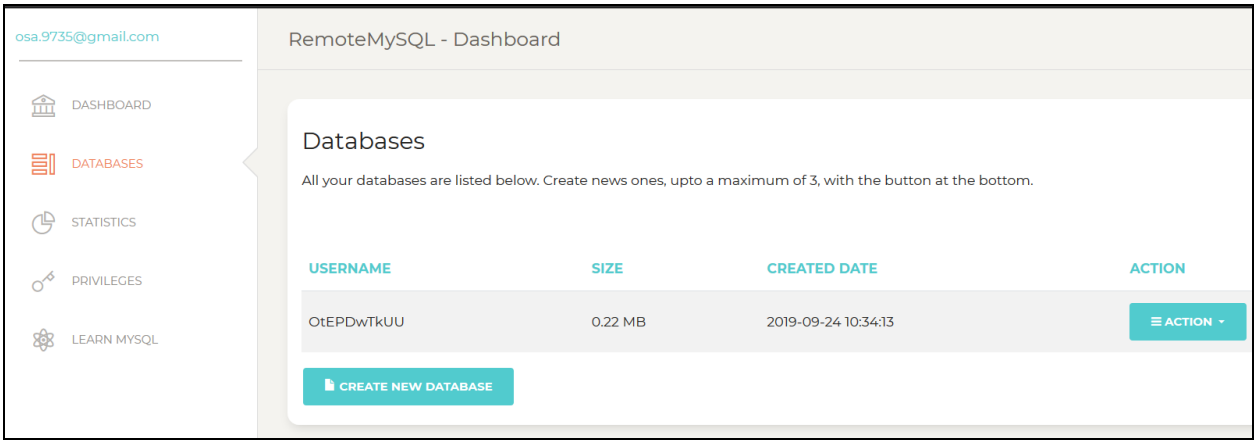

Рисунок 4.1 – Сервіс RemoteMySQL

Для реалізації клієнтської частини додатку, а також для його API було обрано ресурс «Heroku» (рис. 4.2). Даний сервіс надає змогу користувачам реалізувати як прості так і складні веб-орієнтовані додатки. На рисунку 4.3 показаний перелік методів розробки які підтримуються даним ресурсом.[18]

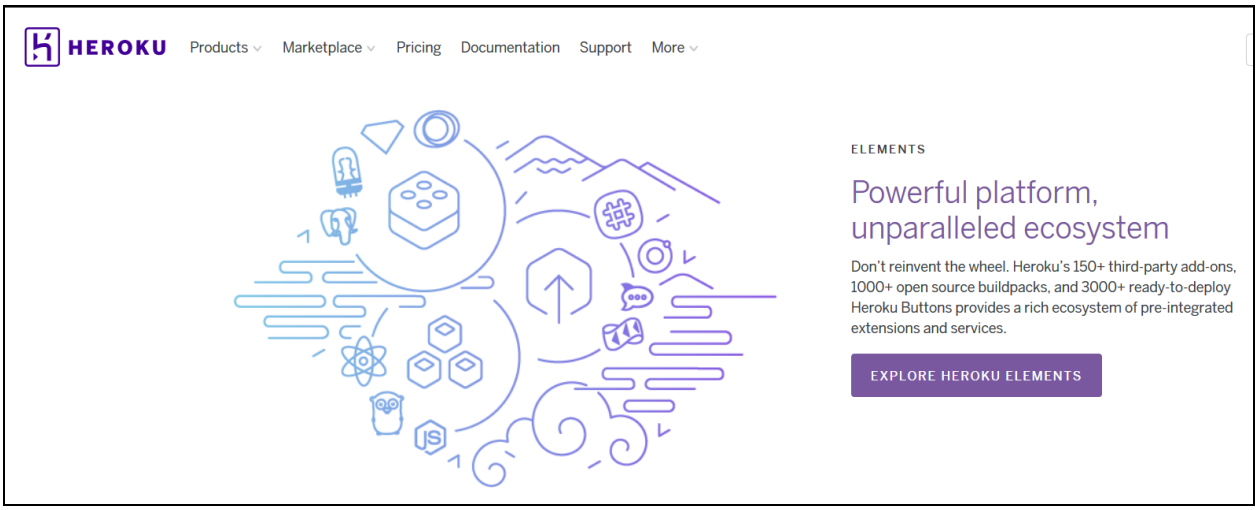

Рисунок 4.2 – Сервіс «Heroku»

В даній дипломній роботі буде використовуватися платформа Node.js, яка підтримується даним хостинговим ресурсом.

Клієнтську частину буде реалізовано фреймворком Vue.js, що реалізується на уже обраній платформі.

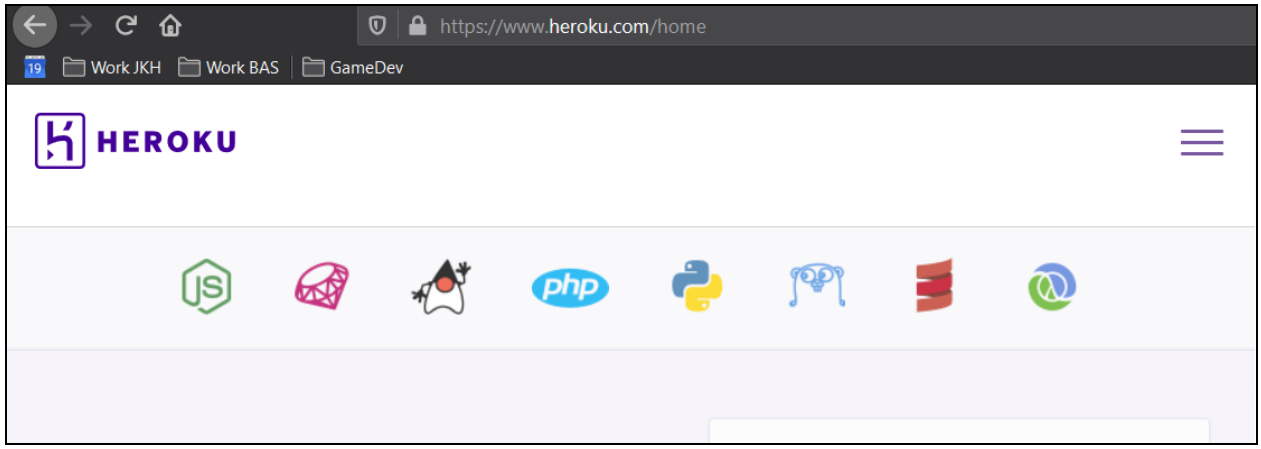

Рисунок 4.3 – Доступні методи розробки на ресурсі «Heroku»

Даний сервер дозволяє зареєстрованим корисувачам запускати свої додатки 550 годин на місяць, але якщо указати дані кредитної картки, то можна отримати 1000 годин. На сайті можна знайти детальну статистику по користувацьким додаткам (рис. 4.4). У нашому випадку додаток буде працювати лише в під час розробки, щоб не витрачати години.

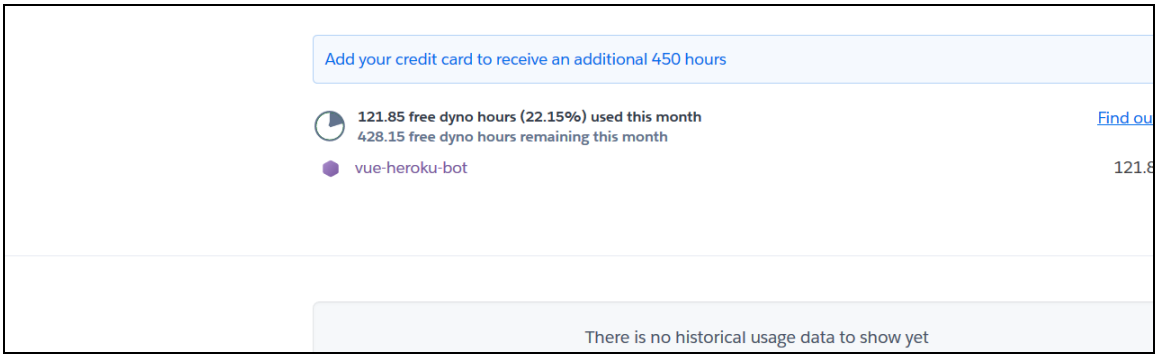

Рисунок 4.4 – Стан використаних годин на хостингу.

Для того, щоб додаток розпочав роботу потрібно або завнтажити свій проект на веб-ресурс для зберігання вихідного коду проектів «GitHub», або ж встановити спеціальний додаток на комп'ютер, який дозволить підключатися до серверів Heroku напряму. Для розробки було обрано другий метод.

4.2 Налаштування проекту

Далі необхідно завантажити та встановити платформу Node.js, для цього переходимо на офіційний сайт та завантажуємо LTS версію додатку (рис. 4.5) – це та яка на даний момент буде довготривало підтримуватися розробниками. Після завантаження – проводимо інсталяцію продукту.[19]

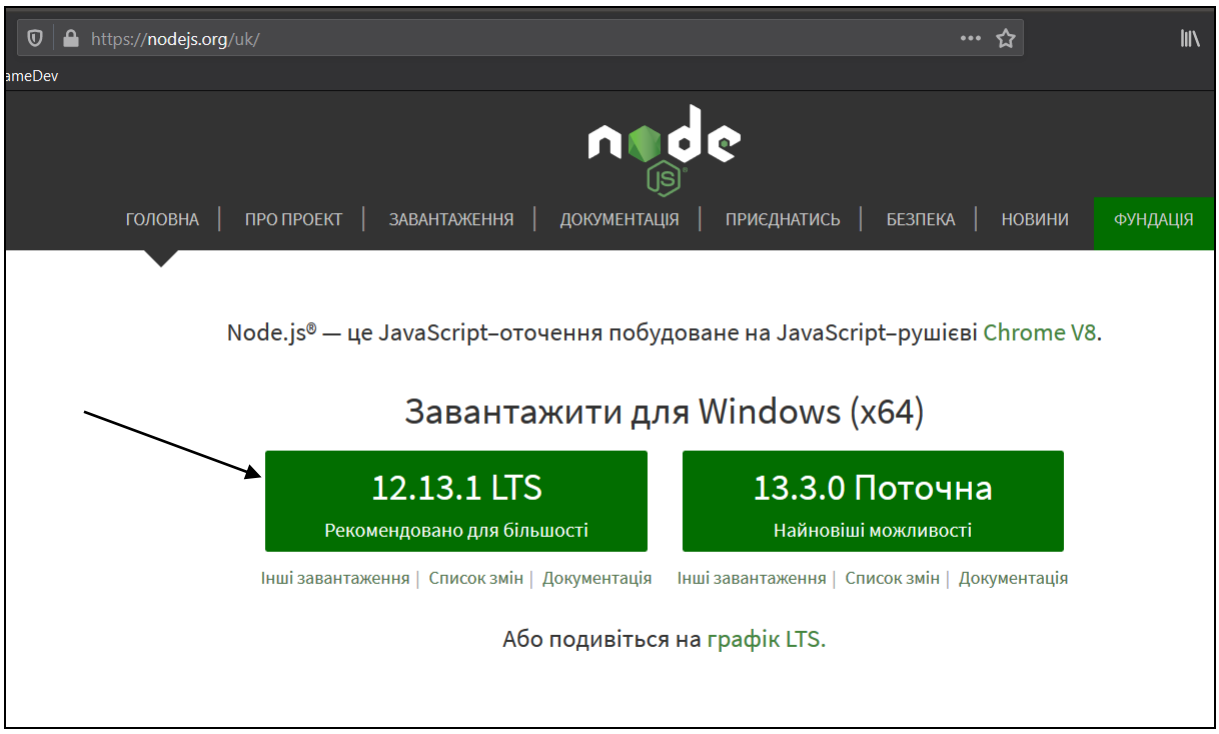

Рисунок 4.5– Вибір версії node.js.

