Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Каленик Павло Іванович

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Веб-застосунок для планування заходів під час подорожі на базі методів побудови навігаційних маршрутів

Спеціальність:  
121 – Інженерія програмного забезпечення

Спеціалізація:  
Інженерія програмного забезпечення

Керівник:

Тройніна Анастасія Сергіївна,

канд. техн. наук, доцент

Одеса – 2020

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

# **ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Каленика Павла Івановича, група АС-151

1. Тема роботи: «Веб-застосунок для планування заходів під час подорожі на базі методів побудови навігаційних маршрутів»  
Керівник роботи:Тройніна Анастасія Сергіївна, канд. техн. наук, доцент  
затверджені наказом ректора від «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р. No \_\_\_\_\_\_\_

2. Зміст роботи: аналіз вимог до програмного продукту, план виконання проекту, проектування програмної системи, програмна реалізація системи, тестування програмної системи, розгортання програмного продукту, охорона праці

3. Перелік графічного матеріалу: Згідно до слайдів презентації

4. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, Дата | |
| завдання видав | завдання  прийняв |
| 7 | доц. Москалюк А.Ю. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів дипломного  проекту (роботи) | Строк виконання етапів проекту( роботи ) | Примітка |
| 1 | Вимоги до програмної системи | 31.08.20 – 29.09.20 | виконав |
| 2 | Планування розробки програмної системи | 30.09.20 – 06.10.20 | виконав |
| 3 | Проектування веб-системи | 07.10.20 – 30.10.20 | виконав |
| 4 | Програмна реалізація | 02.11.20 – 20.11.20 | виконав |
| 5 | Тестування системи | 23.11.20 – 30.11.20 | виконав |
| 6 | Охорона праці | 01.12.20 – 7.12.20 | виконав |

**Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_\_\_\_ Каленик П.І.**

( підпис )

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ Тройніна А.С.**

( підпис )

**Зміст**

[ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 2](#_Toc58746471)

[АНОТАЦІЯ 7](#_Toc58746472)

[ВСТУП 9](#_Toc58746473)

[1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ 10](#_Toc58746474)

[1.1 Опис предметної області і аналогів системи 10](#_Toc58746475)

[1.2 Мета проведення роботи 14](#_Toc58746476)

[1.2.1 Завдання роботи 14](#_Toc58746477)

[1.4 Предмет дослідження 15](#_Toc58746478)

[2. ОСНОВНІ МЕТОДИ ПОБУДОВИ І РОЗПІЗНАВАННЯ ОПИСУ МАРШРУТУ 16](#_Toc58746479)

[2.2.1 Покрокова інструкція 17](#_Toc58746480)

[2.2.2 Зображення траєкторії 18](#_Toc58746481)

[2.2.3 Використання опорних точок 20](#_Toc58746482)

[3. СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ 22](#_Toc58746483)

[3.1 Загальні характеристики 22](#_Toc58746484)

[3.1.1 Очікувані результати роботи 22](#_Toc58746485)

[3.2 Опис функціональних вимог 22](#_Toc58746486)

[3.3 Опис алгоритмів 30](#_Toc58746487)

[3.3.1 Алгоритм зміни маршруту з метою дозаправки 30](#_Toc58746488)

[3.3.2 Алгоритм голосового введення маршруту 33](#_Toc58746489)

[3.3 Нефункціональні вимоги 34](#_Toc58746490)

[3.3.1 Вимоги надійності 34](#_Toc58746491)

[3.3.2 Вимоги до продуктивності 34](#_Toc58746492)

[3.3.3 Вимоги до середовища функціонування 35](#_Toc58746493)

[3.4 Оцінка тривалості розробки 35](#_Toc58746494)

[3.4.1 Визначення оцінки варіантів використання 36](#_Toc58746495)

[3.4.2 Оцінка технічних факторів 40](#_Toc58746496)

[3.4.3 Оцінка зовнішніх факторів 42](#_Toc58746497)

[3.4.4 Результуючі оцінки 43](#_Toc58746498)

[3.5 План проекту 44](#_Toc58746499)

[3.6 Аналіз ризиків 47](#_Toc58746500)

[4. ОПИС ПРОЕКТУВАННЯ 50](#_Toc58746501)

[4.1 Проектування архітектури системи 50](#_Toc58746502)

[4.2 Проектування структури та організації класів 51](#_Toc58746503)

[4.3 Проектування архітектури 52](#_Toc58746504)

[4.4 Проектування структури класів. 54](#_Toc58746505)

[4.5 Проектування концептуальної моделі бази даних 58](#_Toc58746506)

[4.6 Проектування інтерфейсу користувача 62](#_Toc58746507)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУВАННЯ ПЛАНУВАННЯ ЗАХОДІВ ПІД ЧАС ПОДОРОЖІ 69](#_Toc58746508)

[5.1 Набір інструментальних засобів розробки 69](#_Toc58746509)

[5.2 Інструкція розгортання 71](#_Toc58746510)

[6 ТЕСТУВАННЯ СИСТЕМИ 73](#_Toc58746511)

[6.1 Функціональне тестування 73](#_Toc58746512)

[6.2 Модульне тестування 76](#_Toc58746513)

[6.3 Експеримент 77](#_Toc58746514)

[7 ОХОРОНА ПРАЦІ 80](#_Toc58746515)

[ВИСНОВКИ 82](#_Toc58746516)

[СПИСОК ЛІТЕРАТУРИ 83](#_Toc58746517)

[ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ 85](#_Toc58746518)

[ДОДАТОК Б 91](#_Toc58746519)

# **АНОТАЦІЯ**

Пояснювальна записка до дипломної̈ роботи: 84ст., 23рис., 14табл., 2 додатки, 21 джерел.

На сьогоднішній̆ день велика кількість людей̆ користується інформаційними технологіями з метою планування подорожей̆ та проведення вільного часу.

Мета роботи полягає в скорочені часу на розробку маршруту подорожі за рахунок створення веб застосунку для планування подорожі.

Науковою новизною являється створення моделі, що охоплює стадії планування подорожі.

Для практичної апробації була виконана програмна реалізація інтернет майданчика для планування заходів під час подорожі.

Для розробки використовується мова програмування JavaScript і реляційна система управління базами даних PostgreSQL.

Ключові слова: навігаційний маршрут, призначений для користувача інтерфейс, програмне забезпечення, JavaScript, PostgreSQL.

**ANNOTATION**

Explanatory note to the thesis: 74 p., 23 figures, 14 tables, 2 annex, 21 sources.

Today, a large number of people use information technology to plan trips and spend free time.

The purpose of the work is to reduce the time to develop a travel route by creating a model for developing a travel route.

A scientific novelty is the creation of a model that covers the stages of travel planning.

For practical testing, a software implementation of the Internet platform for planning activities during the trip was performed.

JavaScript programming language and relational database management system PostgreSQL are used for development.

Keywords: navigation route, user interface, software, JavaScript, PostgreSQL.

# **ВСТУП**

Важливою задачею для цієї̈ предметної̈ області є можливість завчасно спланувати всі ключові точки, які необхідно відвідати. Існує багато різних систем, які частково вирішують цю проблему — це автомобільні навігатори, мобільні і стаціонарні платформи, веб-додатки та інші. Далі по тексту вони називаються загальним ім'ям — комп'ютерними навігаційно-картографічними системами.

Такі системи пропонують своїм користувачам можливості по знаходженню оптимального шляху з точки A в точку B і виводять, переважно на екран, інструкції по переміщенню, виконання яких проведе користувача по маршруту. Ці інструкції зазвичай не містять знайомих користувачеві орієнтирів. Введення конкретного маршруту в такі системи або неможливий (зазвичай пропонується використовувати повністю автоматична побудова), або сильно обмежений.

Можна виділити наступні недоліки існуючих навігаційно картографічних систем:

1. навігаційно-картографічні системи не враховують необхідність заправляти авто та не прокладають маршрут через заправні станції ;
2. методи введення конкретного маршруту мало функціональні або відсутні, що не дає можливості точно задати конкретний шлях;
3. Не має голосового введення маршруту подорожі.
4. Не має уже запланованих маршрутів.

# **АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ**

# **1.1 Опис предметної області і аналогів системи**

Предметна область – процес одночасного конструювання карти та орієнтації на місцевості. На поточний момент навігаційні системи для широкої аудиторії не повністю задовольняють потреби користувача, щоб проставити конкретний маршрут з проміжними точками. Часом необхідно дізнатися час у дорозі саме по конкретному маршруту. Зручним рішенням буде можливість описати маршрут в приблизному вигляді і попросити комп'ютер запропонувати найбільш підходящі під опис варіанти. Існуючі навігаційні системи для вирішення цього завдання використовують інтерфейс, в якому користувач може поставити на мапі транзитні точки, через які повинен проходити автоматично прокладений маршрут. Однак такий введення даних стає слабо актуальним у зв'язку з розвитком голосового введення інформації ( «OK Google», «Siri»), що підштовхує до розробки універсального рішення.

**1.1.2 Опис аналогів системи**

Конкуренція[1] – економічний̆ процес боротьби товаровиробників за найвигідніші умови виробництва і збуту товарів,​за отримання найбільших прибутків. Водночас – механізм стихійного регулювання виробництва в умовах вільних ринкових відносин.

Конкуренти - це все, хто робить те ж саме що і ми, або займається схожою діяльністю.

Розгляду підлягали програми:

* Apple Maps
* Google Maps
* Яндекс Карти
* яндекс.навігатор

Всі розглянуті в даному розділі навігаційно-картографічні системи являють собою клієнтські програми зі зберіганням інформації на віддалених серверах. Таким чином, програмна система ділиться на дві частини:

* серверна частина:
  + зберігає інформацію про карти, зображеннях, POI — позначає будь-яке місце інтересу (від англійського Point Of Interest): будівлі, організації, парки, пам'ятники мистецтва, пам'ятки, зупинки громадського транспорту і т.д.;
  + прокладає маршрути;
  + доступна по мережі Інтернет з будь-якої точки світу;
  + має постійно оновлювану і актуалізуються картографічну інформацію відповідно до змінами в реальному світі
* клієнтська частина:
  + зберігає локальні настройки користувача;
  + кешируєт останню отриману від сервера інформацію
  + на деякий час, певне розробником, після чого перевіряє, чи є в ній зміни і в разі їх наявності запитує у сервера оновлені дані;
  + візуалізує отримані від сервера дані в зручному для користувача вигляді;
  + здійснює локалізацію положення користувача і відображення його поточні координати;
  + додаток оновлюється рідше і вимагає повну пере установку, вироблену операційною системою в автоматичному режимі за згодою користувача.

Мобільна навігаційно-картографічна система Apple Maps від однойменної компанії Apple працює тільки на пристроях компанії, пропонує вже стандартні способи опису маршруту у вигляді зображення траєкторії і покрокової інструкції до переміщення, як це показано на ілюстраціях нижче.

Введення конкретного шляху в системі Apple Maps неможливий, є тільки функціонал по завданню стартової і кінцевої точки маршруту, далі він розраховується автоматично на серверній стороні додатку. Додавання проміжних точок трансферу, як в автомобільних навігаторах, також неможливо. Система Apple Maps не має функціоналу для персоналізації при побудові опису маршруту.

Навігаційно-картографічна система Google Maps від компанії Google реалізована для безлічі платформ, в тому числі для пристроїв фірми Apple і пристроїв на базі власної операційної системи Android. Функціонал мало відрізняється від розглянутої вище системи Apple Maps. Ті ж самі способи опису маршруту: візуальне уявлення траєкторії шляху і покрокова інструкція до переміщення. Введення проміжних точок транзиту неможливий.

Система Яндекс Карти від компанії Яндекс широко поширена в основному на території СНД. Додаток сконцентровано на надання довідкової інформації по розташуванню об'єктів і поточному стану заторів. Функціонал по формуванню опису прокладеного маршруту можливий тільки у вигляді візуального представлення траєкторії. Введення маршруту можливий тільки через початкову та кінцеву точку без функціоналу щодо його уточнення.

Мобільна навігаційно-картографічна система яндекс.навігатор від тієї ж компанії Яндекс на відміну від Яндекс Карт ближче до навігаційних рішень для автомобіля. Вона має можливість опису всього маршруту тільки за допомогою візуального представлення траєкторії шляху. Як введення використовує точки початку, закінчення маршруту і транзитні місця. Прокладання маршруту відбувається в автоматичному режимі і не може бути змінено.

Конкурентна перевага – перевага над конкурентами,​ що досягається за рахунок пропонування споживачам товарів вищої якості або завдяки низьким цінам.

Детальний аналіз ринку[2] та конкурентів наведено в таблиці 1.3

Таблиця 1.3 - Аналоги системи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал /  Програмна система | Google Maps | Apple Maps | Яндекс Карти | TripPlanner |
| Ручна побудова маршруту | + | - | + | + |
| Збереження маршруту подорожі | + | - | + | + |
| Автоматичне прокладання маршруту через заправні станції | - | - | - | + |
| Знаходження найвигіднішого місця для заправки | - | - | - | + |
| Голосове введення маршруту | - | + | + | + |

З таблиці порівняння аналогів системи можна зробити висновок, що у представлених програмних систем, у порівнянні з TripPlanner, реалізовані не всі необхідні функції. Тому було вирішено створити програму, яка задовільняла всі потреби в плануванні мандрівок.

# **1.2 Мета проведення роботи**

Мета роботи полягає в скорочені часу на розробку маршруту подорожі. Мета досягається за рахунок створення веб застосунку для планування подорожі, який реалізує побудову персоналізованих маршрутів для спрощення взаємодії людини з комп'ютером на основі введених даних про маршрут подорожі.

# **1.2.1 Завдання роботи**

Для досягнення поставленої мети необхідно послідовно вирішити такі завдання:

1. дослідити і провести аналіз існуючих навігаційно картографічних рішень;
2. дослідити існуючі методи побудови опису (виведення) маршруту;
3. розробити алгоритм для побудови маршруту подорож.

Процес створення маршруту подорожі за допомогою алгоритму, що змінює маршрут з метою дозаправки чи алгоритм, який дозволяє ввести маршрут голосом.

Продукт становить цінність для мандрівників, а саме дозволяє економити час на планування подорожі.

## **1.4 Предмет дослідження**

Предметом дослідження являється система побудови маршруту подорожі на базі методів побудови навігаційних маршрутів.

**Висновок**

В першому розділі було розглянуто наявні картографічні рішення. Було виділено наступні недоліки систем, що розглядалися: в деяких системах неможливо задати проміжні точки маршруту, системи не враховують потребу в пальному, не всі системи дають змогу голосом ввести необхідну точку маршруту.

Проаналізувавши конкурентів, можна зробити висновок, що програмна система TripPlanner має переваги порівняно з її аналогами. Це може посприяти гарному попиту на ринку.

Виходячи з отриманої̈ вище інформації̈, необхідно знайти рішення цієї проблеми. Рішення проблеми може полягати в наступному:

* дати цінність, якої немає;
* дати більше цінності;
* дати цінність за меншу вартість.

Як було зазначено вище, проблемою споживача є значна витрата грошових і часових ресурсів на формування подорожі. Це дуже обмежені ресурси, тому нам потрібно дати користувачеві можливість їх економити. Цю проблему вирішує веб-застосування, що дозволяє будувати маршрут майбутньої подорожі.