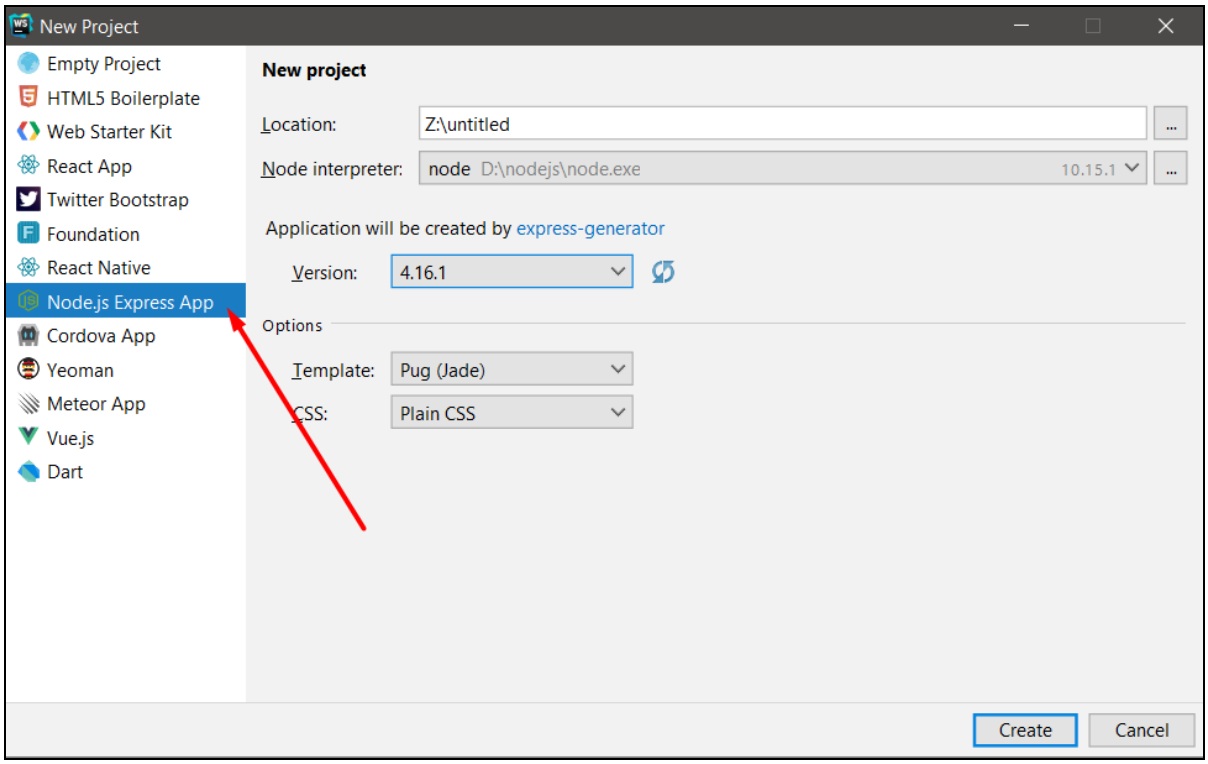

Рисунок 4.6– Створення проекту.

Тепер у нас є можливість створювати Node.Js додатки та запускати їх на локальному сервері. Наступним кроком було створення проекту в додатку JetBrains WebStorm. Для цього, при створенні нового проекту потрібно обрати метод розробки «Node.js», назвати проект та натиснути кнопку «Create» (рис. 4.6).

Після того як проект створено, необхідно встановити фреймворк vue.js, який встановлюється через термінал який вмонтовано в програмний продукт JetBrains WebStorm.

Виконується встановлення командою *npm install -g vue-cli*, (рис. 4.7) після її введення виконається завантаження всіх необхідних компонентів в проект, тому необхідне підключення до інтернету, щоб виконати дану команду.

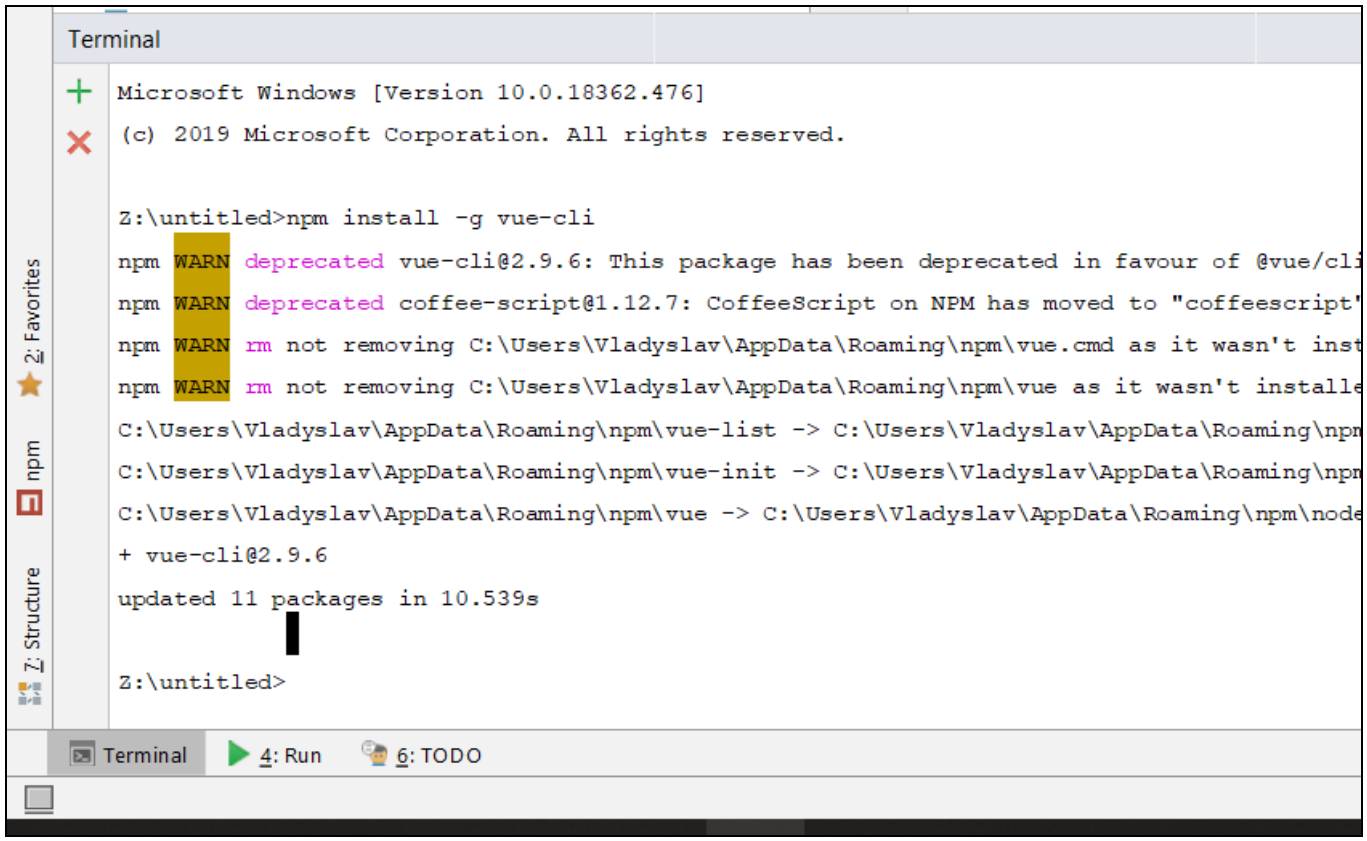

Рисунок 4.7– Встановлення фреймворку Vue.js .

Після того як фреймворк Vue.js було встановлено потрібно виконати команду ініціалізації проекта *vue init webpack my-project*.

Далі, під час встановлення, термінал запитає у нас чи потрібно встановлювати бібліотеку «vue-router», у відповідь потрібно написати «y», тобто погодитися на встановлення (рис. 4.8). Дана бібліотека призначена для: створення вкладених маршрутів чи уявлень, модульної конфігурація маршрутизатора, зручного контролю навігації. Також ця бібліотека маршрутизації являється офіційною для Vue.js.

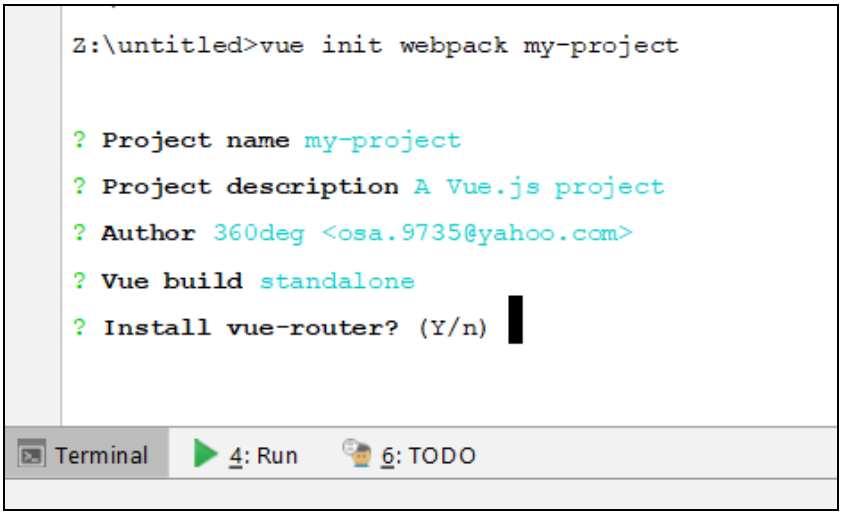

Рисунок 4.8– Ініціалізація проекту .

Після погодження на встановлення «vue-router», термінал запитає ще декілька питань (рис. 4.9). Відповіді на які – це персональні вподобання розробника, які впливають тільки на процес розробки програмного продукту та його тестування.