# **ОСНОВНІ МЕТОДИ ПОБУДОВИ І РОЗПІЗНАВАННЯ ОПИСУ МАРШРУТУ**

Проведений вище аналіз навігаційно-картографічних систем дозволяє виділити два існуючих методу опису маршруту і два методи розпізнавання маршруту по опису:

* методи побудови опису маршруту[4]:
  + покрокова інструкція;
  + зображення траєкторії;
* методи розпізнавання опису маршруту:
  + введення опорних точок маршруту шляхом їх розміщення мишкою на стаціонарному комп'ютері; між ними маршрут прокладається автоматично (даний спосіб дозволяє максимально точно задати траєкторію, при цьому як і раніше враховує правила дорожнього руху і ніколи не прокладає маршрут по дорозі з одностороннім рухом у зворотному напрямку, що безсумнівно є плюсом);
  + введення точок транзиту шляхом завдання їх конкретних координат або імені POI (менш зручний з точки зору уточнення складних маршрутів; частина автомобільних навігаторів, де переважно використовується метод додавання транзитних точок, підтримують їх обмежена кількість і не дозволяють повністю описати бажаний маршрут).

Далі більш детально розглядається кожен метод побудови і розпізнавання опису маршруту з урахуванням всіх виявлених мінусів і плюсів.

# **2.2.1 Покрокова інструкція**

Покрокова інструкція являє собою послідовність дій (рис.2.1), суворе чергове виконання яких приведе користувача з стартовою точки в фінішну.

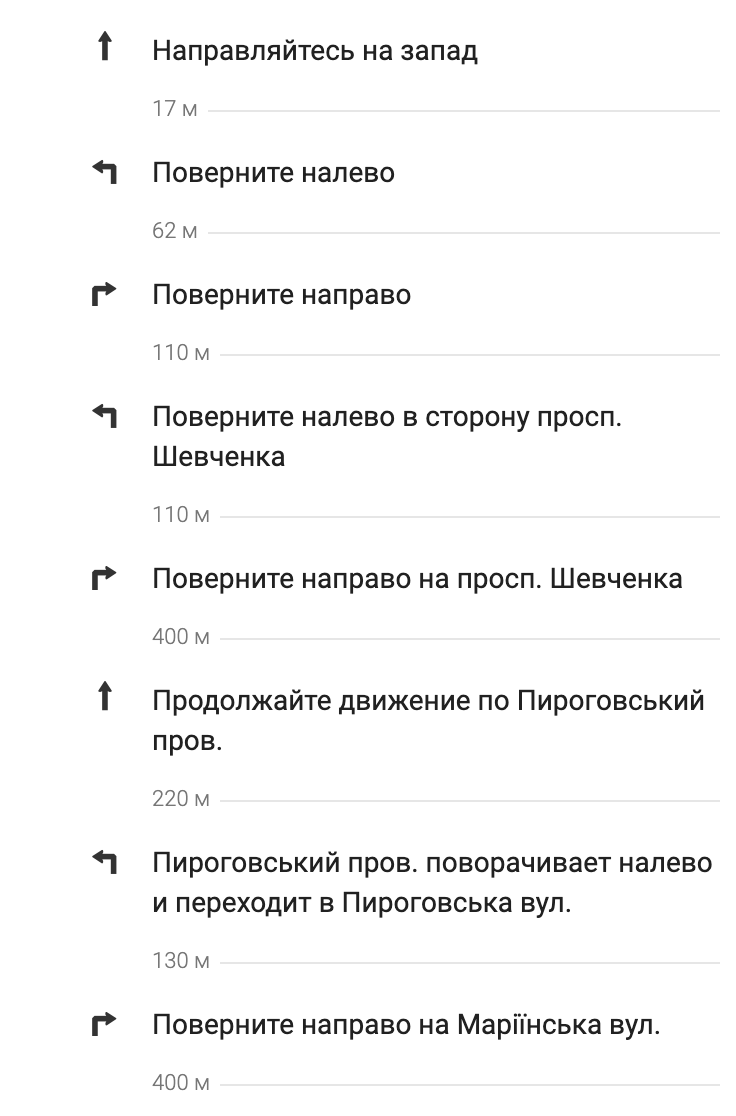


Рисунок 2.1. Опис маршруту у вигляді покрокової інструкції до переміщення в навігаційно-картографічних системах Google Maps

Даний метод опису шляху дозволяє:

* дізнатися точну послідовність дій, що приводить з точки старту в точку закінчення шляху;
* зрозуміти приблизну складність маршруту, виходячи з кількості інструкцій;
* приблизно оцінити загальну тривалість руху по маршруту, виходячи з кількості інструкцій;
* дізнатися дрібні деталі, навіть на дуже довгому маршруті.

При цьому, покрокова інструкція не дозволяє:

* точно визначити своє місце розташування на шляху;
* зрозуміти загальний напрямок руху;
* легко запам'ятати інструкцію цілком і проїхати по шляху по пам'яті;
* при русі зорієнтуватися відносно добре помітних об'єктів на шляху через відсутність відомостей про них в описі;
* зорієнтуватися в розташуванні маршруту щодо великих для нього об'єктів: міст, вулиць, будівель.

# **2.2.2 Зображення траєкторії**

Зображення траєкторії (рис. 2.2) являє собою вид на карту зверху з виділеної ламаною лінією маршруту, що з'єднує стартову і фінішну точки. Чим довше маршрут, тим менше детальним стає уявлення карти через обмеження відображають пристроїв.

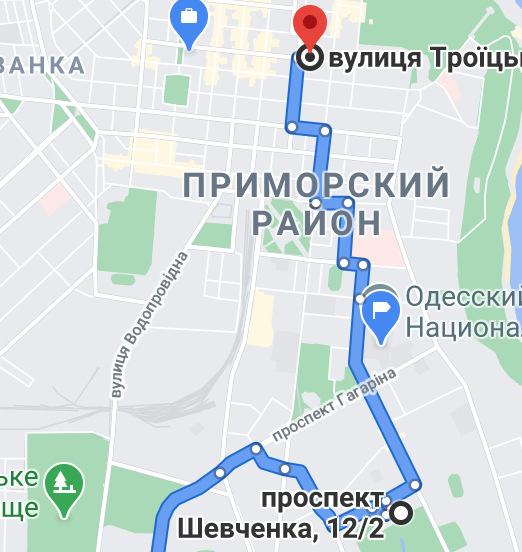


Рисунок 2.2 — Зображення траєкторії

Таким чином, виходячи з проведеного аналізу систем, зображення траєкторії дозволяє:

* зорієнтуватися в розташуванні маршруту щодо великих для нього об'єктів: міст, вулиць, будівель;

Зрозуміти загальний напрямок руху і приблизну довжину шляху. При цьому, зображення траєкторії не дозволяє:

* використовувати даний метод при тривалому русі по незнайомій місцевості, так як при довгих маршрутах пропадають дрібні деталі шляху, що роблять даний метод марним при русі по незнайомій місцевості;
* зорієнтуватися відносно добре помітних об'єктів на шляху через відсутність відомостей про них;
* дізнатися точну інформацію про проїжджаємо об'єктах, вулицях і т.д.;
* запам'ятати довгий маршрут і проїхати по ньому по пам'яті.

# **2.2.3 Використання опорних точок**

Для завдання конкретного маршруту за результатами аналізу систем були виявлені два способи, схожі один на одного:

* завдання стартовою і кінцевої точки маршруту і додавання точок транзиту, через які він повинен проходити; маршрут між усіма точками прокладається автоматично, кількість транзитних точок обмежена і тим самим не будь-який маршрут може бути заданий точно так, як того бажає користувач;
* завдання стартовою і кінцевої точки і коригування маршруту за допомогою переміщення його частин в потрібні місця з використанням комп'ютерного маніпулятора миші; додавання додаткових точок не обмежена їх числом, маршрут між усіма точками прокладається автоматично.

Таким чином, для цих двох варіантів можна виділити загальну назву - опис маршруту з використанням опорних точок. Першим мінусом даного підходу є необхідність додавання нових опорних точок в опис, якщо в автоматичному режимі маршрут був прокладений не у відповідності з бажанням користувача. Другим мінусом є сувора вимога до апаратного забезпечення і наявності графічного інтерфейсу. Третій мінус - неможливість використовувати знайомі саме цьому користувачу об'єкти і частини маршруту, за якими він часто переміщається. Іншими словами, відсутня персоналізація даних, що вводяться.

**Висновок**

В другому розділі було проаналізовано основні методи побудови і розпізнавання маршруту. Було виділено наступні способи: покрокова інструкція, використання опорних точок та зображення траєкторії.

Для використання в системі, яка розроблюється, було обрано зображення траєкторії, оскільки цей метод дозволяє простіше орієнтуватися в просторі, запам’ятати візуально, та пройти його по пам’яті.

# **СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ**

Специфікація вимог програмного забезпечення відповідно до IEEE 830 (Software requirements specification)[5] - повний опис поведінки програми, яку потрібно розробити. Зокрема, вона містить ряд сценаріїв користувачів, які описують всі варіанти взаємодії між користувачами і програмним забезпеченням.

Призначені для користувача сценарії засобом представлення функціональних вимог. Крім сценаріїв для користувача, специфікація також містить нефункціональні вимоги, які накладають обмеження на дизайн або реалізацію (такі як вимоги продуктивності, стандарти якості, або проектні обмеження).

# **3.1 Загальні характеристики**

Далі описані характеристики для проекту системи для створення веб застосунку для планування подорожей, який дозволяє, враховувати розхід пального, формувати маршрут подорожі та ділитися ним.

# **3.1.1 Очікувані результати роботи**

В результаті роботи буде реалізований веб застосунок для планування маршруту майбутньої подорожі. Застосунок включає в себе алгоритм, який дозволяє побудувати маршрут подорожі, введений голосом, та опираючись на тип транспорту в подорожі.

# 

# **3.2 Опис функціональних вимог**

Функціональні вимоги регламентують[6] функціонування або поведінку системи і відповідають на питання «що повинна робити система» в тих чи інших ситуаціях. Вони визначають основний фронт робіт розробника і визначають цілі, завдання та сервіси, що надаються програмної системою. Найбільш поширеними методами специфікації функціональних вимог є створення діаграми варіантів використання і запис сценаріїв варіантів використання.

На рис.3.1 представлено декілька типів акторів.  
Актор «Користувач» є користувачем, якому необхідно авторизуватися або зареєструватися для кикористання веб-застосування. Актор «Адміністратор» є користувачем, якому необхідно авторизуватися та мати права доступу адміністратора. Актор «Google API»[7][8] є стороннім сервісом для роботи з картою. Актор «Weather API» є стороннім сервісом для отримання прогнозу погоди. Актор Fuel API є стороннім сервісом для отримання списку автозаправних станцій та цін на пальне. На рис. 3.1 наведено основні варіанти використання та зв’язок акторів з ними.

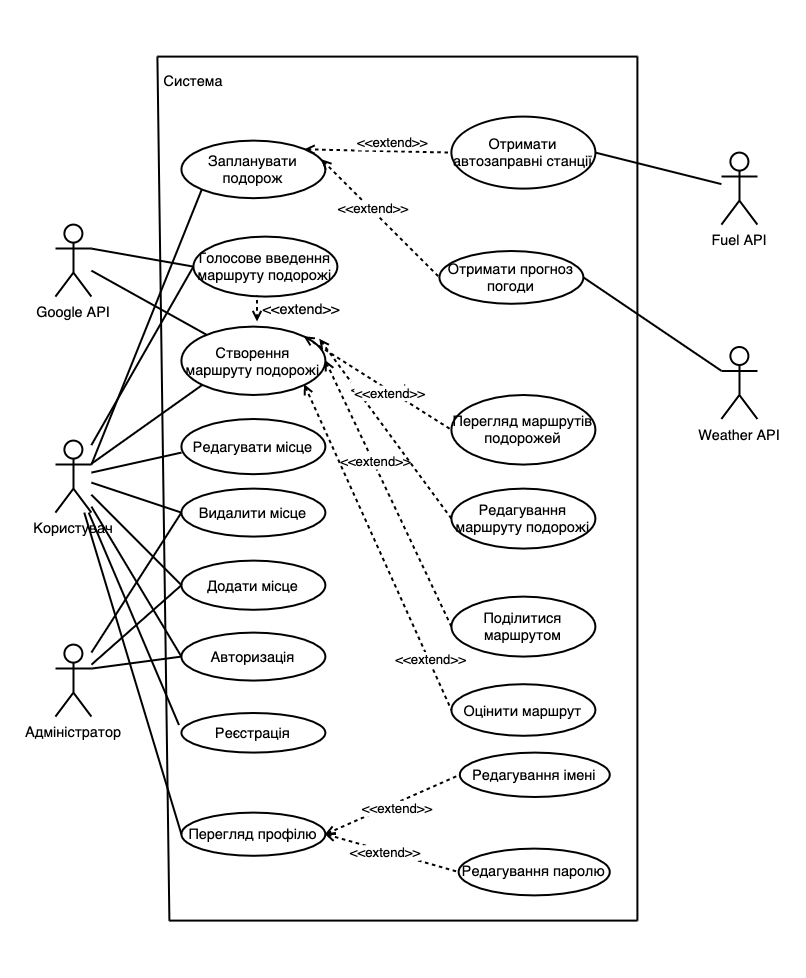


Рисунок 3.1 – Діаграма варіантів використання

Наведемо опис основних варіантів використання з рис.3.1.

Реєстрація

Актори: Користувач

Передумова: Користувач ввів логін, пароль та натиснув кнопку “Зареєструватися”

Сценарій:

1. Підключення до БД
2. Отримання інформації користувача
3. Система валідує всі введені дані
   1. Хоча б одне поле не пройшло валідацію
   2. Користувач отримує повідомлення про помилку
4. Система додає нового користувача в БД
5. Пере направлення користувача на головну сторінку

Авторизація

Актори: Користувач, Адміністратор

Передумова: Актор ввів логін та пароль

Сценарій:

1. Підключення до БД
2. Отримання логіну та паролю
3. Система валідує всі введені дані
   1. Хоча б один параметр не пройшов валідацію
   2. Система відправляє повідомлення з помилкою
4. Система порівнює отримані дані з тими, що зберігаються в БД
   1. Користувача не знайдено в БД
   2. Система відправляє повідомлення з помилкою
5. Актор перенаправляється на головну сторінку

Додати місце

Актори: Користувач, Адміністратор

Передумова: Актор залогінений

Сценарій:

1. Підключення до БД
2. Користувач обирає координату місця
3. Користувач вводить опис місця
4. Система зберігає в БД в таблиці “Places
   1. Не знайдено жодного запису в таблиці
   2. Система відправляє порожній масив
5. Система відправляє масив з місцями, їх координатою, назвою та коротким описом

Перегляд місць певної групи

Актори: Користувач

Передумова: Актор залогінений

Сценарій:

1. Підключення до БД
2. Користувач відправляє область в якій необхідно отримати точки та фільтр
3. Система робить пошук по БД в таблиці “places” по заданих фільтрах
   1. Не знайдено жодного запису в таблиці
   2. Система відправляє порожній масив
4. Система відправляє масив з місцями, їх координатою, назвою та коротким описом.

Формування маршруту подорожі

Актори: Користувач, Weather API, Fuel API

Передумова: Актор залогінений та отримав список місць в певній області

Сценарій:

1. Підключення до БД
2. Користувач обирає на карті місце, яке хотів би відвідати
   1. На карті не має жодної точки
   2. Користувач отримує повідомлення про те, що він може спочатку додати місце, а вже потім планувати подорож
3. Користувач заповнює поля дати
4. Система валідує всі дані
   1. Хоча б одне з полів не пройшло валідацію
   2. Система відправляє повідомлення з помилкою
5. Система передає Weather API координату та обрану дату
   1. Передана дата, координата є не валідними або не має прогнозу погоди на обрану дату
   2. Система повертає помилку про неможливість отримання прогнозу погоди
6. Система передає Fuel API координати маршруту
   1. Передані координати є не валідними або не має автозаправних станцій по цьому маршруту
   2. Система повертає помилку про неможливість отримання даних про вартість пального
7. Система отримує всі автозаправні станції та ціни на них
8. Система перевіряє чи вистачить пального на відрізок шляху, якщо ні, то система оновлює маршрут подорожі.
9. Система отримує прогноз погоди у JSON-вигляді
10. Система зберігає в таблиці “planned\_trips” id користувача, координату місця, прогноз погоди

Отримання прогнозу погоди

Актори: Користувач, Weather API

Передумова: Користувач залогінений та сформував хоча б один маршрут подорожі

Сценарій:

1. Користувач натискає на картці подорожі кнопку “Оновити прогноз”
2. Система відправляє Weather API координату точки та дату у вигляді JSON-об'єкту
   1. Передана дата, координата є не валідними або не має прогнозу погоди на обрану дату
   2. Система повертає помилку про неможливість отримання прогнозу погоди
3. Система отримує прогноз погоди
4. Система робить пошук по БД в таблиці “planned\_trips” подорожі по унікальному ідентифікатору
   1. Не знайдено жодного запису в БД з таким id
   2. Система повідомляє що такий запис не знайдено і повертає помилку
5. Система оновлює дані в БД таблиці “planned\_trips” в колонці “forecast”
6. Система оновлює інтерфейс користувача

Експортування нагадувань в Google-календар

Актори: Користувач, Google API

Передумова: Користувач залогінений та створив хоча б одну точку запланованої подорожі

Сценарій:

1. Користувач натискає на картці подорожі “Експортувати в календар”
2. Відкривається вікно для вводу логіну та паролю для Google аккаунту
3. Користувач заповнює всі поля та погоджується з тим, що надає свою інформацію та доступ до календаря
4. Система валідує всі поля
   1. Користувач ввів невірні дані для входу або такий користувач не зареєстровано
   2. Система повертає помилку
5. Користувач підтверджує створення нової події в календарі
   1. Користувач не підтвердив створення події або інший не опрацьований варіант події
   2. Система повертає помилку у створенні події в календарі
6. Система створює нову подію в Google календарі
   1. Google API повертає помилку в створенні нової події
   2. Система перенаправляє Користувача на голову сторінку
7. Система показує повідомлення про успішний експорт

Видалення місць

Актори: Адміністратор, Користувач

Передумова: Користувач залогінився як адміністратор або користувач створив це місце

Сценарій:

1. Підключення до БД
2. Адміністратор відправляє область, в якій необхідно отримати точки.
3. Система робить пошук по БД в таблиці “Places”
   1. Не знайдено жодного запису в таблиці
   2. Система відправляє порожній масив
4. Система оновлює інтерфейс
5. Адміністратор обирає місце на карті та натискає кнопку видалити
6. Система робить пошук по БД в таблиці “Places” по id місця
   1. Запису з таким id не знайдено
   2. Система повертає помилку
7. Система видаляє запис з таблиці “Plases” з таким id
   1. Помилка при видалені
   2. Система повертає помилку
8. Система оновлює інтерфейс

## **3.3 Опис алгоритмів**

Для вирішення задач дипломної роботи використовується:

* теорія алгоритмів[9] для опису структури алгоритму та його розробки
* розпізнавання та синтез мовлення на свої веб-сторінки.

### **3.3.1 Алгоритм зміни маршруту з метою дозаправки**

* Вхід: маршрут подорожі
* Вихід: оновлений маршрут подорожі
* Для початку необхідно отримати вид пального автомобіля, розхід на 100км та скільки літрів пального в баці. Наступним кроком алгоритму є розбиття всього маршруту на відрізки. Розбиття відбувається по точкам маршруту, який задав користувач. Далі виконується перевірка по відрізкам маршруту. Алгоритм розраховує скільки пального залишилося в баці автомобіля та перевіряє чи воно закінчиться на цьому відрізку маршруту.
* Необхідно вирахувати коли закінчиться паливо.
* Наступним кроком буде отримання всіх автозаправних станцій від початку відрізка і до точки, де планується закінчення палива.
* Серед автозаправних станцій знаходиться найвигідніша, та оновлюється маршрут подорожі. Графічна схема описаного алгоритму представлена на рис. 3.2

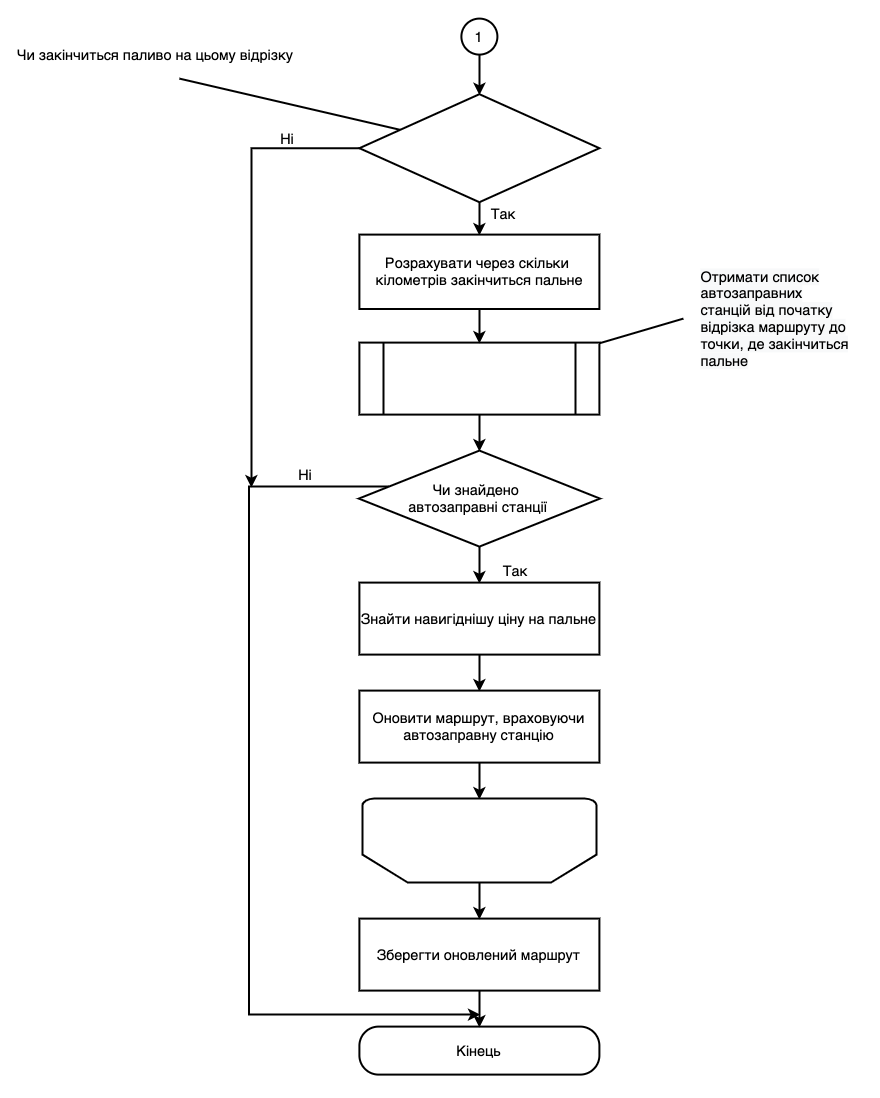


Рисунок 3.2 Алгоритм зміни маршруту з метою дозаправки

### 

### **3.3.2 Алгоритм голосового введення маршруту**

Задача алгоритму обробити введену голосом точку маршруту та додати її до подорожі.

Алгоритм використовує сторонній сервіс для розпізнавання голосу і перетворення його в текст – Web Speech API[10]. Сервіс дозволяє розробникам використовувати сценарії для генерування тексту в мовлення і використовувати розпізнавання мови як вхід для форм безперервного диктування та контролю. API JavaScript дозволяє веб-сторінкам контролювати активацію та час, а також обробляти результати та альтернативи.

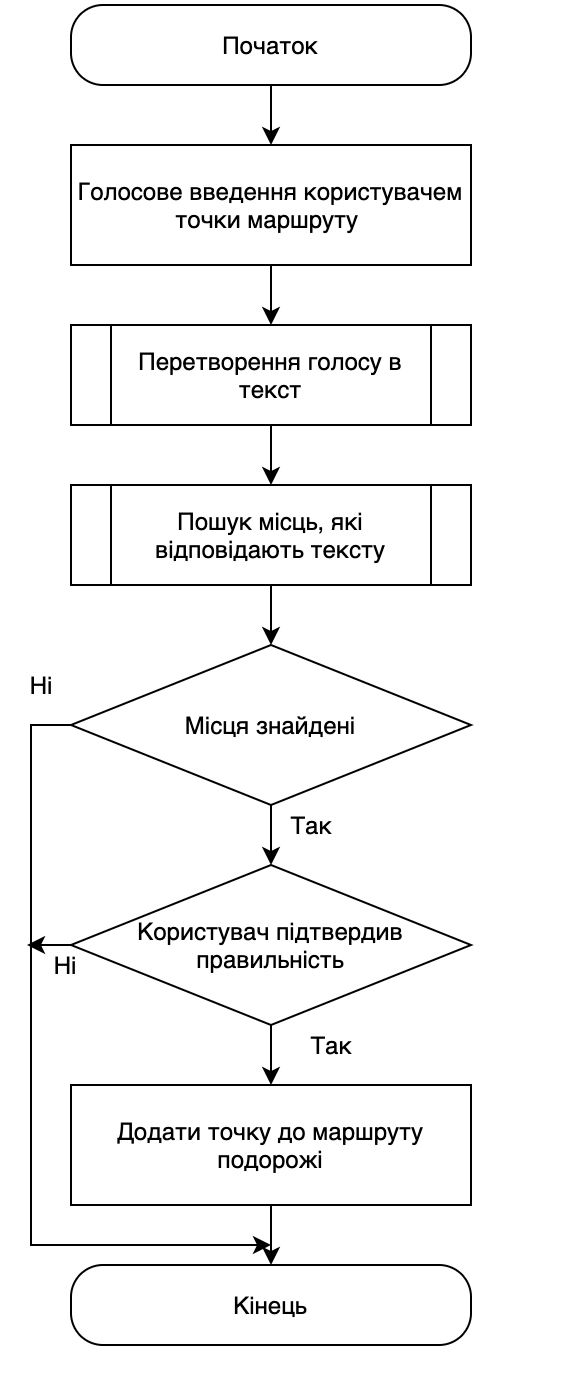


Рисунок 3.3 Алгоритм вводу маршруту голосом

## **3.3 Нефункціональні вимоги**

Нефункціональні вимоги описують обмеження, в яких виконуються функції.

Вимоги можуть бути поділені на:

1. вимоги продукту;
2. вимоги процесу;
3. зовнішні вимоги.

Ці вимоги повинні бути зрозумілими, тобто повинен бути метод перевірки виконання умов. Такі вимоги як: "зручний", "надійний", "ефективний" не можуть бути перевірені, отже, вони не відповідають формулюванню.

### **3.3.1 Вимоги надійності**

Надійність та стійкість функціонування системи забезпечуються за рахунок:

1. Перевірки полів для введення;
2. Відсутності в програмі призначеного для користувача інтерфейсу, що дозволив би йому внести такі зміни, які могли порушити роботу програмного продукту;
3. Стійкість до відмов – є доступ до базового набору операцій.

### **3.3.2 Вимоги до продуктивності**

До системи висуваються наступні вимоги з продуктивності:

1. максимальний час відгуку на запит 5 секунд;
2. час обробки операції менше 1 секунди;
3. система може обслуговувати кілька користувачів за один раз.

### **3.3.3 Вимоги до середовища функціонування**

Вимоги до апаратного забезпечення:

1. процесор 1,3 GHz та вище або мобільний процесор;
2. оперативна пам'ять об’ємом 1 Гб та вище.

Програмні вимоги:

1. операційна система MS Windows 10 / MacOS / Android/iOS;
2. наявність браузеру Google Chrome;

## **3.4 Оцінка тривалості розробки**

Параметричні методи оцінки використовують зв'язок між історичними даними та іншими параметрами, щоб отримати оцінку за допомогою математичної формули. Метод Use Case Point (UCP) розроблений в 1993 році. Він заснований на використанні для оцінки розміру програмного забезпечення прикладів з уніфікованої мови моделювання (Unified Modeling Language - UML). UCP оцінює багато елементів, такі як виконавці, технічна складність і складність середовища.

Визначення оцінки акторів: актор «Адміністратор», «Користувач» матимуть коефіцієнт 3, оскільки всі вони будуть використовувати графічний користувальницький інтерфейс. Google, Weather, Fual APIs матимуть коефіцієнт 1.

Таким чином виходить, що UAW = 2\*3 + 1\*3 = 9

Оцінимо ВВ, за кількістю класів, необхідних для реалізації. Розглянемо всі прецеденти нашої системи і на основі таблиці запишемо скільки приблизно класів буде використовуватися для реалізації того чи іншого прецеденту (табл. 2.1).

Таблиця 3.1 – Оцінка реалізації прецедентів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Нескорегована оцінка | Вага | Кількість класів | Опис |
| 1 | Простий | 5 | 3 | До 5 транзакцій |
| 2 | Середній | 10 | 5 | Від 5 до 10 транзакцій |
| 3 | Важкий | 15 | 0 | Більше 10 транзакцій |

Метод банальної оцінки варіантів використання – це метод оцінки використовується для прогнозування обсягу програмного забезпечення в проектах розробки ПЗ. Метод застосовується при використанні UML для розробки програмного забезпечення.

UUCW(UUCP) = 3\*5 + 10\*5 = 65

### **3.4.1** **Визначення оцінки варіантів використання**

Таблиця 3.2 - Визначення оцінки варіантів використання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Варіант використання | Транзакції | Кількість транзакцій | Вага | Тип |
| 1 | Вхід | Підключення до БД; Отримання інформації користувача;  Порівняння даних користувача з даними в БД. | 3 | 5 | Простий |

Продовження таблиці 3.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Варіант використання | Транзакції | Кількість транзакцій | Вага | Тип |
| 2 | Реєстрація | Отримання даних від користувача; валідація даних;  запис в БД. | 3 | 5 | Простий |
| 3 | Перегляд цікавих місць на карті з їх описом та відміткою про враження | Вибір певної області на карті з усіма, розташованими на ній, визначними точками  Запит до БД;  Оновлення інтерфейсу користувача | 3 | 10 | Середній |
| 4 | Перегляд місць певної групи | Вибір області на карті; Вибір фільтра по місцям;  Запит до БД;  Оновлення інтерфейсу користувача | 4 | 10 | Середній |

Продовження таблиці 3.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Варіант використання | Транзакції | Кількість транзакцій | Вага | Тип |
| 5 | Формування маршруту подорожі | Отримання даних від користувача;  Перевірка на правильність введення;  Запис до БД;  Оновлення інтерфейсу користувача | 4 | 10 | Важкий |
| 6 | Отримання прогнозу погоди | Запит з БД даних про подорож;  Запит до Weather API прогнозу погоди на обрану дату;  Оновлення інтерфейсу користувача | 3 | 5 | Середній |
| 7 | Отримання автозаправних станцій | Запит з БД даних про подорож; Запит до Fuel API даних про вартість палива на шляху; Оновлення інтерфейсу | 3 | 5 | Середні |

Продовження таблиці 3.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Варіант використання | Транзакції | Кількість транзакцій | Вага | Тип |
| 7 | Експорт нагадування про точки маршруту в Google-календар | Запит до БД подорожі;  Авторизація користувача в Google-календарі;  Створення нагадувань в календарі | 3 | 10 | Високий |
| 8 | Перегляд адміном останніх активних користувачів | Запит до БД останніх активних користувачів.  Оновлення інтерфейсу | 2 | 5 | Середній |
| 9 | Видалення місця | Запит до БД;  Оновлення інтерфейсу;  Видалення місця;  Запит до БД;  Оновлення інтерфейсу | 5 | 5 | Високий |

Розрахуємо нескоректована оцінку варіантів використання:

UUCW = 5\*5 + 4\*10 = 65.