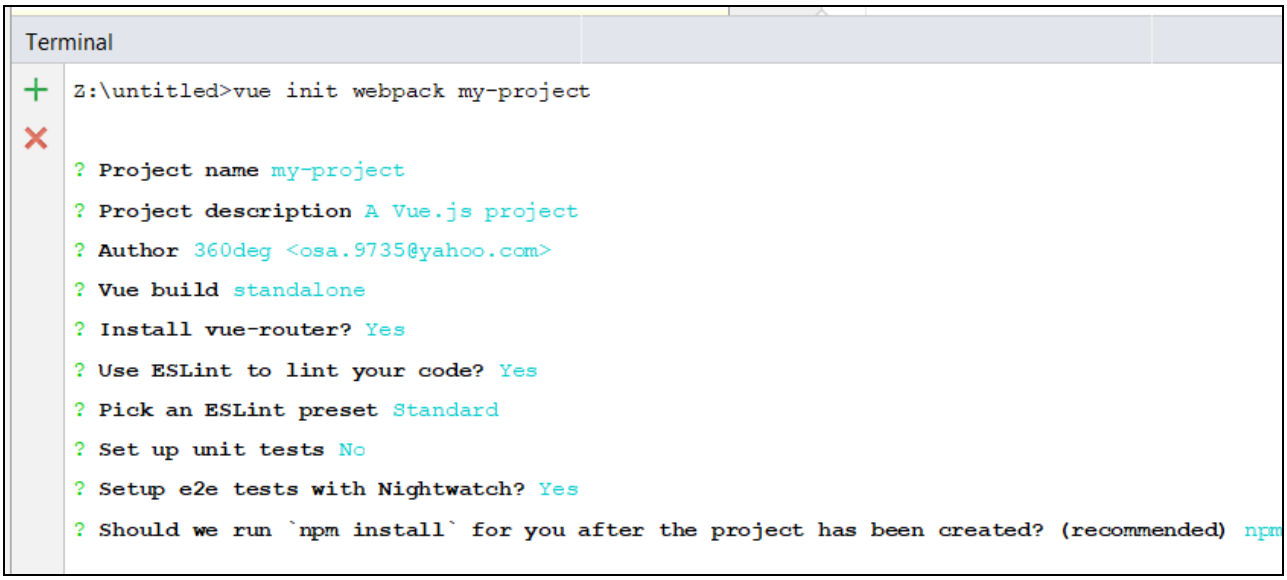

Рисунок 4.9– Ініціалізація проекту .

Якщо все було виконано правильно, то після введення команди *npm start* розпочнеться компіляція проекту і потім, перейшовши на локальну адресу «http://localhost:8080», можна побачити стартову сторінку фреймворка vue.js (рис. 4.10).

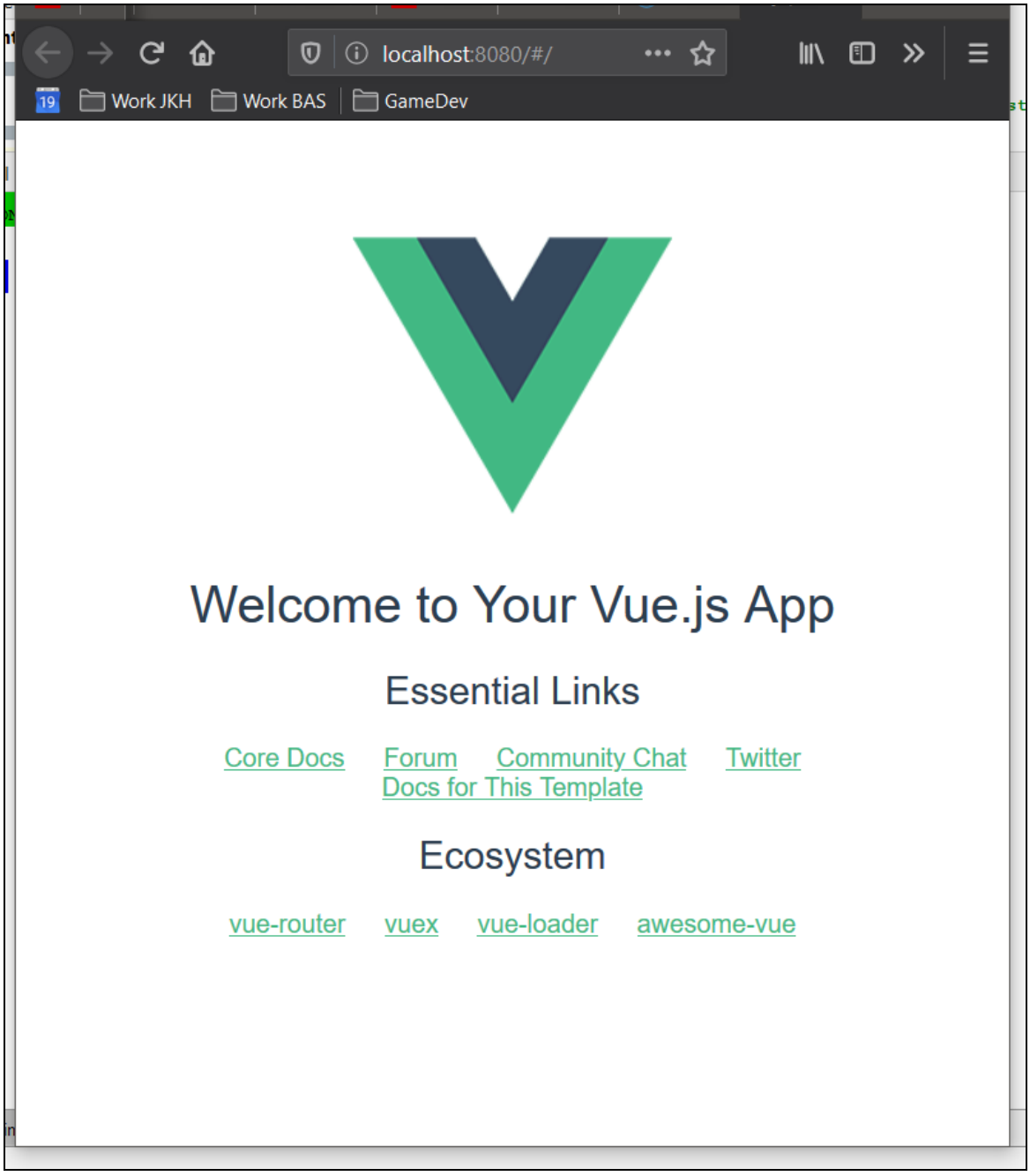

Рисунок 4.10– Стартова сторінка пустого проекту.

Завершаючим кроком в підготовці проекта до роботи буде створення файлу, необхідного для хостингового сервісу, який і буде запускати створюваний програмний продукт (рис. 4.11). Назва файлу повинна бути «Procfile», а вміст слід записати таким: *web: npm start.*

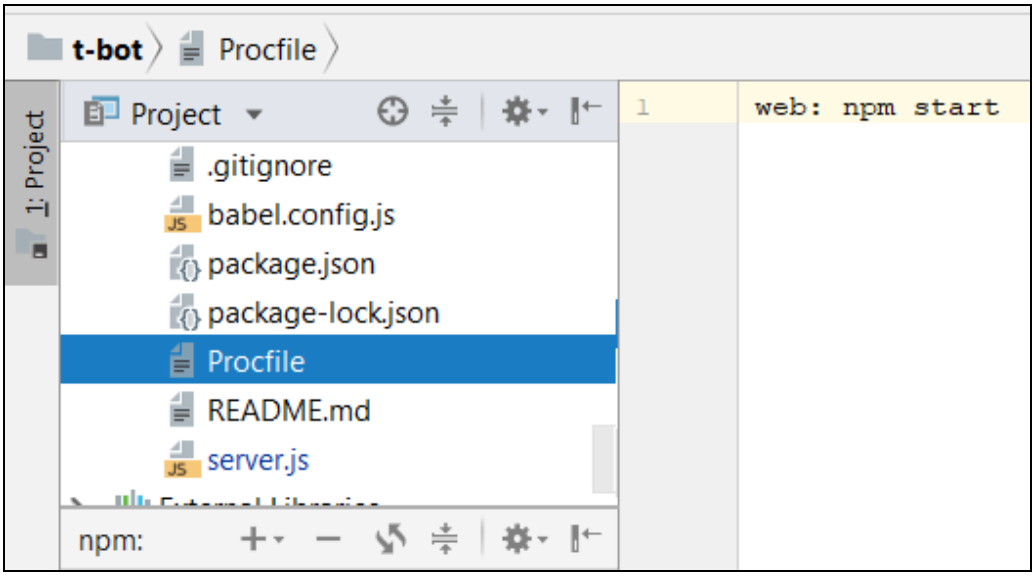

Рисунок 4.11– Procfile проекту.

4.3 Реалізація бази даних

Для реалізації бази даних необхідно завантажити бібліотеку «mysql» в проект. Щоб це зробити необхідно відкрити термінал та ввести в ньому команду «npm install mysql» (рис. 4.12). Після успішного завантаження у нас з’явиться можливість підключення до бази даних інструментами імпортованої бібліотеки.

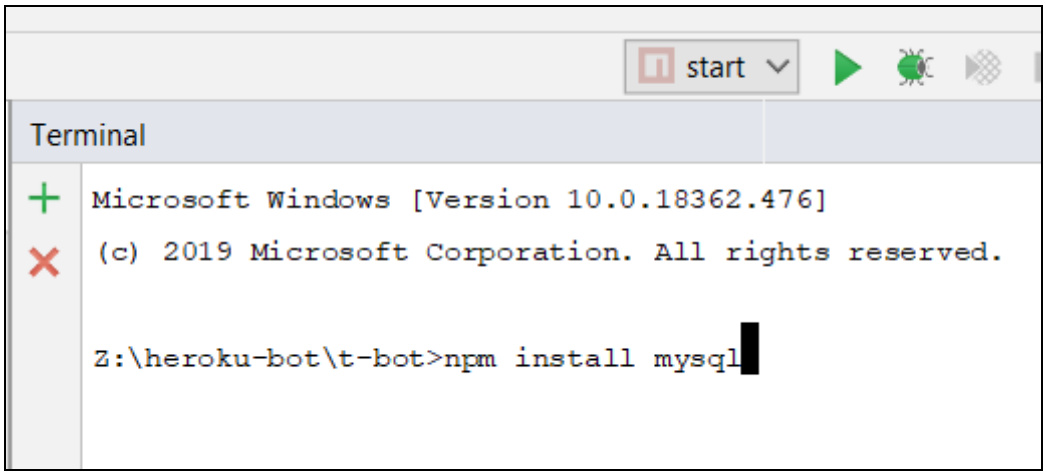

Рисунок 4.12– Завантаження бібліотеки mysql.

Наступним кроком буде створення файлу із розширенням *.js в якому і буде прописане підключення до бази даних. Файл був названий як connection.js.

В файлі потрібно прописати необхідні дані для з'єднання із базою даних:

1) Назва хоста.
2) Номер порта.
3) Користувач під яким виконується вхід.
4) Пароль для вищезазначеного користувача.
5) Назва бази даних.

Ці дані необхідно передати як об'єкт у вбудований метод «createPool» бібліотеки Mysql.

Щоб імпортувати бібліотеку mysql, необхідно прописати *const mysql = require('mysql')* в javascript-файлі де буде реалізоване під'єднання до бази даних проекту. Обов'язковим являється те, що потрібно експортувати з'єднання для того, щоб з'явився доступ до даних поза межами редагованого файлу. На рисунку 4.13 зображений фінальний вигляд файлу підключення.

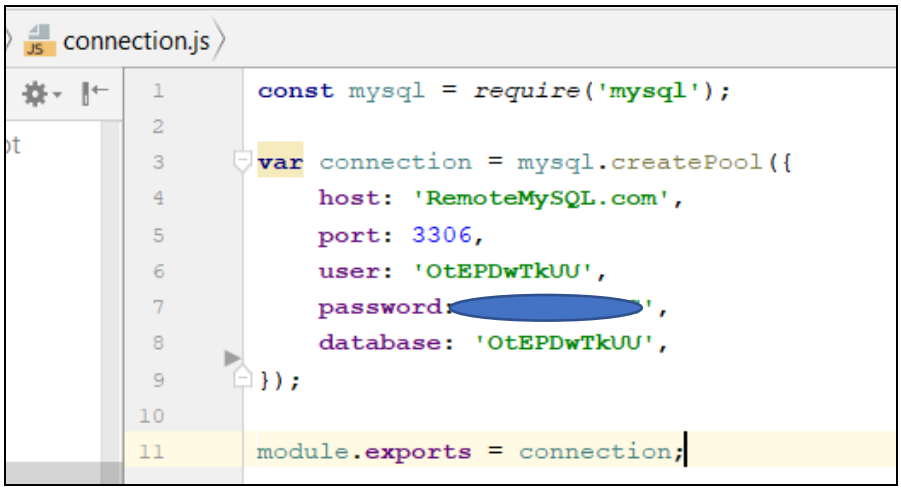

Рисунок 4.13– Створення зв’язку з базою даних.

Для того щоб звертатися до змінної «connection», потрібно імпортувати даний файл там де проводиться звернення.

Наступним кроком було створення універсального методу виконання запитів у базу даних.

Для запитів які повертають значення, наприклад, отримання списку користувачів було написано метод із функцією «callback». Ця функція повертає значення тільки в тому випадку коли отримує ті чи інші дані. Її необхідно використовувати в тому випадку якщо метод являється асинхронним і відповідь потрібно отримати якнайшвидше.

Для запитів які не потребують зворотнього зв’язку, наприклад вставка нових даних чи навпаки – видалення уже існуючих даних, було створено метод який виконує запит в базу даних без очікування відповіді.

Метод *run* виконує запит із зворотнім зв’язком, а метод *runVoid* (рис. 4.14) виконує запит до бази даних і не повертає ніякої відповіді від сервера назад на API.

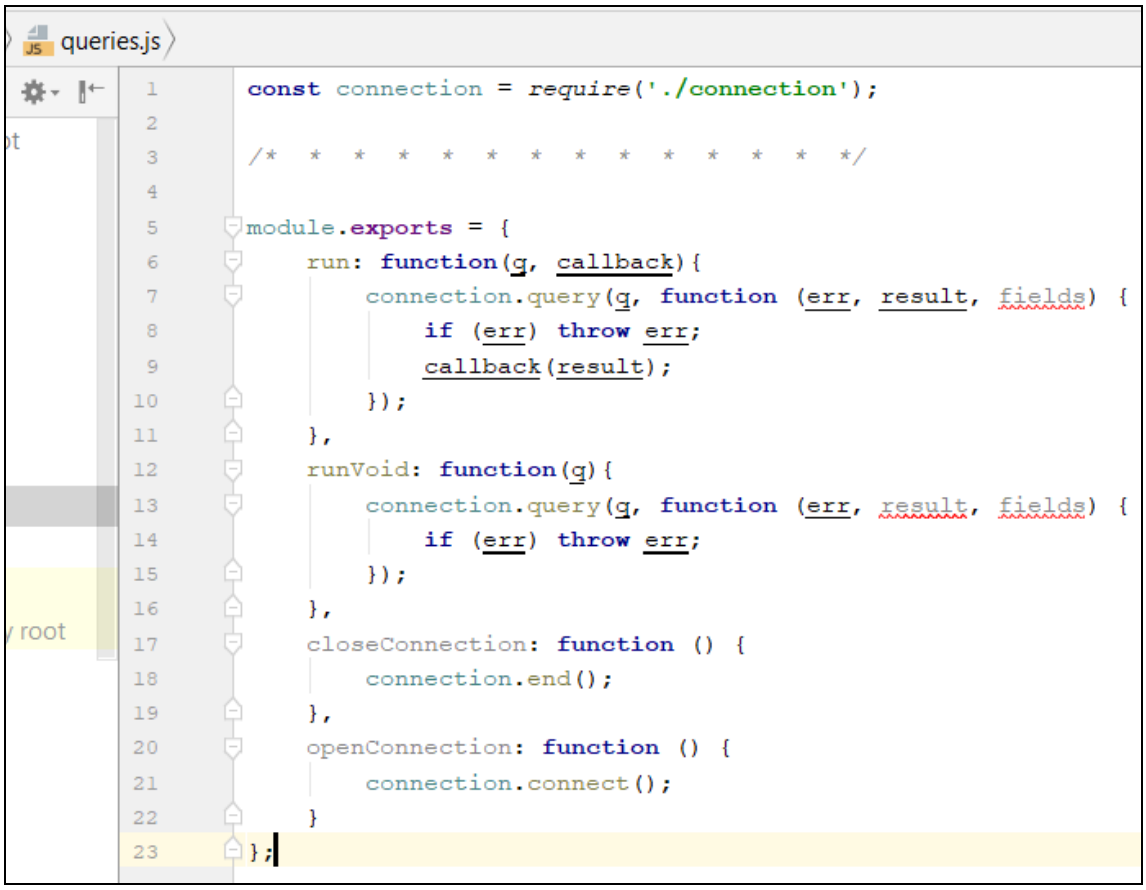

Рисунок 4.14– Створення методів виконання запитів.

4.4 Розробка API

Для того, щоб мати доступ до створеного телеграм-боту необхідно завантажити спеціальну бібліотеку «node-telegram-bot-api», використовуючи термінал (рис. 4.15).

У нашому випадку, термінал уже вмонтовано у програму розробки веб-орієнтованого продукту, але також можна виконати всі команди в консолі операційної системи.

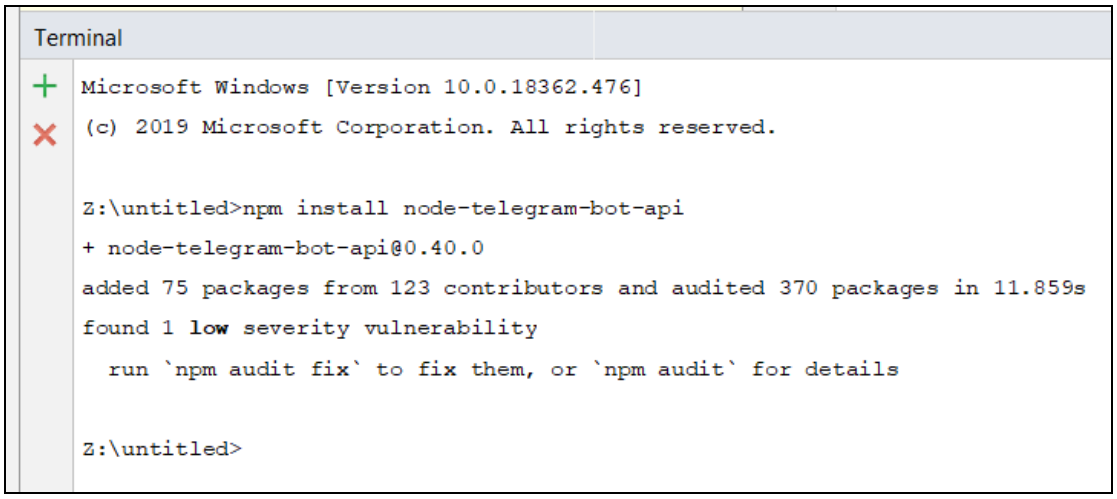

Рисунок 4.15– Завантаження бібліотеки телеграм бота.

Після встановлення бібліотеки потрібно викликати її функцією «require» (рис. 4.16).

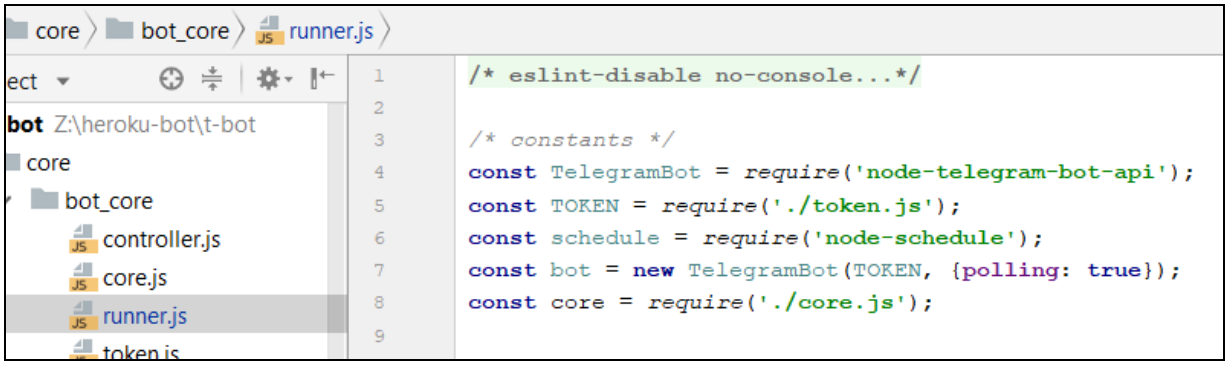

Рисунок 4.16– Виклик бібліотеки телеграм бота.

Щоб користуватися ботом, необхідно присвоїти змінній новий клас, та передати в нього унікальний токен, який видає месенджер при реєстрації нового телеграм-бота. Токен розміщено в окремий файл (рис.4.17), після чого, його було експортовано, для надання доступу до змінної поза межами створеного файлу.

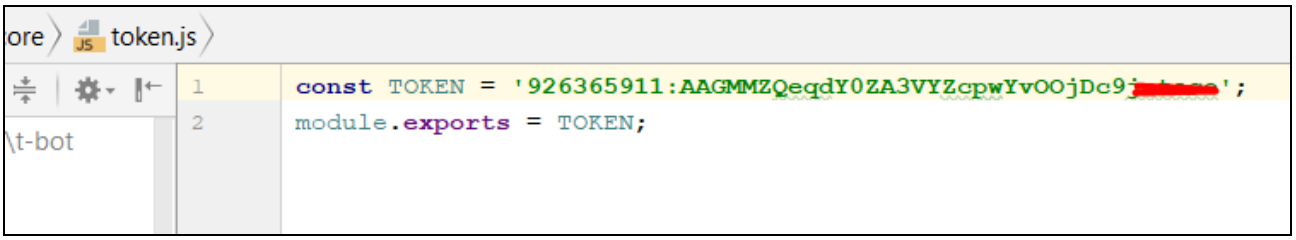

Рисунок 4.17– Експорт токена.

Для того щоб отримувати повідомлення додатком існує вбудована в бібліотеку функція «on», яка отримує інформацію про повідомлення та про його відправника. Приклад реалізації метода «on» зображений на рисунку 4.18.

```
function runMessageReceiver(){
    bot.on('message', function(msg) {

        switch (msg.text) {
            case undefined: console.log('Undefined value!'); break;
            case '/start': core.reg_registration(msg.from.id, msg.from.first_name, msg.from.last_name, msg.from.username); break;
        }


        let chatId = msg.chat.id;
        if (chatId === 366741945){
            let date_ob = new Date();
            let time = date_ob.getHours() + " " + date_ob.getMinutes();
            bot.sendMessage(chatId, time);
        }
        bot.sendMessage(chatId, 'You sent something!');
    });
}


module.exports = {
    run: function(){
        runMessageReceiver();
    },
```

Рисунок 4.18– Виклик бібліотеки телеграм бота.