Показник UCP обчислюємо за формулою:

UCP = UAW + UUCW = 8 + 55 = 63.

Тепер визначаємо UUCW і UCP, але за рахунок обчислення кількості класів, використаних в системі.

### **3.4.2 Оцінка технічних факторів**

Оцінка технічних факторів дає коефіцієнт для оцінки складності архітектури. Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 3.3 - Оцінка факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Опис фактору | Вага | Оцінка | Пояснення |
| Т1 | Розподілена система | 2 | 2 | Усі основні модулі повинні бути чітко пов’язані між собою |
| Т2 | Час відгуку | 1 | 1 | Оскільки система передбачає взаємозв'язок користувача з графічним інтерфейсом та API, тому час відгуку системи повинен бути мінімальним |
| Т3 | Ефективність кінцевого користувача | 1 | 2 | Продукт повинен бути простим і зрозумілим у використанні |
| Т4 | Складність обробки | 1 | 1 | Для шифрування інформації при передачі будуть використовуватися POST-запити та SSL сертифікат |
| Т5 | Повторне використання коду | 1 | 3 | Ключовою особливістю обраної бібліотеки є пере використання існуючих класів. |

Продовження таблиці 3.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Опис фактору | Вага | Оцінка | Пояснення |
| Т6 | Простота установки | 0,5 | 1 | Процес інсталяції повинен бути інтуїтивно зрозумілим |
| Т7 | Простота використання | 0,5 | 5 | Кінцевий продукт повинен бути простий у використанні та інтуїтивно зрозумілим для непідготовлених користувачів |
| Т8 | Портативність | 2 | 2 | Програма передбачає кросплатформовість, тому буде писатися на платформі node.js, мова програмування - javascript |
| Т9 | Простота внесення змін | 1 | 1 | Кінцевий продукт може бути модернізований в майбутньому |
| Т10 | паралельні обчислення | 1 | 3 | Паралельні обчислення можна проводити на різних кластерах або суперкомп’ютерах |
| Т11 | Вимоги до безпеки | 1 | 3 | Система повинна захищати отримані від користувача дані від зовнішніх програм та інших користувачів, та не давати доступ до API без авторизації̈ |
| Т12 | Доступ з боку зовнішніх користувачів | 1 | 1 | Система не передбачає використання її іншими зацікавленими особами. |
| Т13 | Спеціальні вимоги до навчання користувачів | 1 | 1 | Необхідно бути впевненим користувачем ПК |

Вирахуємо TFaсtor:

TFaсtor = 2\*2 + 1\*1 + 1\*2 + 1\*1 + 1\*3 + 0.5\*1 + 0.5\*5 + 2\*2 + 1\*1 + 1\*3 + + 1\*3 + 1\*1 + 1\*1 = 27

Оцінка технічного чинника обчислюється за формулою:

TCF = 0,6 + (0,01 \* TFactor) = 0,6 + (0,01 \* 27) = 0, 87

### **3.4.3 Оцінка зовнішніх факторів**

Оцінка зовнішніх факторів дає коефіцієнт для організаційних ризиків при розробці. Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 3.4 - Оцінка зовнішніх факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Опис фактору | Вага | Оцінка | Пояснення |
| F1 | Знайомство з технологією | 1,5 | 3 | Розробник не досить добре знайомий з предметною областю. |
| № | Опис фактору | Вага | Оцінка | Пояснення |
| F2 | Досвід в подібних проектах | 1,5 | 3 | Є невеликий досвід в розробці схожих проектів. |
| F3 | Досвід використання об'єктно-орієнтованого підходу | 1 | 2 | Є необхідний досвід в ООП розробці. |
| F4 | Наявність провідного аналітика | 0,5 | 1 | Немає досвіду в отриманні вимог. |
| F5 | Мотивація | 1 | 5 | Висока мотивація. |
| F6 | Стабільність вимог | 2 | 3 | Вимоги можуть змінюватися в процесі розробки проекту. |
| Продовження таблиці 3.3 | | | | |
| № | Опис фактору | Вага | Оцінка | Пояснення |
| F7 | Часткова зайнятість | -1 | 4 | Часткова занятість працівників можливо, так як проект може бути розбитий̆ на підзадачі і розрахувати так, що навіть незважаючи на часткову занятість працівників, вся необхідна робота виконається в термін |
| F8 | Складні мови програмування | -1 | 2 | Мова була вивчена задовго до початку проекту. |
| F7 | Часткова зайнятість | -1 | 4 | Часткова занятість працівників можливо, так як проект може бути розбитий̆ на підзадачі і розрахувати так, що навіть незважаючи на часткову занятість працівників, вся необхідна робота виконається в термін |
| F8 | Складні мови програмування | -1 | 2 | Мова була вивчена задовго до початку проекту. |

Проаналізувавши дані з таблиці 3.4 можна зробити висновок, що на розробку найбільше впливають досвід в подібних проектах та стабільність вимог.

Сума добутків вагових коефіцієнтів і оцінок для кожного з зовнішніх факторів визначає показник EFactor:

EFactor = 1.5\*3 + 1.5\*3 + 1\*2 + 0.5\*1 + 1\*5 + 2\*3 + (-1)\*4 + (-1)\*2 = 16.5

Оцінка зовнішнього фактора обчислюється за формулою:

EF = 1,4 + (-0,03 \* EFactor) = 1,4 + (-0,03 \* 16.5) = 0.905

### **3.4.4 Результуючі оцінки**

Для визначення тривалості розробки потрібно, знайти якій кількості робочих годин відповідає один UCP. Для цього підраховується кількість чинників з множин F1 - F8, оцінки яких за абсолютним значенням перевищують 3 (в даному випадку значення F1, F2 і F7 перевищує 3), отже, для розрахунку тривалості розробки приймається, що одному UCP відповідає 28 робочих годин.

AUCP = UCP\*TCF\*EF = 20\*(AW+UUCP)\*TCF\*EF =

= 20\*(8+65)\*0,87\*0,905=1149,5 ≈ 144 робочих дня при 8-годинній праці на добу

## **3.5 План проекту**

На даному етапі нам необхідно побудувати WBS, для того що б побачити послідовність виконання робіт, зв'язок між ними, а також час виконання кожної̈ роботи.

Зобразимо структуру декомпозиції̈ робіт в розгорнутому вигляді (рис. 3.2):

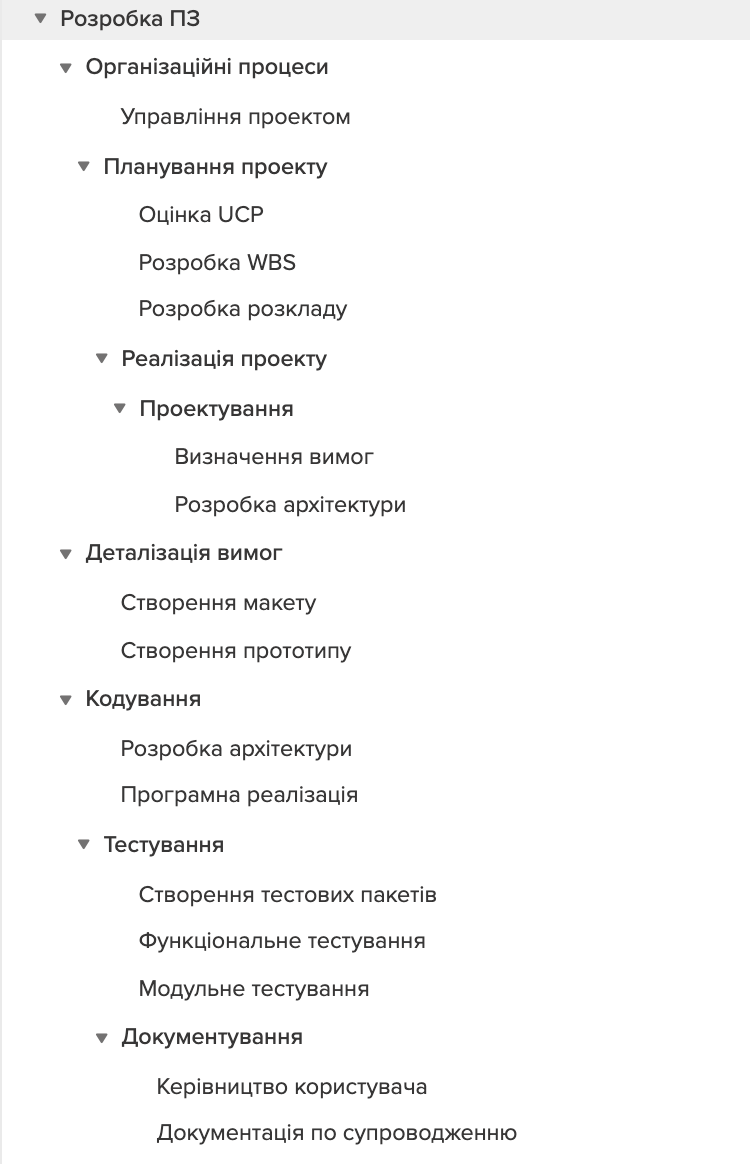


Рисунок 3.4 – Структура декомпозиції̈ робіт в розгорнутому вигляді

На основі запланованих робіт ( рис. 3.4 ) побудуємо діаграму Ганта.

Діаграма Ганта[11] – це популярний тип стовпчастих діаграм (гістограм), який використовується для ілюстрації плану, графіка робіт з якого-небудь проекту. Є одним з методів планування проектів. Використовується в застосунках з управління проектами.

Діаграма Ганта складається із смуг, орієнтованих уздовж осі часу. Кожна смуга на діаграмі представляє окреме завдання в складі проекту (вид роботи), її кінці - моменти початку та завершення роботи, її протяжність - тривалість роботи.

Ключовим поняттям діаграми Ганта є «віха» – мітка значимого моменту в ході виконання робіт, спільний кордон двох або більше завдань. Віхи дозволяють наочно відобразити необхідність синхронізації, послідовності у виконанні різних робіт. Віхи, як і інші кордону на діаграмі, не є календарними датами.

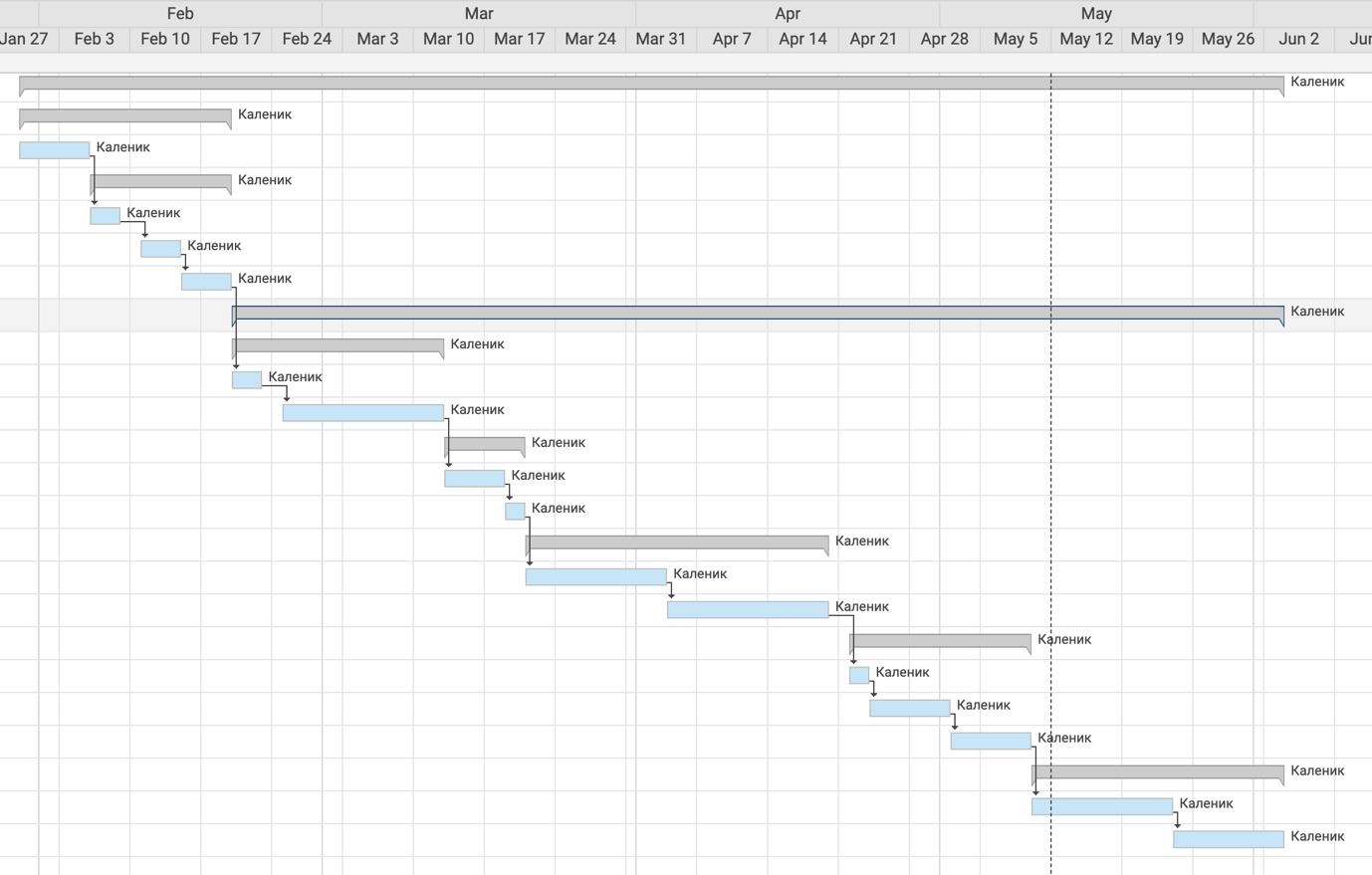


Рисунок 3.5 – Діаграма Ганта для процесу розробки ПЗ

## 

## **3.6 Аналіз ризиків**

Побудуємо діаграму Ішикави для всього проекту (рис 3.5). Дана діаграма показує чинники, що впливають на незадовільну оцінку проекту. На діаграмі описані ризики, пов’язані з низькою оцінкою дипломного проекту, а саме: нестача часу, неактуальність теми, нестача досвіду, пропущені терміни та невідповідність стандартам пояснювальної записки.

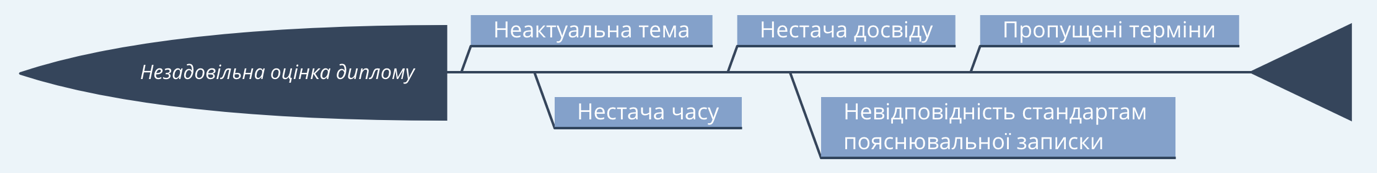


Рисунок 3.4 – Діаграма Ішикави для визначення найбільш суттєвих проблем проекту

Розглянемо детально всі фактори, що впливають на проект, а також їх підпричини.

Неактуальна тема ( рис. 3.6 )

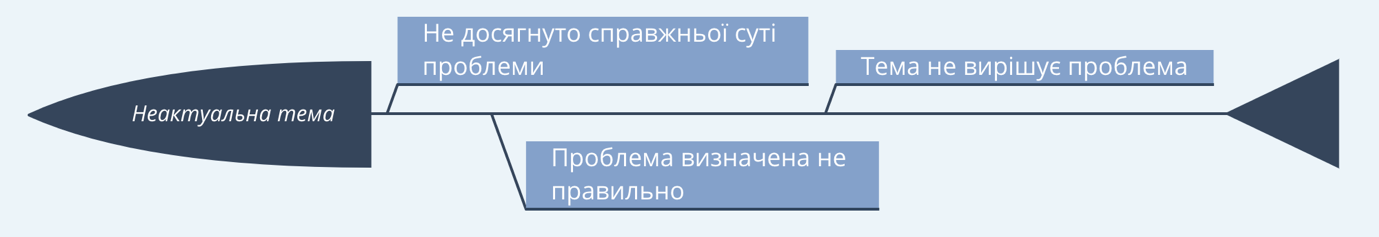


Рисунок 3.6 – Неактуальна тема

Нестача досвіду ( рис. 3.7 )

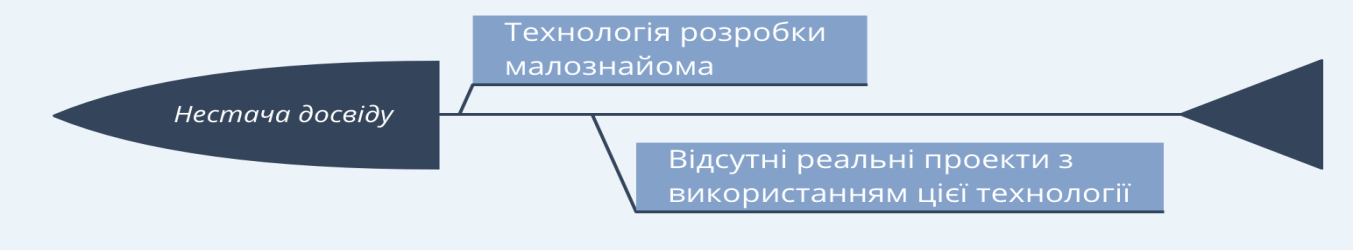


Рисунок 3.7 – Нестача досвіду

Пропущені терміни ( рис 3.8 )

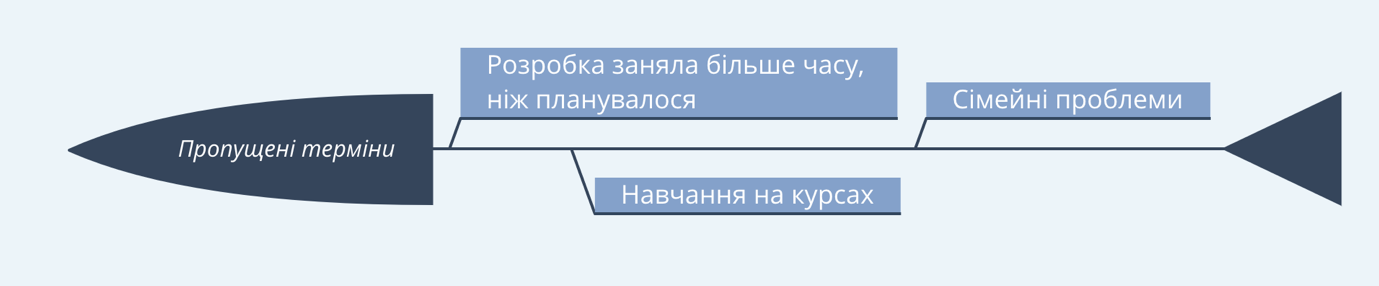


Рисунок 3.8 – Пропущені терміни

Нестача часу ( рис 3.9 )

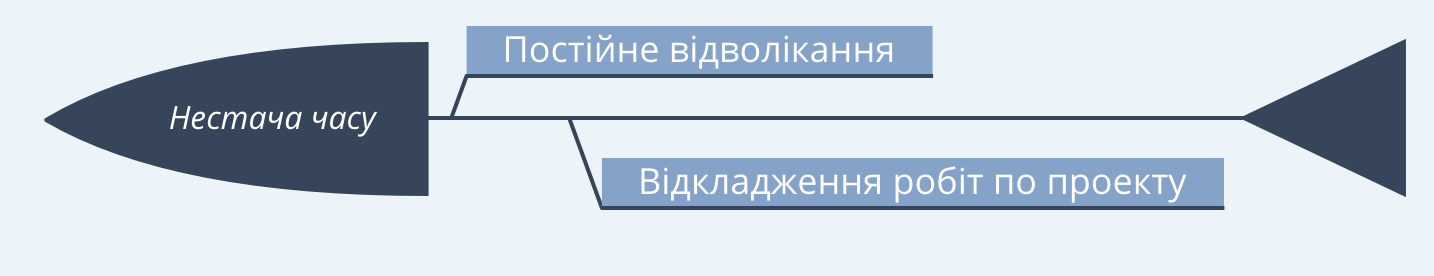


Рисунок 3.9 – Нестача часу

Невідповідність стандартам пояснювальної записки ( рис 3.10 )

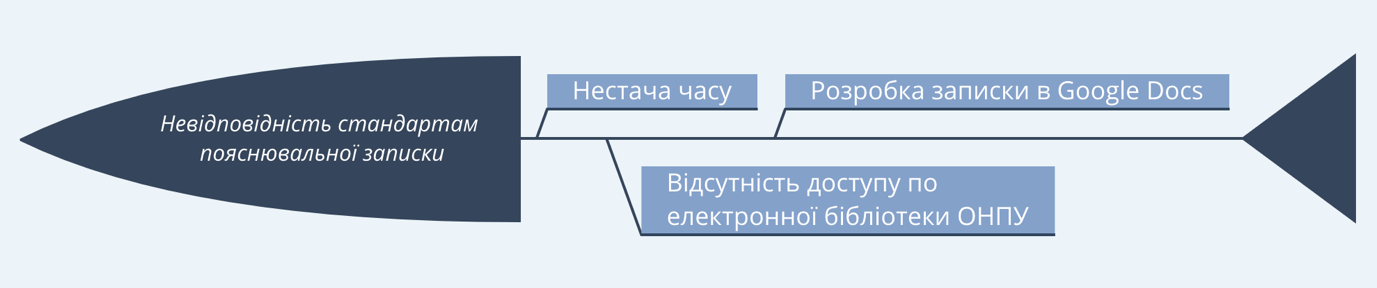


Рисунок 3.10 – Невідповідність стандартам пояснювальної записки

Таблиця 3.5 – Ризики проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ризик | Ймовірність | Вплив | Ранг |
| 1 | Неактуальна тема | 2 | 3 | 6 |
| 2 | Нестача досвіду | 3 | 3 | 9 |
| 3 | Пропущені терміни | 1 | 3 | 3 |
| 4 | Нестача часу | 3 | 2 | 6 |
| 5 | Невідповідність стандартам пояснювальної записки | 3 | 3 | 9 |

З вищенаведеної таблиці можна зробити висновок, що найбільший ризик під час розробки проекту становить: “Невідповідність стандартам пояснювальної записки” та “Нестача досвіду”

**Висновок**

В третьому розділі було описано функціональні вимоги до системи. Для опису вимог було використано діаграму функціональних вимог. Були визначені які актори містить система, описано сценарії варіантів використання системи. Проведено аналіз алгоритмів, які використовує система.

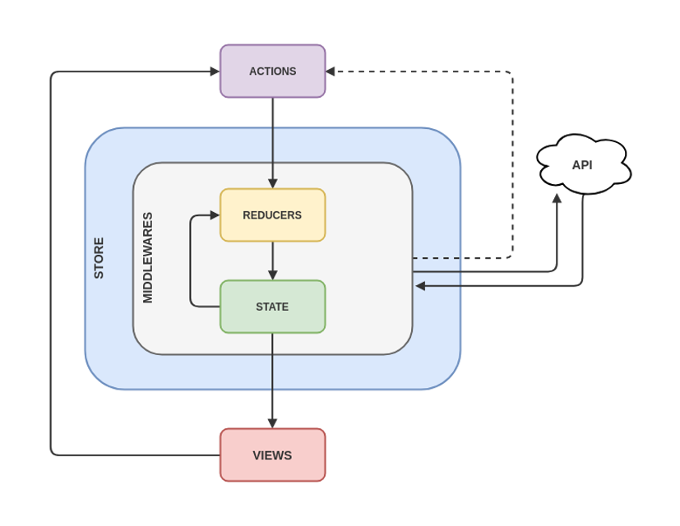
Проведено аналіз ризиків проекту за допомогою діаграми Ішикави. Основними факторами, що можуть вплинути на розробку проекту є нестача досвіду та невідповідність стандартам пояснювальної записки.

Було оцінено тривалість розробки проекту. Вона становить 144 дні. Розроблено діаграму Ганта для процесу розробки ПЗ.

# **ОПИС ПРОЕКТУВАННЯ**

## **4.1 Проектування архітектури системи**

Виходячи з наших функціональних вимог, було розбито програмну систему на структурні компоненти і залежності між компонентами. Залежність показує що один компонент надає сервіс, необхідний іншому компоненту. Модель програмної системи показано на рис. 4.1.

Рисунок 4.1 – Модель архітектури

Клієнтська частина веб-застосування розділена на три частини, а саме:

* view – відображення даних. До View відносяться всі компоненти інтерфейсу користувача: форма логіну/реєстрації, мапа для вводу точок маршруту і т.д.
* reducer – тимчасове сховище даних, які неможливо перезаписати, а лише створити знову. В ньому зберігаються локальні данні інтерфейсу. Наприклад, інформація про залогіненого користувача;
* action – інструмент для зміни даних в reducer. створює нове значення в об’єкті reducer.
* middleware – перехоплює кожен action, який відправляє запит на сервер та передає результат в action. Action, в свою чергу, створює нове значення в об’єкті reducer.

Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб−сторінку і відправляє її клієнту по мережі з використанням протоколу HTTP. Cервер взаємодіє з рівнем бази даних.

## **4.2 Проектування структури та організації класів**

Наведемо приклади діаграми послідовностей. Користувач обирає за допомогою посилань один із компонентів – Users, Places чи Trips. В свою чергу іде запит від Action до одного з контролерів – Places, Users, Trips. На цьому етапі контролер робить запити до бази даних та повертає відповідь.

Action отримує відповідь та, якщо необхідно, зберігає в Reducer для подальшого оновлення інтерфейсу.

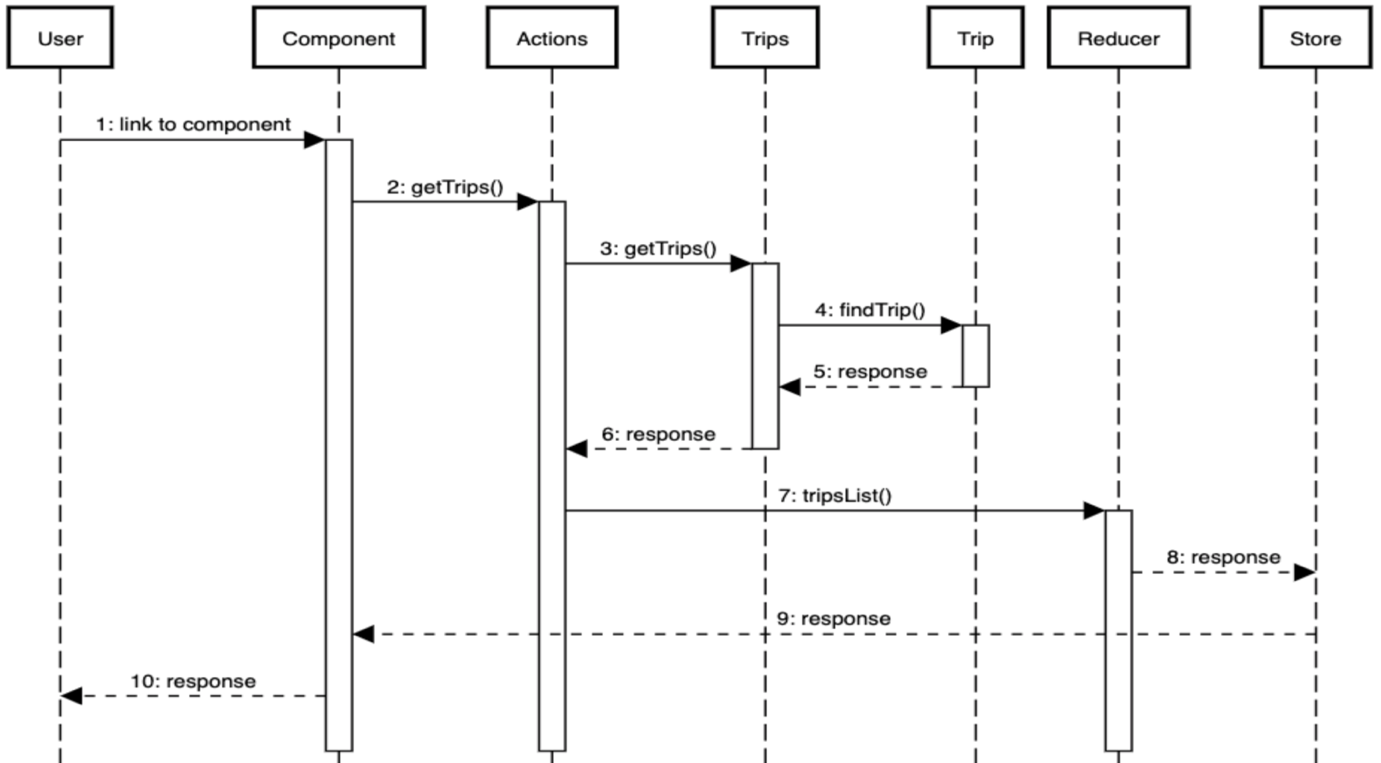


Рисунок 4.2 – Діаграма послідовностей для «Отримання подорожей»

# **4.3 Проектування архітектури**

Опис архітектури проектованої системи доцільно виконати з використанням діаграми компонентів.

Діаграма компонентів (component diagram) відображає ієрархію підсистем, структурних компонентів і залежності між ними. фізичними компонентами виступають бази даних, виконувані файли, додатки, бібліотеки, інтерфейси ІС тощо. При використанні діаграми компонентів для відображення внутрішньої структури компонентів, інтерфейси складеного компонента делегуються в певні інтерфейси внутрішніх компонентів.