Якщо користувач натискає на кнопку «/start», коли відкриває телеграм бота вперше, то проходить автоматична реєстрація користувача в базі даних. Але в тому випадку коли користувач видаляє бота із акаунту, запис в базі даних залишається, при повторній реєстрації, новий запис про користувача не створюється, а оновлюється уже існуючий (рис. 4.19). В базі даних інформація зберігається як показано на (рис. 4.20);

```
    /*** РЕЄСТРАЦІЯ ***/
    static reg_registration(telegram_id, telegramUserName, telegramUserSName, telegramUserNick){
        serv.run(q.isRegistered(telegram_id), function(results) {
            if(results[0].Count === 1){
                serv.runVoid(q.updateUser(telegram_id, telegramUserName, telegramUserSName, telegramUserNick));
            } else {
                let login = telegram_id;
                let pass = generator.generate({
                    length: 4
                });
                serv.runVoid(q.addUser(telegram_id, telegramUserName, telegramUserSName, telegramUserNick, login, pass));
            }
        });
    }

    /*** ВХІД ***/
    static login_isUserExists(l, p, callback){
        serv.run(q.isUserExist(l, p), function (results) {
            if (results.length === 1) {
                let b = {
                    'userName': results[0].TelegramUserName,
                    'userSName': results[0].TelegramUserSName,
                    'userId': results[0].User_id,
                };
                callback(b);
            }
        });

    }
```

Рисунок 4.19– Методи реєстрації та перевірки авторизації.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

User_id	Telegram_id	TelegramUserName	StringKey	Comment	TelegramUserNick	UserLogin	UserPass	TelegramUserSName
1	366741945	Vlad	NULL	ya	reabl_on			Pecherytsia
4	353284498	Anastasia	NULL	Настюха	frostlady			Fedorova
5	389126277	Edward	NULL	NULL	edge_s			Sylenko
6	307915040	Danil	NULL	NULL	nem0sheeva			Gaponyuk
7	390361397	Vlad	NULL	NULL	Vladosik_QQ			Pravedniy

Рисунок 4.20– Зберігання інформації про користувачів в базі даних.

4.5 Створення клієнтської частини додатку

Для того щоб відправляти та отримувати дані з бази даних та з API, необхідно реалізувати систему GET/POST запитів. Щоб це зробити необхідно імпортувати бібліотеку «vue-resource», в головному файлі клієнтської частини,

після чого прописати *«Vue.use(vueResource)»*, щоб можна було використовувати дану бібліотеку на сторінках клієнтської частини додатку (рис. 4.21).

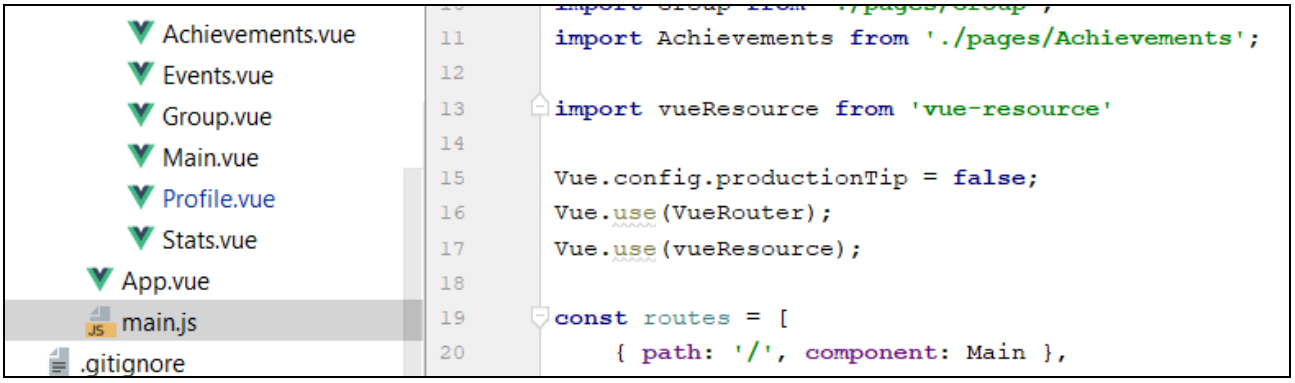

Рисунок 4.21– Імпортування бібліотеки «vue-resource».

Тепер можна використовувати запити до сервера. Зробити це можна наступним чином:

1) Створити обробник події натиску на кнопку (рис. 4.22).
2) Реалізувати метод входу користувача в систему, при цьому передавати через POST запит введені користувачем дані як це зображено на рисунку 4.23.
3) Створити новий маршрут для запиту на API проекту (рис. 4.24).

Маршрути, або так звані «роути» необхідні будь-якому складному веб-орієнтованому додаткові. Вони слугують для навігації по сторінкам сайту, а також для реалізації запитів на API.

«Роути» указують додатку, що робити у випадку, якщо користувач ввів неіснуючу адресу домені, або якщо передалися ті чи інші параметри разом із посиланням веб-ресурсу.

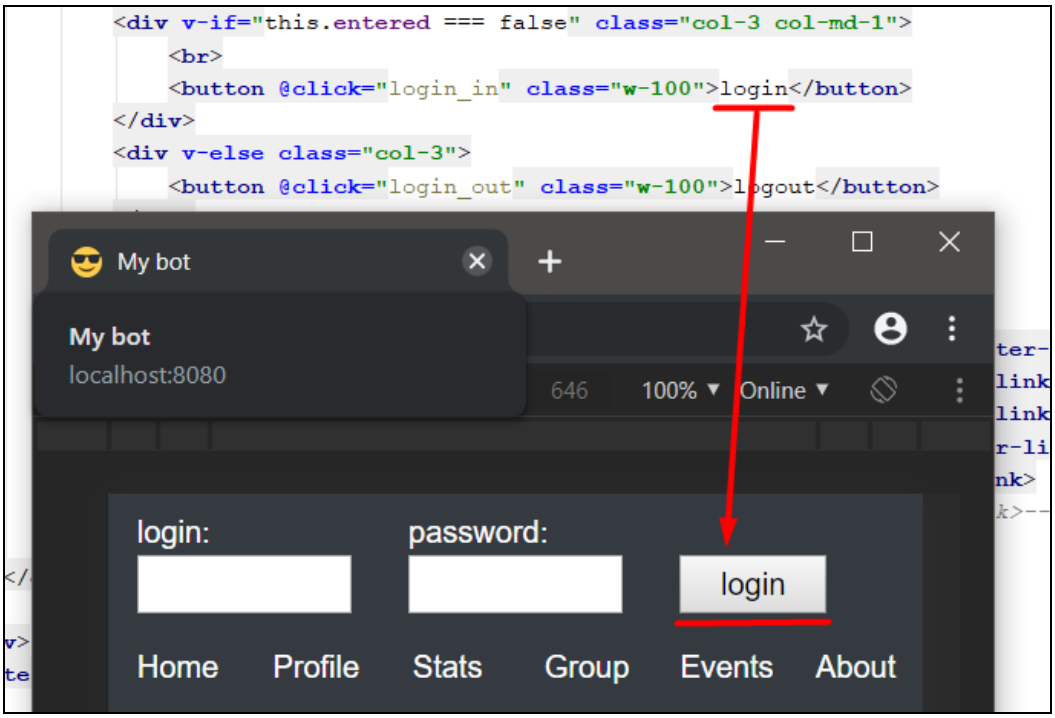

Рисунок 4.22– Демонстрація кнопки входу в систему.

```
methods: {
    login_in(){
        let login = document.getElementById("login");
        let pass = document.getElementById("pass");
        let me = this;
        Vue.http.post('/login', {
            'userLogin': login.value,
            'userPass': pass.value
        }).then(function (data) {
            me.entered = data.body.entered;
            me.userName = data.body.userName;
            me.userSName = data.body.userSName;
        });
    },
    login_out(){
        this.entered = false;
        Vue.http.get('/logout');
    }
```

Рисунок 4.23– Демонстрація методів входу та виходу із системи.

```
// Перевірка логіна і вход в сесію
app.use('/login', function (req, res) {
    core.login_isUserExists(req.body.userLogin, req.body.userPass, function (result) {
        if(result){
            req.session.user = req.body.userLogin;
            req.session.userId = result.userId;
            req.session.userName = result.userName;
            req.session.userSName = result.userSName;
            res.send({
                'entered': true,
                'userName': result.userName,
                'userSName': result.userSName
            });
        } else {
            res.send({
                'entered': false
            });
        }
    });

});
```

Рисунок 4.24– Авторизація користувача на API.

В маршрут «/login» надсилаються дані про логін та пароль користувача. Перше, що виконує функція – це перевірка на правильність введення даних, чи існує такий користувач в системі, якщо його не знайдено в базі даних, то вхід не відбудеться. Якщо дані було введено правильні, то відбудиться запис cookie-файлу в браузер користувача, який буде містити в собі дані про авторизацію, а також відправить на клієнтську частину інформацію про ім’я та прізвище користувача.

ВИСНОВКИ

Використання телеграм ботів для спілкування з потенційною аудиторією на сьогодні стає дуже популярним. Багато компаній користуються послугами розробників для створення їх персональних ботів.

Для реалізації моєї задачі ми проаналізували широкий спектр аналогів, проте конкурентоспроможних сервісів не було знайдено. Для розширення функціоналу телеграм бота та скорочення часу на введення даних від користувачів було вирішено розробити веб інтерфейс.

Для планування кількості етапів та відповідних годин на них було створено модель WBS та діаграму Ганта, які дозволили здійснити якісний розподіл робіт на весь термін виконання проекту. Після цього ми запланували можливі ризики та втрати від них.

Після аналізу кількох середовищ розробки було обрано програмний продукт JetBrains WebStorm.

Реалізований продукт має суттєві переваги в порівнянні зі схожими продуктами, в першу чергу це поєднання функціоналу телеграм боту разом з веб інтерфейсом, що покращує спілкування користувача з додатком.

Використання розробленого продукту дозволить раціонально розподілити завдання в дитячих колективах, а також надати необхідну мотивацію до роботи, що сприятиме ефективному виконанню тієї чи іншої задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Разработка ботов : веб-сайт. URL: https://evergreens.com.ua/ru/articles/ create-bot.html (дата звернення: 11.10.2019).
2. Смыслова Л. В. Чат-бот как современное средство интернет-коммуникаций : Молодой ученый. 2018 №9 С 36-39 URL https://moluch.ru/archive/195/48623/ (дата звернення: 19.09.2019).
3. Сервіс «НеЗабить» : веб-сайт. URL: http://nezabit.ru/ (дата звернення: 01.10.2019).
4. Google calendar: веб-сайт. URL: https://calendar.google.com/calendar/r.
5. Как создавать ботов в Telegram: веб-сайт. URL: https://habr.com/ru/post/262247/ (дата звернення: 02.09.2019).
6. PHP vs Node.js: веб-сайт. URL: https://habr.com/ru/post/273259/.
7. Порівняння Vue.js та Angular: веб-сайт. URL: https://buttercms.com/blog/comparing-angular-vs-vue (дата звернення: 24.10.2019).
8. What is an API: веб-сайт. URL: https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/ (дата звернення: 22.10.2019).
9. Архітектура системи URL: http://lib.mdpu.org.ua/e-book/vstup/L6.htm (дата звернення: 23.11.2019).
10. Трьохрівнева архітектура системи: веб-сайт. URL: https://studopedia.su/5_46061_trohrivneva-arhitektura.html (дата звернення: 24.10.2019).
11. UML діаграми URL: веб-сайт.: https://prog-cpp.ru/uml-classes/(дата звернення: 24.10.2019).