Основними цілями побудови діаграм компонентів:

* визначення архітектури проектованої системи;
* побудова концептуальної і фізичної моделей баз даних;
* представлення структури вихідного і специфіки виконуваної коди системи;
* багаторазове використання певних фрагментів
* програмного коду.

Основними компонентами проектованої системи можна виділити

клієнтську частину, серверну частину (попередня обробка запитів), рівень бізнес-логіки, і рівень сховищ даних - сховище для зберігання даних пов'язаних з мандрівками і сховище для зберігання активності користувача. Зазначені компоненти зображені на рис. 4.3.



Рисунок 4.3 - Діаграма компонентів веб-застосунку планування заходів під час подорожі

# 

# **4.4 Проектування структури класів.**

Діаграма класів[12] є найбільш вживаним інструментом при моделюванні програмної системи. Вони є однією з форм статичного опису системи з точки зору її проектування і показують її структуру. Діаграма класів не відображає динамічну поведінку об'єктів зображених на ній класів.

Діаграма класів (design class diagram) ілюструє специфікації програмних класів і інтерфейсів в застосуванні. Діаграма класів може відбивати різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також типи їх зв'язку. Зазвичай на таку діаграму виноситься наступна інформація:

- класи, асоціації та атрибути;

- інтерфейси з їх операціями і константами;

- методи;

- інформація по типи атрибутів

- способи навігації;

- залежності.

На відміну від діаграми класів з моделі предметної області,

Діаграма класів проектування відображають визначення програмних сутностей, а не поняття предметної області.

В системі було виділено наступні класи: Action, Component, Reducer – вони є узагальненими. Кожен React компонент повинен мати свої власні класи Action, Reducer, Component. Клас Store – це множина всіх класів Reducer. Задача класу Route полягає в тому, щоб визначити який із класів Users, Places чи Trips повинен бути викликаний. Клас Component відповідає за відображення користувацького інтерфейсу та відстеження дій користувача: натискання на кнопки чи заповнення полів. Зміни в класі Store викликають зміну класу Component. Діаграма класів показана на рис 4.4

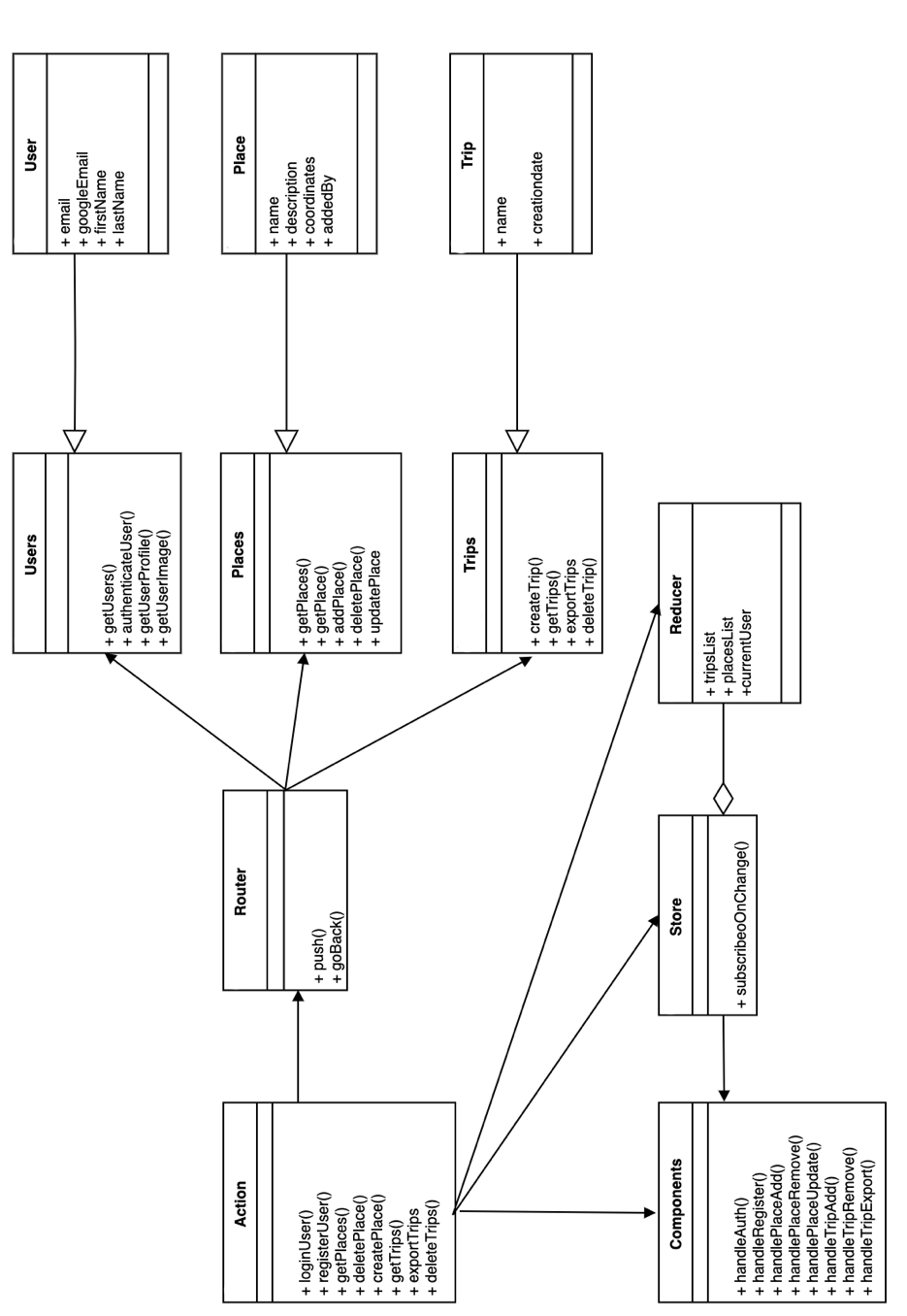


Рисунок 4.4 – Діаграма класів для TripPlanner

У таблиці 4.1 зображено опис методів класів.

Таблиця 4.1 – Опис методів класів проекту

|  |  |
| --- | --- |
| **Метод** | **Призначення** |
| createUser | Метод класу Users, який створює користувача |
| authenticateUser | Метод класу Users, який авторизує користувача |
| getUserProfile | Метод класу Users, який повертає профайл користувача |
| getUserImage | Метод класу Users, який повертає аватар користувача |
| getPlaces | Метод класу Places, який повертає список місць |
| getPlace | Метод класу Places, який повертає інформацію по одному місцю |
| addPlace | Метод класу Places, який створює нове місце |
| deletePlace | Метод класу Places, який видаляє одне місце |
| updatePlace | Метод класу Places, який оновлює інформацію по місцю |
| createTrip | Метод класу Trips, який створює нову подорож |
| getTrips | Метод класу Trips, який отримує список подорожей |
| deleteTrip | Метод класу Trips, який видаляє подорож |
| exportTrip | Метод класу Trips, який відправляє запит на створення нагадування в Google календарі |
| tripsList | Метод класу Reducer, який вказує як змінюються стани в редюсері при отриманні списку подорожей |
| placesList | Метод класу Reducer, який вказує як змінюються стани в редюсері при отриманні списку місць |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| **Метод** | **Призначення** |
| currentUser | Метод класу Reducer, який вказує як змінюються стани в редюсері при профайлу користувача |
| loginUser | Метод класу Action, який відправляє запит до серверу на авторизацію користувача |
| registerUser | Метод класу Action, який відправляє запит до серверу на реєстрацію користувача |
| getPlaces | Метод класу Action, який відправляє запит до серверу на отримання списку місць |
| getPlace | Метод класу Action, який відправляє запит до серверу на отримання детальної інформації по певному місцю |
| deletePlace | Метод класу Action, який відправляє запит до серверу на видалення місця |
| createPlace | Метод класу Action, який відправляє запит до серверу на створення місця |
| getTrips | Метод класу Action, який відправляє запит до серверу на отримання списку подорожей |
| exportTrips | Метод класу Action, який відправляє запит до серверу про успішний експорт нагадування |
| deleteTrip | Метод класу Action, який відправляє запит до серверу на видалення подорожі |
| handleAuth | Метод класу Component, який викликає Action та передає введені дані користувача |
| handleRegister | Метод класу Component, який викликає Action та передає введені дані користувача |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| **Метод** | **Призначення** |
| handlePlaceAdd | Метод класу Component, який викликає Action та передає введені дані |
| handlePlaceRemove | Метод класу Component, який викликає Action та передає идентифікатор місця |
| handlePlaceUpdate | Метод класу Component, який викликає Action та передає ідентифікатор подорожі і введені дані |
| handleTripAdd | Метод класу Component, який викликає Action та передає введені дані |
| handleTripRemove | Метод класу Component, який викликає Action та передає ідентифікатор подорожі |
| handleTripExport | Метод класу Component, який викликає Action та передає ідентифікатор подорожі |

## **4.5 Проектування концептуальної моделі бази даних**

Проаналізувавши інформацію, що потрібно зберігати у системі, були виділені наступні сутності: користувач, місце, подорож, заплановане місце та останні активні користувачі.

Основні визначні місця веб-застосування отримує з Google API, проте база даних веб-застосування містить таблицю з всіма місцями, які були добавлені користувачами.

Побудова реляційної моделі даних ґрунтується на тому, що будь-який набір даних може бути представлений у вигляді відношення, який оформлюється, але формі таблиці, де дані представляються атрибутами і значеннями на перетині відповідного атрибута із записом (кортежом).

На основі цих сутностей була розроблена модель бази даних, що також містить відношення сутностей та представлена на рисунку 4.5

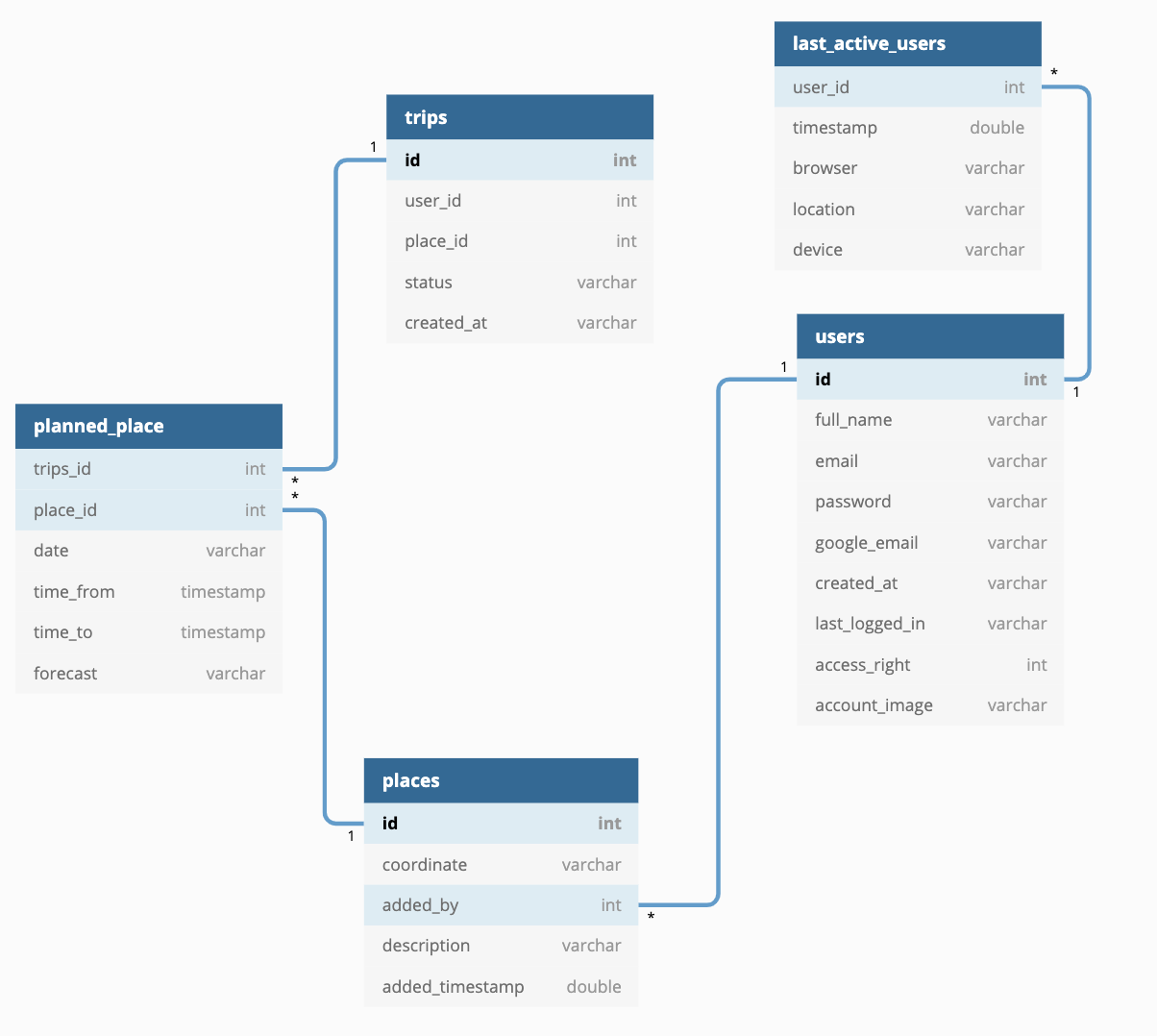


Рисунок 4.5 – Схема бази даних

Представлена схема бази даних є реляційною. Основною перевагою реляційних баз даних є те, що можливо легко масштабувати систему. Дані з формально описаних таблиць можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних. Має табличний спосіб вистави даних, а на зовнішньому рівні, що задається набором однорідних таблиць. Кожний об'єкт записується рядком у таблиці. Рядок називається записом. Запис складається з полів  різного типу.

Сутність – користувач, повністю описує дані будь-яких користувачів системи табл. 4.2

Таблиця 4.2 – Опис полів таблиці «Користувачі»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Концептуальна назва | | | | Користувачі |
| Фізичне найменування | | | | Users |
| Опис атрибутів | | | | |
| Концептуальне ім'я | Фізичне ім'я | Тип | Формат (байт) | Коментар |
| Ідентифікатор | Id | Ціле | 4 | Первинний ключ |
| Ім’я | full\_name | Текстовий | 6 |  |
| Електрона пошта | Email | Текстовий | 6 |  |
| Пароль | Password | Текстовий | 10 | Зберігається в зашифрованому виді |
| Google пошта | google\_email | Текстовий | 6 |  |
| Дата створення | created\_at | Текстовий | 6 | Обов’язкове |
| Рівень доступу | status | Ціле | 2 | Обов’язкове |
| Останній вхід | last\_logged\_in | Ціле | 6 |  |
| Аватар користувача | account\_image | Текстовий | 10 |  |

Сутність – місце, зберігає інформацію про всі місця, які відомі системі.

Таблиця 4.3 – Опис полів таблиці «Місця»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Концептуальна назва | | | | Місця |
| Фізичне найменування | | | | Events |
| Опис атрибутів | | | | |
| Концептуальне ім'я | Фізичне ім'я | Тип | Формат (байт) | Коментар |
| Ідентифікатор | Id | Ціле | 4 | Первинний ключ |
| Ідентифікатор користувача | coordinate | Текстовий | 6 | Обов'язкове |
| Ідентифікатор місця | added\_by | Ціле | 4 | Обов'язкове. Зовнішній ключ на таблицю Користувачів |
| Статус | added\_timestamp | Ціле | 4 | Обов'язкове |
| Дата створення | description | Текстовий | 20 |  |

Сутність – подорожі описує дані усіх подорожей, які відомі системі.