12. Методологія IDEF0: веб-сайт. URL: https://itteach.ru/bpwin/metodologiya-idef0 (дата звернення: 24.11.2019).
13. Реляційні бази даних: веб-сайт. URL: https://studopedia.com.ua/1_16805_sklad-relyatsiynoi-bazi-danih.html (дата звернення: 24.11.2019).
14. Vue.js для сомневающихся: веб-сайт. URL: https://habr.com/ru/post/329452/(дата звернення: 24.11.2019).
15. Бази даних MySql: веб-сайт. URL: http://ukrhosting.ua/bazi_danih_mysql-p-263950.html. (дата звернення: 24.11.2019).
16. Справочник по Bot API: веб-сайт. URL: https://tlgrm.ru/docs/bots/api. (дата звернення: 24.11.2019).
17. Remotemysql about page: веб-сайт. URL: https://remotemysql.com/#about. (дата звернення: 24.11.2019).
18. Heroku home page: веб-сайт. URL: https://www.heroku.com/home. (дата звернення: 24.11.2019).
19. Nodejs.org ukraine: веб-сайт. URL:https://nodejs.org/uk/.(дата звернення: 24.11.2019).
20. GET та POST запити Mozilla developer: веб-сайт. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods. (дата звернення: 24.11.2019).
21. Документація Telegram: Приклади ботів : веб-сайт. URL: https://tlgrm.ru/docs/bots/samples (дата звернення: 24.09.2019).
22. NPM та vue бібліотека: веб-сайт. URL: https://www.npmjs.com/package/vue. (дата звернення: 24.11.2019).
23. Бібліотека mysql: веб-сайт. URL: https://www.npmjs.com/package/mysql. (дата звернення: 30.09.2019).

24. Написання телеграм-бота: веб-сайт. URL: https://habr.com/ru/post/350858/. (дата звернення: 24.11.2019).
25. Меню для телеграм-бота: веб-сайт. URL: https://habr.com/ru/post/459604/. (дата звернення: 24.11.2019).
26. Як створити якісний веб-продукт? веб-сайт. URL: https://tproger.ru/translations/23-practical-web-design-tips/. (дата звернення: 24.11.2019).
27. HTML – початок створення сторінки: веб-сайт. URL: http://ru.html.net/tutorials/html/. (дата звернення: 24.11.2019).
28. Веб-дизайн та веб-розробка: веб-сайт. URL: https://www.motocms.com/blog/ru/web-design-i-razrabotka/. (дата звернення: 07.10.2019).
29. T-bot: веб-сайт. URL: https://github.com/360deg/t-bot. (дата звернення: 02.09.2019).
30. Знайомство з JavaScript: веб-сайт. URL: https://www.ibm.com/developerworks/ru/library/wa-javascriptstart/index.html. (дата звернення: 01.10.2019).

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

А.1 Деталізація мети проекту методом SMART

Мета роботи: розробити програмний додаток підтримки планування заходів в дитячих колективах. Даний проект надаватиме користувачеві розробленого програмного продукту можливість раціонально розподілити завдання в дитячих колективах, що сприятиме ефективному виконанню тієї чи іншої задачі.

А.2 Планування змісту структури робіт ІТ-проекту

Ієрархічна структура робіт (WBS)– це графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. На верхньому першому рівні WBS фіксується програмний продукт. Наступний ІІ рівень відповідає діям або основним заходам для досягнення поставленої мети проекту. Наступним кроком являється розбивка цих дій до виконання дій елементарних робіт. Елементарні роботи – це роботи, які мають один чіткий результат, який використовується при прийнятті цієї роботи; на які призначений один конкретний відповідальний; на неї необхідно обчислити витрати праці і тривалість виконання. У даному проекту елементарними роботами являються Розробка ПП, Формування ТЗ, Тестування, Аналіз предметної області.

OBS-структура проекту – це структура виконавців проекту. Визначається за переліком пакетів робіт нижнього рівня кожної гілки WBS-структури. Представляється відповідальними за виконання елементарних робіт.

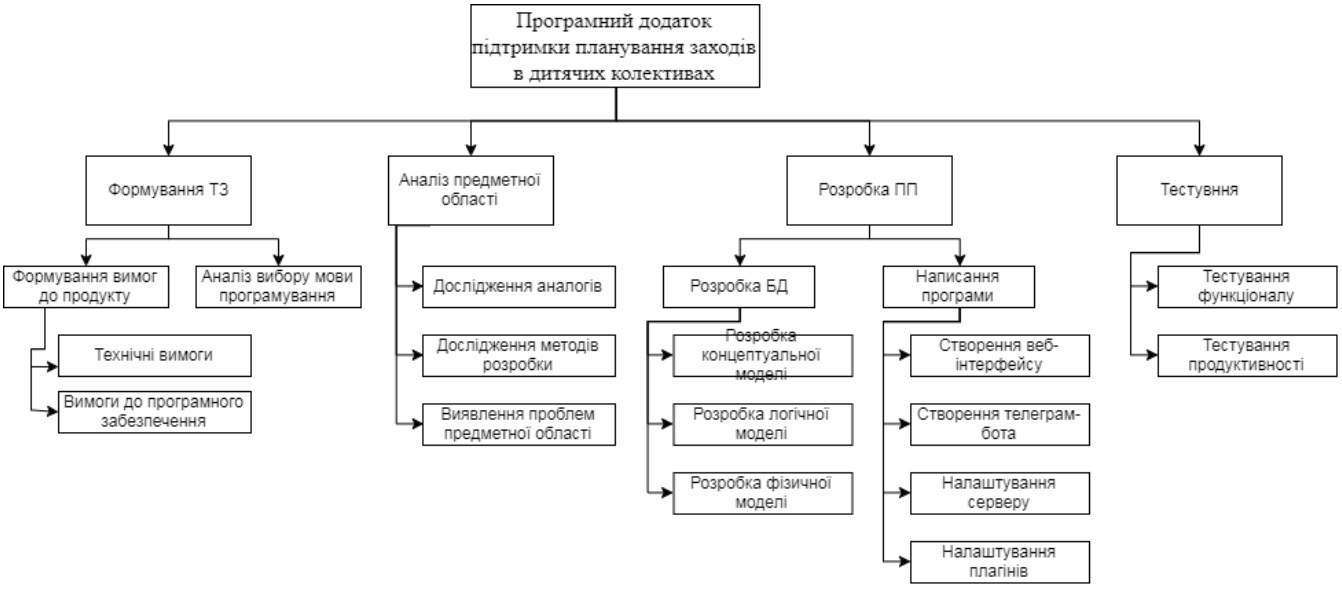

Рисунок А.1 – Ієрархічна структура робіт (WBS)

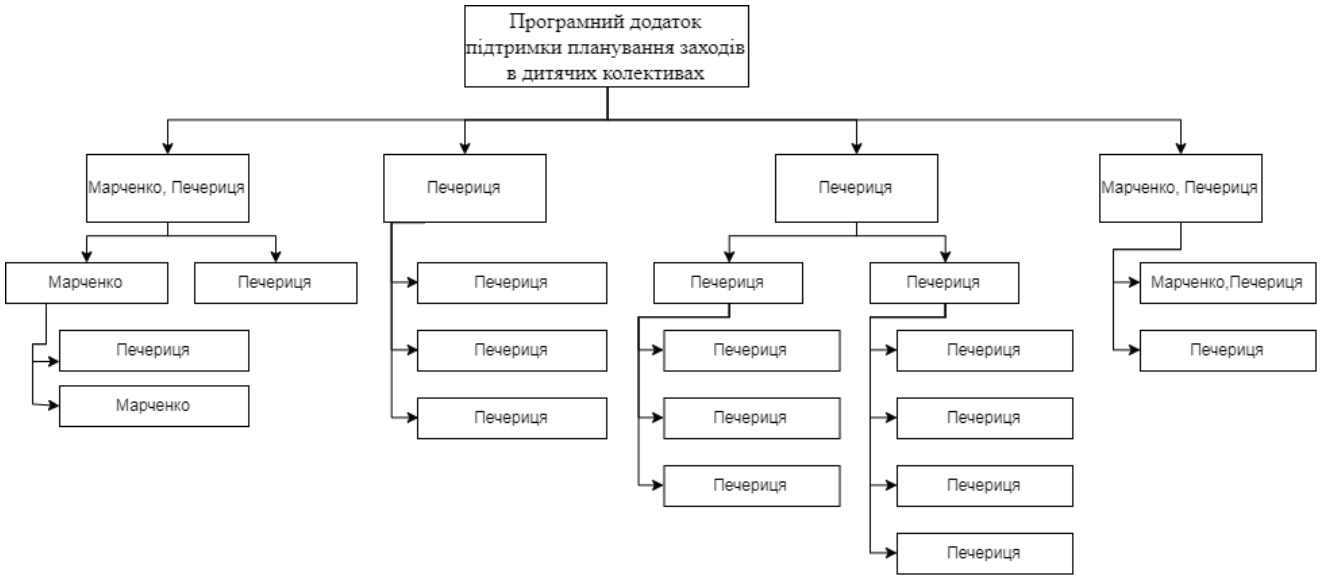

Рисунок А.2 – Структура виконавців (OBS)

А.3 Побудова календарного графіку виконання IT – проекту

Для побудови календарного графіку виконання проекту використовується діаграма Ганта, що будувалася за допомогою використання програмного продукту

MS Project 2016. Дана діаграма необхідна для того щоб задати часові межі під час розробки проекту. Діаграму Ганта можна переглянути на рисунку А.3.

А.4 Планування ризиків проекту

Під час виконання плану проекту можуть виникнути такі ризики :

- Виявлення помилок при тестуванні
- Закриття безкоштових хостингових сервісів
- Поломка техніки
- Блокування месенджера «Telegram» в Україні
- Виникнення проблем зі здоров’ям розробника

Було побудовано відповідний план дій у разі виявлення вже існуючого ризику (табл. А.1). В результаті здійснення ризиків можливе збільшення об’єму робіт по проекту, що призведе до зростання бюджету проекту та витрат на нього.

Таблиця А.1- План дій при ризиках

Ризики	**План А**	**План Б**
Виявлення помилок при тестуванні	Виправити помилки та провести тестування ще раз	-
Закриття безкоштових хостингових сервісів	Знайти альтернативні хостингові сервіси	Використовувати локальний сервер
Поломка техніки	Допрацьовувати проект на кафедрі	Купити нову техніку

Продовження таблиці А.1

Блокування месенджера «Telegram»	Використовувати VPN сервіси	-
Виникнення проблем зі здоров’ям розробника	Вилікувати розробника	Працювати в зворому стані над проектом

Після цього було побудовано таблицю ймовірності виникнення кожного ризику, а також таблицю можливих втрат від ризику.