Таблиця. 4.4 – Опис полів таблиці «Подорожі»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Концептуальна назва | | | | Подорожі |
| Фізичне найменування | | | | Trips |
| Опис атрибутів | | | | |
| Концептуальне ім'я | Фізичне ім'я | Тип | Формат (байт) | Коментар |
| Ідентифікатор | Id | Ціле | 4 | Первинний ключ |
| Ідентифікатор користувача | User\_id | Ціле | 4 | Обов'язкове. Ідентифікатор мандрівника |
| Дата створення | created\_at | Текстовий | 5 | Обов'язкове |
| Статус експорту | status | Ціле | 4 | Обов'язкове. Подорож вже експортована в Google-календар |
| Ідентифікатор місця | place\_id | Ціле | 4 | Обов'язкове |

Наступна таблиця (табл 4.5) описує залежність таблиць «Подорожі» та «Місця». Таблиця описує в якій подорожі буде відвідуватися певне місце

Таблиця 4.5 – Опис полів таблиці «Заплановане місце»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Концептуальна назва | | | | Заплановане місце |
| Фізичне найменування | | | | planned\_place |
| Опис атрибутів | | | | |
| Концептуальне ім'я | Фізичне ім'я | Тип | Формат (байт) | Коментар |
| Ідентифікатор місця | place\_id | Ціле | 4 | Обов’язкове |
| Ідентифікатор подорожі | trips\_is | Ціле | 4 | Обов’язкове |
| Дата | date | Текстовий | 10 | Обов'язкове |
| Прогноз погоди | forecast | Текстовий | 10 |  |
| Початок | Time\_from | Час | 4 | Обов’язкове |
| Кінець | Time\_to | Час | 4 | Обов’язкове |

Таблиця «Останні активні користувача» зберігає інформацію про те які користувачі користувалися веб-застосуванням.

Таблиця 4.6 – Опис полів таблиці «Останні активні користувачі»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Концептуальна назва | | | | Користувачі що користувалися веб-застосуванням |
| Фізичне найменування | | | | Users\_at\_events |
| Опис атрибутів | | | | |
| Концептуальне ім'я | Фізичне ім'я | Тип | Формат (байт) | Коментар |
| Ідентифікатор користувача | Id\_user | Ціле | 4 | Первинний ключ |
| Дата | timestamp | Ціле | 4 | Обов’язковий |
| Браузер | browser | Текстовий | 6 |  |
| Місце знаходження | location | Текстовий | 6 |  |
| Пристрій з якого зайшли | device | Текстовий | 6 |  |

## **4.6 Проектування інтерфейсу користувача**

Після відкриття веб-сторінки користувач може авторизуватися або зареєструватися у системі.

Після успішної авторизації користувач отримує сторінку з основною інформацією (риc. 4.6). Зверху знаходиться заголовок. Якщо користувач увійшов через Google, то є можливість персонально його привітати та використати його фотографію.

Нижче заголовка користувач бачить карту з його місцем знаходження (синій маркер) та визначними місцями поблизу. Зеленими маркерами відмічені місця, які були додані іншими користувачами «TripPlanner», а червоними – вами.

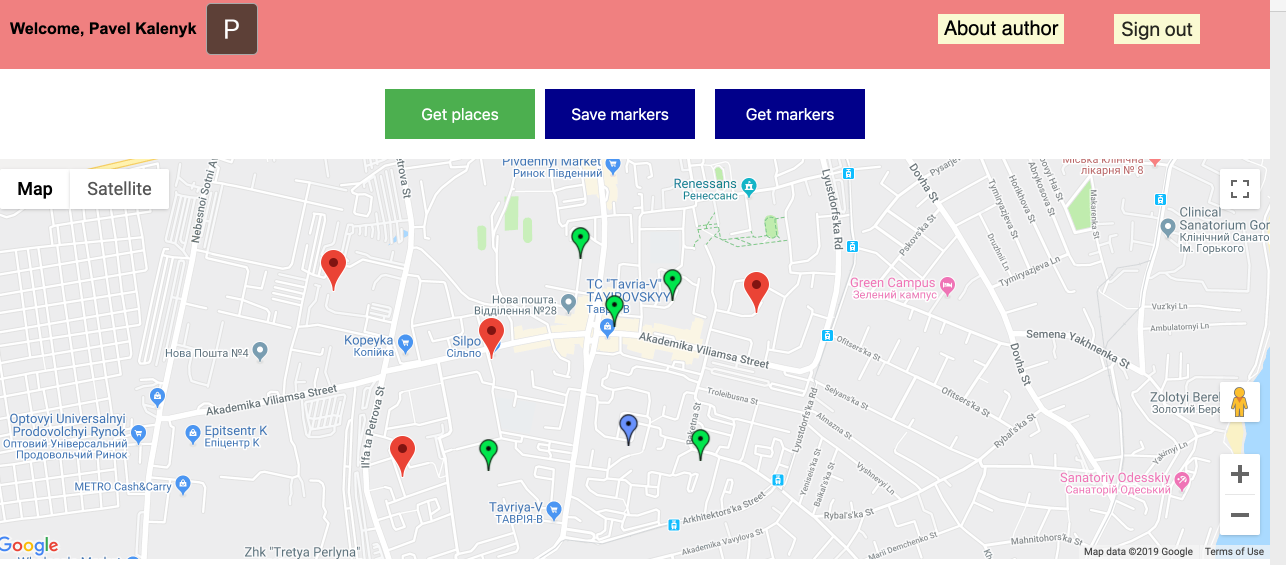


Рисунок 4.6 – Інтерфейс залогіненого користувача

Основні данні цікавих точок отримуються з Google API – це дозволяє одразу отримати велику базу цікавих місць. Якщо ж користувач не знайшов необхідного місця, то є можливість самостійно його створити.

Щоб додати свою точку, користувач повинен клікнути в будь-якому місці карти після чого перед користувачем з’явиться модальне вікно з полями для вводу заготовку, опису та коментаря ( рис. 4.7)

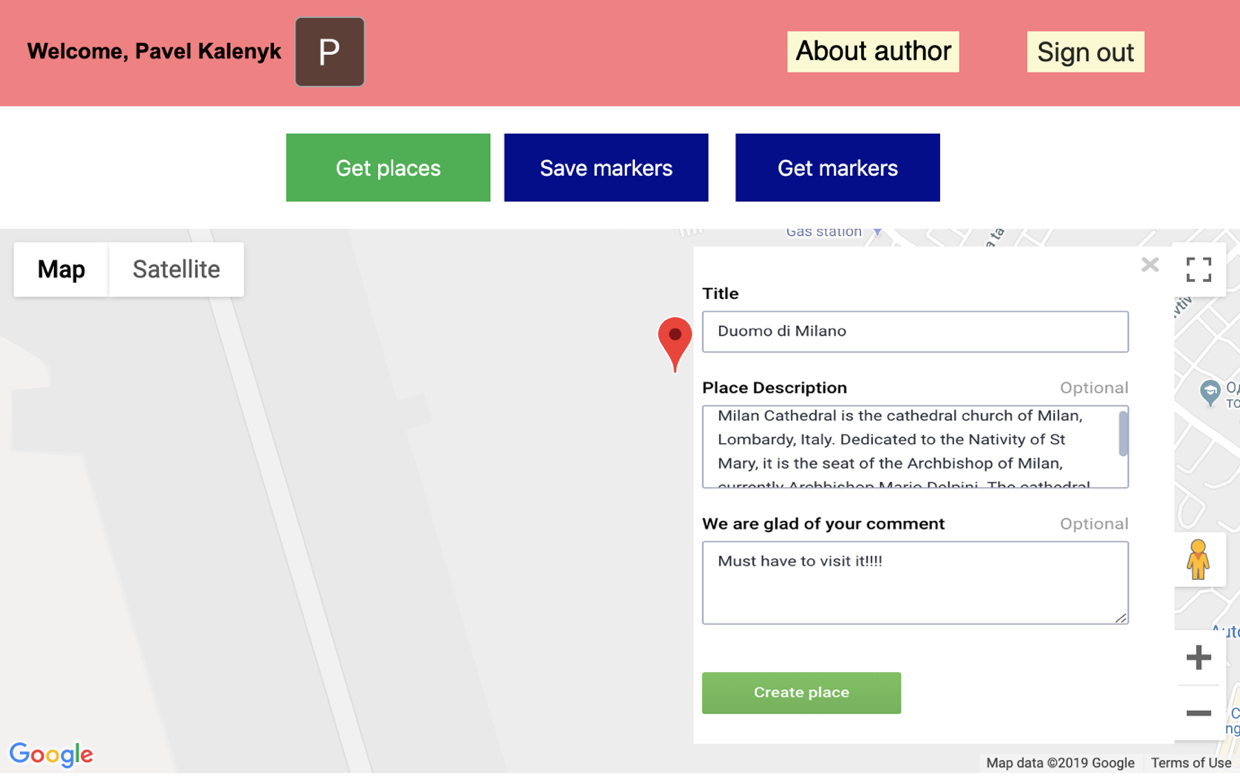


Рисунок 4.7 – Створення нового місця на карті

Щоб отримати більш повну інформацію про місце, необхідно натиснути на маркер. Поруч з’явиться модальне вікно з фотографією, коротким описом та кнопкою «Додати до маршруту» (рис. 4.8)

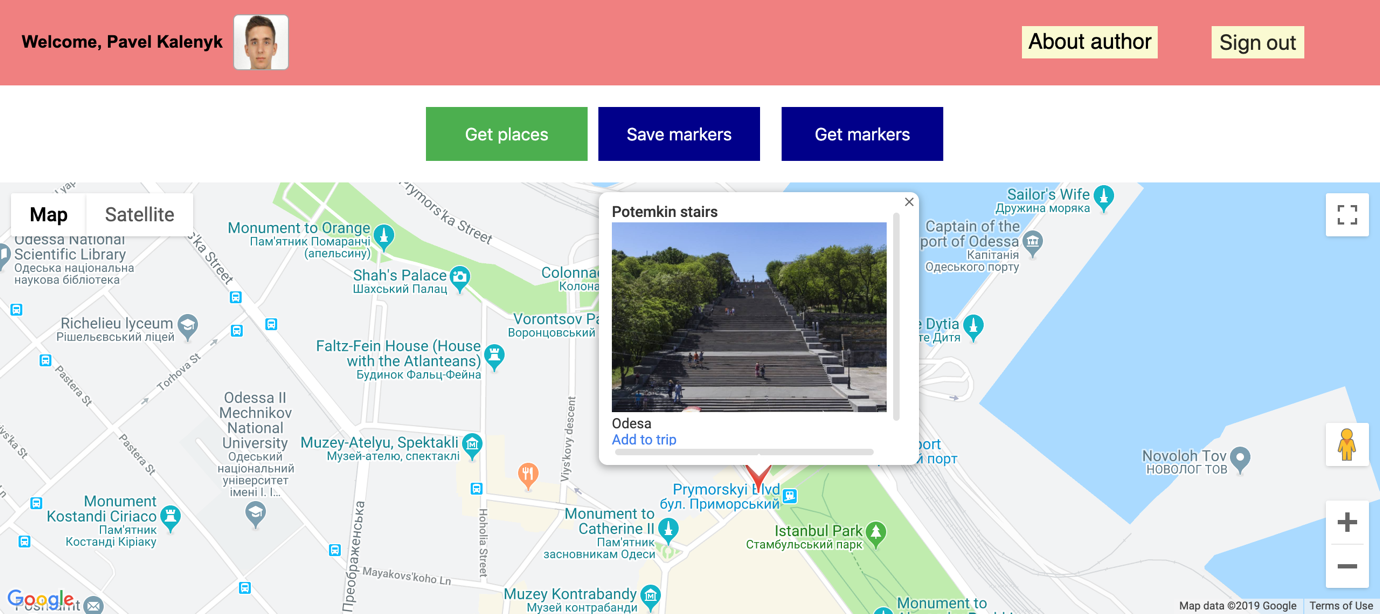


Рисунок 4.8 – Отримання повної інформації про обране місце

Після натискання «Додати до подорожі» відкривається діалогове вікно. В вільному полі необхідно ввести назву подорожі, а в календарі та на годиннику відмітити коли і скільки часу планується на відвідування місця



Рисунок 4.9 – Додавання точки до маршруту подорожі

Праворуч показаний календар з уже доданими точками та часом, який виділено на маршрут. Ліворуч користувач може додати точку в «Обране», додати до маршруту, при цьому вказавши скільки часу займе відвідування місця.

На рис. 4.10 зображена подорож з місцями, які необхідно відвідати. На картці кожної зупинки подорожі вказано часовий проміжок – скільки часу буде витрачено на цю точку маршруту, дату та прогноз погоди в вигляді картинки.

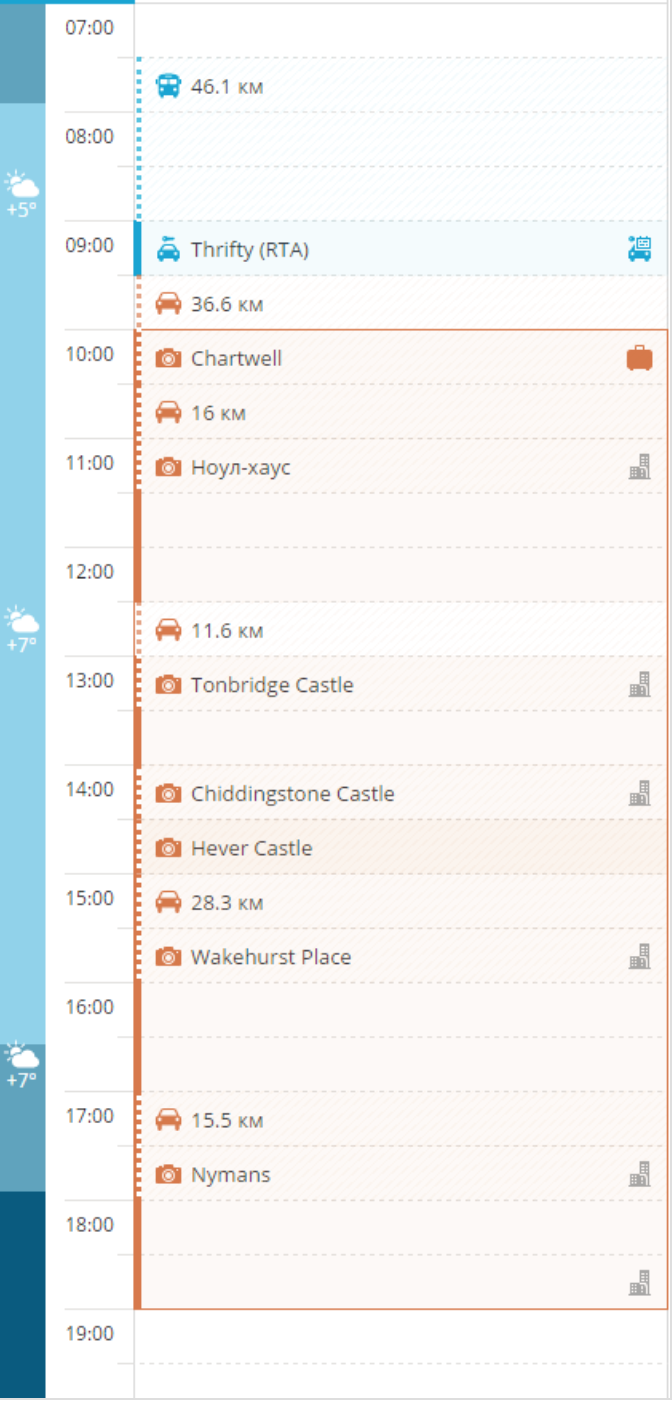


Рисунок 4.10 – Список запланованих місць

Якщо користувач обрав вид транспорту — автомобіль, при цьому вказавши тип, розхід та залишок пального, система побудує маршрут через автозаправні станції з найбільш вигідною ціною за пальне (рис. 4.11).