Таблиця А.2- таблиця виникнення ризиків

	Виникнення	R1	R2	R3	R4	R5
<40%	малоймовірні			+		
40-60%	ймовірні	+			+	+
>60%	можливі		+			

	Втрати	R1	R2	R3	R4	R5
До 3 днів	мінімальні		+	+		
3- 7 днів	середні	+				
>7днів	максимальні				+	+

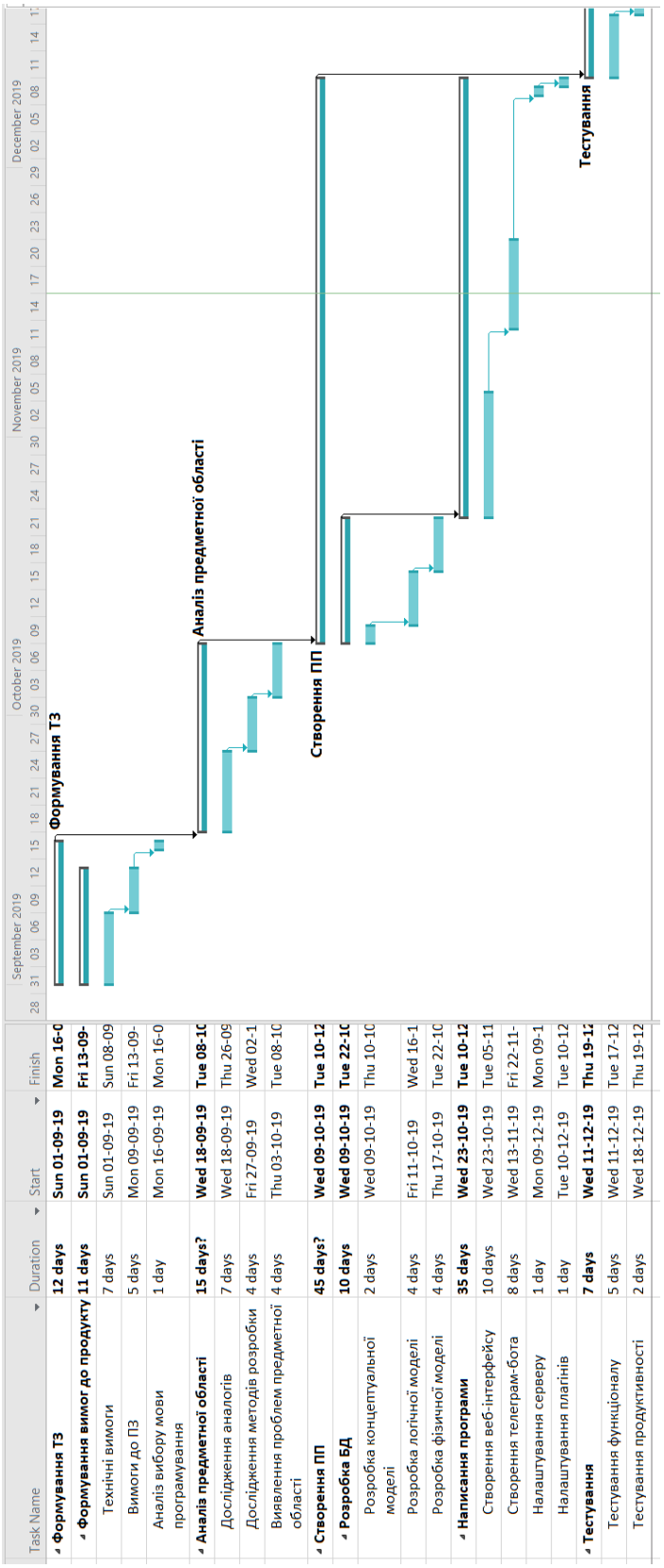

Рисунок А.3 – Діаграма Ганта

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ

Package.json

```
{
  "name": "t-bot",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint",
    "start": "node server.js"
  },
  "dependencies": {
    "axios": "^0.19.0",
    "body-parser": "latest",
    "bootstrap": "^4.4.1",
    "cookie-parser": "^1.4.4",
    "core-js": "^2.6.5",
    "debug": "latest",
    "express": "^4.17.1",
    "express-session": "^1.17.0",
    "generate-password": "^1.4.2",
    "mysql": "^2.17.1",
    "node-schedule": "^1.3.2",
    "node-telegram-bot-api": "^0.30.0",
    "parseurl": "latest",
    "serve-static": "^1.14.1",
    "vue": "^2.6.10",
    "vue-axios": "^2.1.5",
    "vue-resource": "^1.5.1",
    "vue-router": "^3.1.3"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "^3.12.0",
    "@vue/cli-plugin-eslint": "^3.12.0",
    "@vue/cli-plugin-router": "^4.0.0-alpha.4",
    "@vue/cli-service": "^3.12.0",
    "babel-eslint": "^10.0.1",
    "eslint": "^5.16.0",
    "eslint-plugin-vue": "^5.0.0",
    "vue-template-compiler": "^2.6.10"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "rules": {},
    "parserOptions": {
      "parser": "babel-eslint"
```

```
    }
  },
  "postcss": {
    "plugins": {
      "autoprefixer": {}
    }
  },
  "browserslist": [
    "> 1%",
    "last 2 versions"
  ]
}
```

Server.js

```
/* eslint-disable no-unused-vars */
// запускаю роути та сервіси //
const services = require('./core/bot_core/controller');
services.runServices(__dirname);

// анті сліп для хостінга //
let http = require("http");
setInterval(function() {
    http.get("http://vue-heroku-bot.herokuapp.com/");
}, 600000); // every 10 minutes (600000)

// запуск бота //
const runner = require('./core/bot_core/runner');
runner.run();
```

Database/Connection.js

```
const mysql = require('mysql');

var connection = mysql.createPool({
    host: 'RemoteMySQL.com',
    port: 3306,
    user: 'OtEPDwTkUU',
    password: '',
    database: 'OtEPDwTkUU',
});

module.exports = connection;
```

Database/Queries.js

```
const connection = require('./connection');

/*  *  *  *  *  *  *  *  *  *  *  *  *  */

module.exports = {
    run: function(q, callback){
        connection.query(q, function (err, result, fields) {
            if (err) throw err;
            callback(result);
```

```
        });
    },
    runVoid: function(q){
        connection.query(q, function (err, result, fields) {
            if (err) throw err;
        });
    },
    closeConnection: function () {
        connection.end();
    },
    openConnection: function () {
        connection.connect();
    }
};
```

Database/Scripts.js

```
// повертає стрінг запроса //
const query = {
    'isRegistered': function (telegram_id) {
        return 'select count(*) as \'Count\' from Users where Telegram_id = ' +
telegram_id;
    },
    'addUser': function (telegram_id, userName, userSName, userNick, userLogin,
userPass) {
        return 'insert into Users(Users.Telegram_id, Users.TelegramUserName,
Users.TelegramUserSName, \n' +
            '                Users.TelegramUserNick, Users.UserLogin,
Users.UserPass)\n' +
            'values(' + telegram_id + ', \'' + userName + '\', \'' + userSName +
'\', \'' + userNick + '\',\n'+
            ' \'' + userLogin + '\', \''+ userPass + '\')';
    },
    'updateUser': function (telegram_id, userName, userSName, userNick) {
        return 'update Users set \n' +
            '    Users.TelegramUserName = \'' + userName + '\',\n' +
            '    Users.TelegramUserSName = \''+ userSName +'\',\n' +
            '    Users.TelegramUserNick = \'' + userNick + '\'\n' +
            'where Users.Telegram_id = ' + telegram_id + ';';
    },
    'isUserExist': function (login, pass) {

        return 'select \n' +
            '    u.TelegramUserName, \n' +
            '    u.TelegramUserSName, \n' +
            '    u.User_id,\n' +
            '    u.TelegramUserNick,\n' +
            '    u.XP,\n' +
            '    l.MaxXP,\n' +
            '    l.Name\n' +
            'from Users u\n' +
            'join Levels l on l.MaxXP >= u.XP\n' +
            'where UserPass = \'' + pass + '\' and UserLogin = \''+ login +'\'\n' +
            'limit 1';
    },
    'addGroup': function (userId, name) {
        return 'insert into UserGroups(OwnerUser_id, Name)\n' +
```

```
            'values ('+ userId +', \'' + name + '\')';
    },
    'getGroupListByUser': function (userId) {
        return 'select UserGroup_id, Name from UserGroups\n' +
            'where OwnerUser_id = ' + userId;
    },
    'getProfileById': function (userId) {
        return 'select \n' +
            '    u.TelegramUserName, \n' +
            '    u.TelegramUserSName, \n' +
            '    u.User_id,\n' +
            '    u.TelegramUserNick,\n' +
            '    u.XP,\n' +
            '    l.MaxXP,\n' +
            '    l.Name\n' +
            'from Users u\n' +
            'join Levels l on l.MaxXP >= u.XP\n' +
            'where User_id = ' + userId;
    }
};

module.exports = query;
```

Bot_core/token.js

```
const TOKEN = '926365911:AAGMMZQeqdY0ZA3VYZcpwYvOOjo';
module.exports = TOKEN;
```

Bot_core/controller.js

```
module.exports = {
    runServices: function(dirname) {

        /***        ***/
        /*** ROUTES ***/
        /***        ***/

        const express = require('express');
        const runner = require('./runner');
        const bodyParser = require('body-parser');
        const serveStatic = require('serve-static');
        const path = require('path');
        const app = express();
        const session = require('express-session');
        const core = require('./core.js');
        const cookieParser = require('cookie-parser');

        app.use(bodyParser.urlencoded({ extended: false }));
        app.use(bodyParser.json());
        app.use(cookieParser());

        app.use('/', serveStatic(path.join(dirname, '/dist')));
        app.use('/about', serveStatic(path.join(dirname, '/dist')));
        app.use('/profile/:userId', serveStatic(path.join(dirname, '/dist')));
        app.use('/events/:userId', serveStatic(path.join(dirname, '/dist')));
        app.use('/stats/:userId', serveStatic(path.join(dirname, '/dist')));
```

```
app.use('/group/:groupId', serveStatic(path.join(dirname, '/dist')));
app.use('/achievements', serveStatic(path.join(dirname, '/dist')));
app.get('/odz', function(req, res) {
    res.sendFile(path.join(dirname + '/public/odz.html'));
});

// Створюю печеньку сесії
app.use(
    session({
        key: 'user_sid',
        secret: 'lavrovNashPresident',
        resave: false,
        saveUninitialized: false,
        cookie: {
            expires: 600000
        }
    })
);

// Перевірка на наявність сесії
const auth = function(req, res, next) {
    if (req.session && req.session.user)
        return next();
    else
        return res.sendStatus(401);
};

/***          ***/
/*** SERVICES ***/
/***          ***/

// Logout endpoint
app.get('/logout', function (req, res) {
    req.session.destroy();
    res.send("logout success!");
});

// Проверка поточного користувача
app.get('/checker', auth, function (req, res) {
    res.send({
        'entered': true,
        'userName': req.session.userName,
        'userSName': req.session.userSName,
        'userNick': req.session.userNick,
        'userId': req.session.userId,
        'xp': req.session.xp,
        'maxXp': req.session.maxXp,
        'levelName': req.session.levelName,
    });
});

// Проверка поточного користувача
app.use('/getprofile', function (req, res) {
    core.profile_getById(req.body.userId, function (result) {
        if(result){
            res.send({
                'entered': true,
                'userName': result.userName,
```

```
                        'userSName': result.userSName,
                        'userNick': result.userNick,
                        'userId': req.session.userId,
                        'xp': result.xp,
                        'maxXp': result.maxXp,
                        'levelName': result.levelName,
                    });
                }
            });
        });

        // Перевірка логіна і вход в сесію
        app.use('/login', function (req, res) {
            core.login_isUserExists(req.body.userLogin, req.body.userPass, function
(result) {
                if(result){
                    req.session.user = req.body.userLogin;
                    req.session.userId = result.userId;
                    req.session.userName = result.userName;
                    req.session.userSName = result.userSName;
                    req.session.userNick = result.userNick;
                    req.session.xp = result.xp;
                    req.session.maxXp = result.maxXp;
                    req.session.levelName = result.levelName;
                    res.send({
                        'entered': true,
                        'userName': result.userName,
                        'userSName': result.userSName
                    });
                } else {
                    res.send({
                        'entered': false
                    });
                }
            });
        });

        // Створити нову групу
        app.use('/addGroup', auth, function (req, res) {
            core.group_create(req.body.userId, req.body.name);
            res.send("Success!");
        });
        // Получити список груп
        app.use('/viewGroupList', auth, function (req, res) {
            core.group_getList(req.body.userId, function (result) {
                res.send(result);
            });
        });


        // Тестовий
        app.use('/log', function (req, res) {
            runner.sendKek();
            res.send({
                'item': 'key1'
            });
        });
```

```
        /***       ***/
        /*** OTHER ***/
        /***       ***/

        // якщо юзер введе неіснуючу адресу нашого домена, то йому відкриється ця
сторінка
        app.use(function(req, res){
            res.sendFile(path.join(dirname+'/public/404.html'));
        });

        // ставим порт для сервака [ СТАВИТИ В КІНЦІ ]
        const port = process.env.PORT || 8080;
        app.listen(port);
    },
};
```

Bot_core/Runner.js

```
/* eslint-disable no-console */

/* constants */
const TelegramBot = require('node-telegram-bot-api');
const TOKEN = require('./token.js');
const schedule = require('node-schedule');
const bot = new TelegramBot(TOKEN, {polling: true});
const core = require('./core.js');

const testArr = [ 307915040, 308824251, 338561527, 353284498, 366741945, 389126277,
390361397];
/* constants */

schedule.scheduleJob('0 0 6 * * *', function(){



    let date = new Date();
    let m = (date.getMonth()+1);
    let d = date.getDate();

    serv.run("SELECT concat(d.Day, '.', d.Month, '.', ifnull(d.Year,'????')) as
Date, e.Title " +
        "FROM Dates d join Events e on e.Date_id = d.Date_id where d.Month = " + m
+ " and d.Day = " + d,
        function(results) {
            let str = "Сьогодні день народження у:\n";
            if(results.length === 0){
                str = "Сьогодні немає ні у кого дня народження"
            } else {
                results.forEach(function(el){
                    str += el.Date + " " + el.Title + "\n";
                });
            }
            bot.sendMessage(366741945, str);
        });
```

```
    testArr.forEach(function(el){
        bot.sendMessage(el, 'Доброго ранку! Хорошого дня тобі!');

    });
});

function runMessageReceiver(){
    bot.on('message', function(msg) {

        switch (msg.text) {
            case undefined: console.log('Undefined value!'); break;
            case '/start': core.reg_registration(msg.from.id, msg.from.first_name,
msg.from.last_name, msg.from.username); break;
        }


        let chatId = msg.chat.id;
        if (chatId === 366741945){
            let date_ob = new Date();
            let time = date_ob.getHours() + " " + date_ob.getMinutes();
            bot.sendMessage(chatId, time);
        }
        bot.sendMessage(chatId, 'You sent something!');
    });
}
module.exports = {
    run: function(){
        runMessageReceiver();
    },
    sendKek: function () {
        let date_ob = new Date();
        let time = date_ob.getHours() + " " + date_ob.getMinutes();
        bot.sendMessage(366741945, 'WORKING ' + time);
    },
    sendTest: function () {
        bot.sendMessage(366741945, 'WORKING \u{1F604} \u{1F618}');
    }
};
```

Bot_core/core.js

```
/*
* головний клас ядра проекту, в ньому зберігаються всі головні методи
*/
const q = require('../database/scripts');
const generator = require('generate-password');
const serv = require('../database/queries');
class core{
    /* функція котра повертає значення 1 */
    static isValid(){
        return 1;
    }
    /* функція перевіряє чи надіслав користувач стікер, якщо так - повертає його ІД
*/
    static checkSticker(e){
```

```
        if (e){
            return e.file_id;
        } else {
            return null;
        }
    }

    /*** РЕЄСТРАЦІЯ ***/
    static reg_registration(telegram_id, telegramUserName, telegramUserSName,
telegramUserNick){
        serv.run(q.isRegistered(telegram_id), function(results) {
            if(results[0].Count === 1){
                serv.runVoid(q.updateUser(telegram_id, telegramUserName,
telegramUserSName, telegramUserNick));
            } else {
                let login = telegram_id;
                let pass = generator.generate({
                    length: 4
                });
                serv.runVoid(q.addUser(telegram_id, telegramUserName,
telegramUserSName, telegramUserNick, login, pass));
            }
        });
    }

    /*** ВХІД ***/
    static login_isUserExists(l, p, callback){
        serv.run(q.isUserExist(l, p), function (results) {
            if (results.length === 1) {
                let b = {
                    'userName': results[0].TelegramUserName,
                    'userSName': results[0].TelegramUserSName,
                    'userId': results[0].User_id,
                    'userNick': results[0].TelegramUserNick,
                    'xp': results[0].XP,
                    'maxXp': results[0].MaxXP,
                    'levelName': results[0].Name,
                };
                callback(b);
            }
        });
    }

    /*** СТВОРИТИ НОВУ ГРУПУ ***/
    static group_create(userId, name){
        serv.runVoid(q.addGroup(userId, name));
    }
    /*** ВИВЕСТИ СПИСОК ГРУП ЮЗЕРА ***/
    static group_getList(userId, callback){
        serv.run(q.getGroupListByUser(userId), function (results) {
            let b = [];
            for (let i = 0; i < results.length; i++) {
                let el = {
                    'id': results[i].UserGroup_id,
                    'group': results[i].Name,
                };
                b.push(el);
            }
```

```
            callback(b);
        })
    }
    /*** ВИВЕСТИ ДАНІ ПРОФІЛЯ ПО ІД ***/
    static profile_getById(userId, callback){
        serv.run(q.getProfileById(userId), function (results) {
            if (results.length !== 0){
                let el = {
                    'userName': results[0].TelegramUserName,
                    'userSName': results[0].TelegramUserSName,
                    'userId': results[0].User_id,
                    'userNick': results[0].TelegramUserNick,
                    'xp': results[0].XP,
                    'maxXp': results[0].MaxXP,
                    'levelName': results[0].Name,
                };
                callback(el);
            }
        });
    }
}

/* команда яка указує, що повинно експортуватися при імпорті файлу */
module.exports = core;
```

Main.js

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import App from './App.vue';

import About from './pages/About';
import Profile from './pages/Profile';
import Events from './pages/Events';
import Main from './pages/Main';
import Stats from './pages/Stats';
import Group from './pages/Group';
import Achievements from './pages/Achievements';

import vueResource from 'vue-resource'

Vue.config.productionTip = false;
Vue.use(VueRouter);
Vue.use(vueResource);

const routes = [
    { path: '/', component: Main },
    { path: '/about', component: About },
    { path: '/profile/:userId', component: Profile },
    { path: '/events/:userId', component: Events },
    { path: '/stats/:userId', component: Stats },
    { path: '/group/:groupId', component: Group },
    { path: '/achievements', component: Achievements }
];
```

```
const router = new VueRouter({
    routes,
    mode: 'history'
});

new Vue({
    router,
    render: h => h(App),
}).$mount('#app');
```

App.vue

```
<template>
  <div id="app">

      <Header></Header>

      <div class="content">
          <router-view></router-view>
      </div>

      <Footerok></Footerok>
  </div>
</template>

<script>

import Header from "./pages/Parts/Header";
import Footerok from "./pages/Parts/Footerok";

export default {
  name: 'app',
  components: {
      Header,
      Footerok
  }
}
</script>
```

Parts/Header.js

```
<template>
    <div>
        <div class="container-fluid bg-dark">
            <div class="row text-right pt-2 pb-2" style="color:white;">
                <div class="col-md-5 d-none d-md-block"></div>
                <div v-if="this.entered === false" class="col-4 col-md-3">
                    <div class="row">
                        <div class="col-12 text-left">login:</div>
                        <div class="col-12">
                            <input type="text" id="login" class="w-100"/>
                        </div>
                    </div>
                </div>
                <div v-else class="col-5 col-md-1"></div>
```

```
                <div v-if="this.entered === false" class="col-4 col-md-3">
                    <div class="row">
                        <div class="col-12 text-left">password:</div>
                        <div class="col-12">
                            <input type="text" id="pass" class="w-100"/>
                        </div>
                    </div>

                </div>
                <div v-else class="col user-name">
                    {{userName + ' ' + userSName}}
                </div>
                <div v-if="this.entered === false" class="col-3 col-md-1">
                    <br>
                    <button @click="login_in" class="w-100">login</button>
                </div>
                <div v-else class="col-3">
                    <button @click="login_out" class="w-100">logout</button>
                </div>
            </div>
            <div class="row pb-3" v-if="this.entered === true">
                    <router-link class="text col" to="/">Home</router-link>
                    <router-link class="text col" :to="{path:
'/profile/'+this.userId}">Profile</router-link>
                    <router-link class="text col" :to="{path:
'/stats/'+this.userId}">Stats</router-link>
                    <!--<router-link class="text col-2" :to="{path:
'/group/'+this.userId}">Group</router-link>-->
                    <router-link class="text col" to="/events/1">Events</router-
link>
                    <router-link class="text col" to="/about">About</router-link>
                    <!--<router-link to="/achievements">Achievements</router-link>-
->
            </div>
        </div>

    </div>
</template>

<script>
    import Vue from 'vue';
    export default {
        data:function() {
            return {
                entered: false,
                userName: '',
                userSName: '',
                userId: 0
            };
        },
        created: function () {
            let me = this;
            Vue.http.get('/checker').then(function (data) {
                me.entered = data.body.entered;
                me.userName = data.body.userName;
                me.userSName = data.body.userSName;
                me.userId = data.body.userId;
            });
```

```
        },
        methods: {
            login_in(){
                let login = document.getElementById("login");
                let pass = document.getElementById("pass");
                let me = this;
                Vue.http.post('/login', {
                    'userLogin': login.value,
                    'userPass': pass.value
                }).then(function (data) {
                    me.entered = data.body.entered;
                    me.userName = data.body.userName;
                    me.userSName = data.body.userSName;
                    me.userId = data.body.userId;
                    window.location.reload(false);
                });
            },
            login_out(){
                this.entered = false;
                Vue.http.get('/logout');
                window.location.href = "/";
            }
        }
    }
</script>
```

Main.vue

```
<template>
    <div>
        <div class="container mb-5">
            <div class="row">
                <div class="col-12">
                    <span class="main-head">
                        Greetings, friend!
                    </span>
                </div>
            </div>

            <div class="row">
                <div class="col-12">
                    <span class="main-head-2">
                        Is is <span style="color: firebrick">PLANERATOR</span>
application
                    </span>
                    <br>
                    <span class="main-head-2">
                        Use top bar to navigate on service
                    </span>
                </div>
            </div>

            <div class="row">
                <div class="col-12">
                    <img class="w-100" style="max-width: 400px;"
src="../assets/welcome.gif"/>
```

```
                </div>
            </div>
        </div>
        <div v-if="this.admin === true" >
            <div>
                <button @click.prevent="kekLog">log</button>
                <button @click.prevent="isSessionActive">is active</button>
                <button @click.prevent="logout">logout</button>
            </div>
        </div>
    </div>
</template>

<style scoped>
    .main-head{
        font-size: 30px;
    }

    .main-head-2{
        font-size: 22px;
    }
</style>

<script>
    import Vue from 'vue';
    export default {
        data:function() {
            return {
                admin: false
            };
        },
        methods:{
            kekLog(){
                Vue.http.get('/log', {
                    'item': 'key'
                }).then(function (data) {
                    console.log(data);
                });
            },
            isSessionActive(){
                Vue.http.get('/checker').then(function (data) {
                    console.log(data);
                });
            },
            logout(){
                Vue.http.get('/logout').then(function (data) {
                    console.log(data);
                });
            }
        }
    }
</script>
```