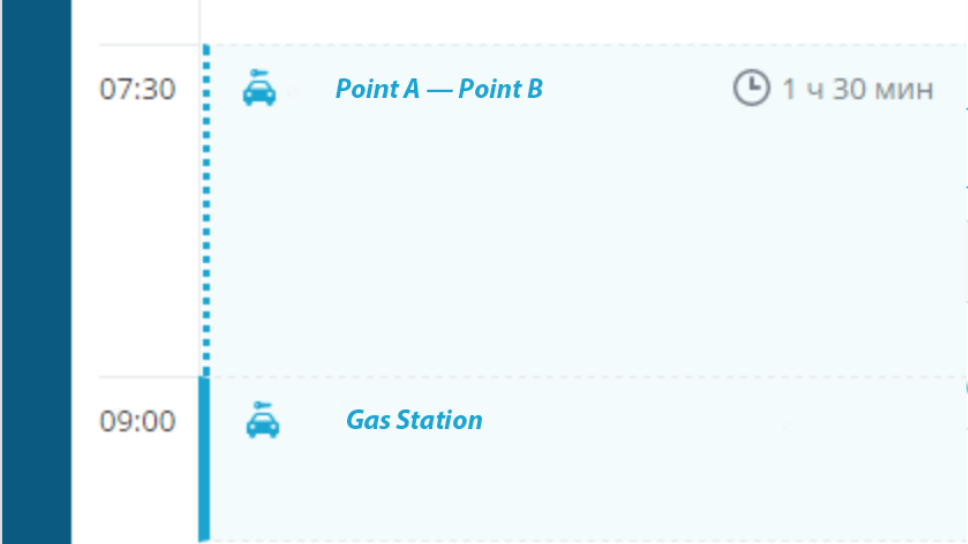


Рисунок 4.11 – Зміна маршруту з метою дозаправки

Зверху в інтерфейсі залогіненого користувача знаходяться фільтри, по яким можливо отримати місця певної групи на карті, наприклад найближчих банкоматів (рис 3.14), що може стати туристу в нагоді.

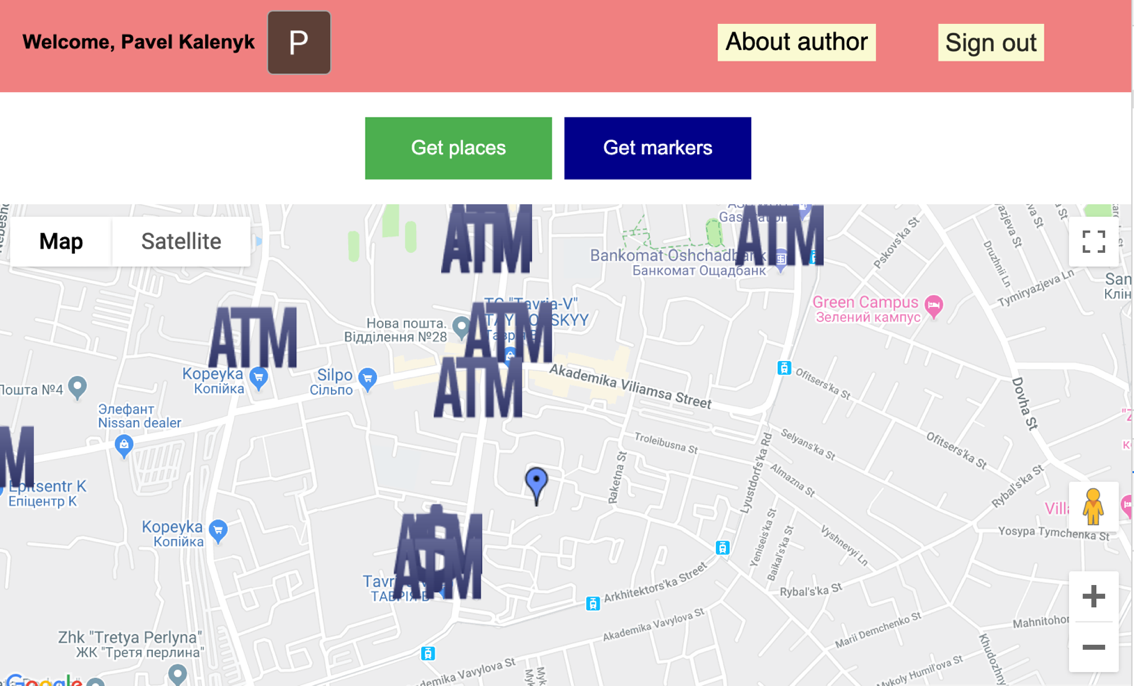


Рисунок 4.12 – Список всіх найближчих банкоматів

**Висновок**

В четвертому розділі було спроектовано архітектуру системи. Створено діаграму класів. Було описано діаграму послідовностей для отримання списку подорожей.

Також було спроектовано концептуальну модель бази даних. Було виділено наступні сутності: «Користувачі», «Місця», «Подорожі», «Заплановане місце», «Останні активні користувачі»

Спроектовано користувацький інтерфейс.

# **5 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУВАННЯ ПЛАНУВАННЯ ЗАХОДІВ ПІД ЧАС ПОДОРОЖІ**

Після того, як отримано детальний проект з компонентами, в яких визначена структура і методи взаємодії, можна братися за реалізацію компонентів, тобто створення і тестування програмних модулів, визначених при проектуванні. Для отримання каркаса розробки можна скористатися case-засобами для створення коду програмних класів на певній мові програмування. різні мови програмування вимагають від програміста певного рівня уваги до деталей при реалізації, результатом чого часто буває компроміс між простотою розробки і продуктивністю роботи програмного продукту.

# **5.1 Набір інструментальних засобів розробки**

Для програмної реалізації була обрана мова JavaScript[13][14], платформа Node.js[15].

JavaScript — динамічно, об’єктно орієнтована мова програмування. Основною перевагою зв’язки React.js[16] та Node.js є програмування цих компонентів використовуючи одну й ту ж саму мову програмування. Ще один з надвеликих плюсів Node.js — це його, майже абсолютна, модульність.

1. React - JavaScript-бібліотека з відкритим вихідним кодом для розробки призначених для користувача інтерфейсів. React розробляється і підтримується Facebook, Instagram і співтовариством окремих розробників і корпорацій. React може використовуватися для розробки односторінкових і мобільних додатків. Його мета - надати високу швидкість, простоту і масштабованість.
2. Redux - бібліотека для JavaScript з відкритим вихідним кодом, призначена для управління станом додатки. Найчастіше використовується в зв'язці з React або Angular для розробки клієнтської частини. Містить ряд інструментів, що дозволяють значно спростити передачу даних сховища через контекст.

Так у нас використовуються додатково такі основні компоненти:

1. Express.js[17] — парсер запитів та основний інструмент для будування REST API;
2. Passport.js[18] — модуль який реалізує сериалізацію та десириалізацію користувача в та з сесії за допомогою обраної бази даних або локально.

У даному проекті використовується PostgreSQL для зберігання даних. Реляційна система керування базою даних, яка є одною з найпоширеніших систем керування базами даних. Використовується як наша основна база даних, яка зберігає усі дані про задачі, користувачів та інші дані;

Для взаємодії з картою, місцями та календарем використовується Google API.

Maps JavaScript API– дозволяє додати карту на свій веб-сайт, надаючи зображення та локальні дані з того ж джерела, що й у Картах Google. Створити карту відповідно до потреб. Віртуалізувати власні дані на карті, інтерактивно взаємодіяти зі світом за допомогою перегляду вулиць.

Places API – дає можливість отримувати дані з тієї ж бази даних, що використовується у Картах Google. Місця містять понад 100 мільйонів підприємств і визначних місць, які часто оновлюються за допомогою списків, підтверджених власником, та внесків, керованих користувачами. Пошук місця може бути виконаний на основі розташування користувача або рядка пошуку. Можливо отримати детальну інформацію про певне місце, включаючи відгуки користувачів. Автозаповнення можна використовувати для автоматичного заповнення імені та / або адреси місця під час введення.

Open Weather API – дозволяє отримувати прогноз погоди, який формується на основі даних більш, ніж 40000 тисяч метеостанцій по всьому світу. Для доступу необхідно в GET-запиті вказати індивідуальний ключ, який розробник отримує при реєстрації. В запиті параметрами передаються: широта та довгота; одиниці вимірювання; вигляд, у якому приходять дані – JSON або HTML форматі.

Fuel API — дозволяє отримувати вартість палива та координати автозаправних станцій. Для доступу необхідно в GET-запиті вказати індивідуальний ключ, який розробник отримує при реєстрації. В запиті параметрами передаються: координати відрізку шляху, вид палива, радіус пошуку та валюта. Дані приходять в JSON вигляді.

## **5.2 Інструкція розгортання**

Всі налаштування системи проводяться на операційній системі Linux. Наступні команди необхідні для коректної роботи веб-застосування для планування подорожей:

1) Встановлення Node.js: sudo apt-get install nodejs

2) Встановлення npm: sudo apt-get install npm

3) Встановлення PgAdmin

4) Встановити git: sudo apt-get install git

5) Зклонувати базу даних, замінивши при цьому ім’я користувача, пароль: mysqldump -u remoteusername -p remotepassword –h kalenyk.diploma.ua databasename > dump.sql mysql -u localusername -p localpassword diplomaDataset < dump.sql

6) Клонування проекту з репозиторію git clone

https://github.com/kalenyk/diploma.git

7) Перехід до папки проекту та встановлення основних модулів Node.js необхідних для роботи сервера cd ~/diploma; npm install

8) Встановлення усіх необхідних клієнтських біблиотек bower install

9) Запуск Node.js серверу node app.js Після усіх проведених вище операцій у користувача буде робочий Node.js сервер, який слухає порт 3000.

# **6 ТЕСТУВАННЯ СИСТЕМИ**

## **6.1 Функціональне тестування**

Функціональне тестування[19] розглядає заздалегідь вказане поведінку і ґрунтується на аналізі специфікацій функціональності компонента або системи в цілому. Функціональні тести ґрунтуються на функціях, виконуваних системою, і можуть проводитися на всіх рівнях тестування. Як правило, ці функції описуються у вигляді варіантів використання системи.

Розробимо тестові сценарії[20] для тестування функціональності системи на основі варіантів використання таблиця 6.1

Таблиця 6.1 – Тестові сценарії

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порялок дій | Очікуваний результат | Фактичний результат |
| 1 | Користувач відправив усі необхідні параметри у запит | Зробити запит POST /signup. | Створений запис користувача у базі даних, користувач переадресован на сторінку авторизації. | Пройдено |
| 2 | Користувач відправив усі необхідні параметри у запиті | 1) Зробити запит POST /login | Для користувача створена сесія, переадресован до основної робочої панелі | Пройдено |

Продовження таблиці 6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порядок дій | Очікуваний результат | Фактичний результат |
| 3 | Користувач авторизований у системі або додав API ключ до заголовку запита та вказав область пошуку точок | 1) Зробити запит POST /api/places. | усі місця відправлені у JSON масиві | Пройдено |
| 4 | Користувач авторизований у системі або додав API ключ до заголовку запита, вказав область пошуку точок і фільтр | 1) Зробити запит POST /api/places. | Усі місця відправлені у JSON масиві | Пройдено |
|  | Користувач авторизований у системі або додав ключ до заголовку запиту та отримав список місць | 1)Зробити запит GET https://community-open-weather-map.p.rapidapi.com/weather  2) Зробити запит POST /trip/create | Подорож створена | Пройдено |

Продовження таблиці 6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порядок дій | Очікуваний результат | Фактичний результат |
| 6 | Користувач залогінений або додав ключ до заголовку запиту | 1)Зробити запит GET https://community-open-weather-map.p.rapidapi.com/weather | Отримано JSON-об’єкт з прогнозом погоди | Пройдено |
| 7 | Користувач залогінений або додав ключ до заголовку запиту. Користувач залогінився в Google-акаунт | 1)Підтвердити створення нового нагадування в Google-календарі | Створено нове нагадування | Пройдено |
| 8 | Адміністратор залогінився або додав ключ до заголовку запиту | 1)Зробити запит POST /places/  2)Зробити запит DELETE /places/id | Видалено запис місця з таблиці | Пройдено |

Продовження таблиці 6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порядок дій | Очікуваний результат | Фактичний результат |
| 9 | Адміністратор залогінився або додав ключ до заголовку запиту | 1)Зробити запит POST /users/active/ | Отримано масив останніх активних користувачів | Пройдено |

## **6.2 Модульне тестування**

Виконаємо модульне тестування функцій системи «Створення нової подорожі». Проведемо тестування методом чорного ящику. Це дозволить перевірити на коректність окремі модулі вихідного коду програми з відповідними керуючими даними.

Таблиця 6.2 – Тестування створення нової подорожі

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип | Опис параметрів | Очікуваний результат | Фактичний результат |
| 1 | Невірний | Користувач не вводить назву подорожі | Помилка про невір.дані | Помилка про невір.дані |
| 2 | Невірний | Користувач вводить минулу дату | Помилка про невір. дату | Помилка про невір. Дату |
| 3 | Невірний | Користувач вводить невірний час | Помилка про невір. час. | Помилка про невір. Час |

## **6.3 Експеримент**

Було проведено експеримент, щоб визначити чи було досягнуто мети в дипломній роботі, а саме: скорочення витрат часу на планування подорожі.

В ньому взяли участь 20 студентів ОНПУ. Їм була поставлена задача запланувати маршрут прогулянки в незнайомому місті спочатку без веб-застосування для планування подорожі, а потім використовуючи його. В таблиці 6.3 наведено результати дослідження.

Таблиця 6.3 – Результати дослідження

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ім’я Користувача | Час (без TripPlanner), хв | Час (TripPlanner), хв |
| 1 | Користувач 1 | 120 | 110 |
| 2 | Користувач 2 | 50 | 20 |
| 3 | Користувач 3 | 78 | 72 |
| 4 | Користувач 4 | 130 | 110 |
| 5 | Користувач 5 | 102 | 94 |
| 6 | Користувач 6 | 29 | 19 |
| 7 | Користувач 7 | 43 | 38 |
| 8 | Користувач 8 | 57 | 47 |
| 9 | Користувач 9 | 100 | 92 |
| 10 | Користувач 10 | 120 | 89 |
| 11 | Користувач 11 | 143 | 129 |
| 12 | Користувач 12 | 234 | 210 |
| 13 | Користувач 13 | 142 | 115 |
| 14 | Користувач 14 | 163 | 128 |
| 15 | Користувач 15 | 89 | 49 |
| 16 | Користувач 16 | 99 | 85 |

Продовження таблиці 6.3

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | Користувач 17 | 232 | 190 |
| 18 | Користувач 18 | 78 | 39 |
| 19 | Користувач 19 | 34 | 25 |
| 20 | Користувач 20 | 76 | 69 |

Двоє студентів із двадцяти взагалі знайшли не достовірну інформацію про точку подорожі, що, безумовно, негативно вплине на якість майбутньої поїздки.

На рис. 6.1 зображено стовпчасту порівняння результатів дослідження.

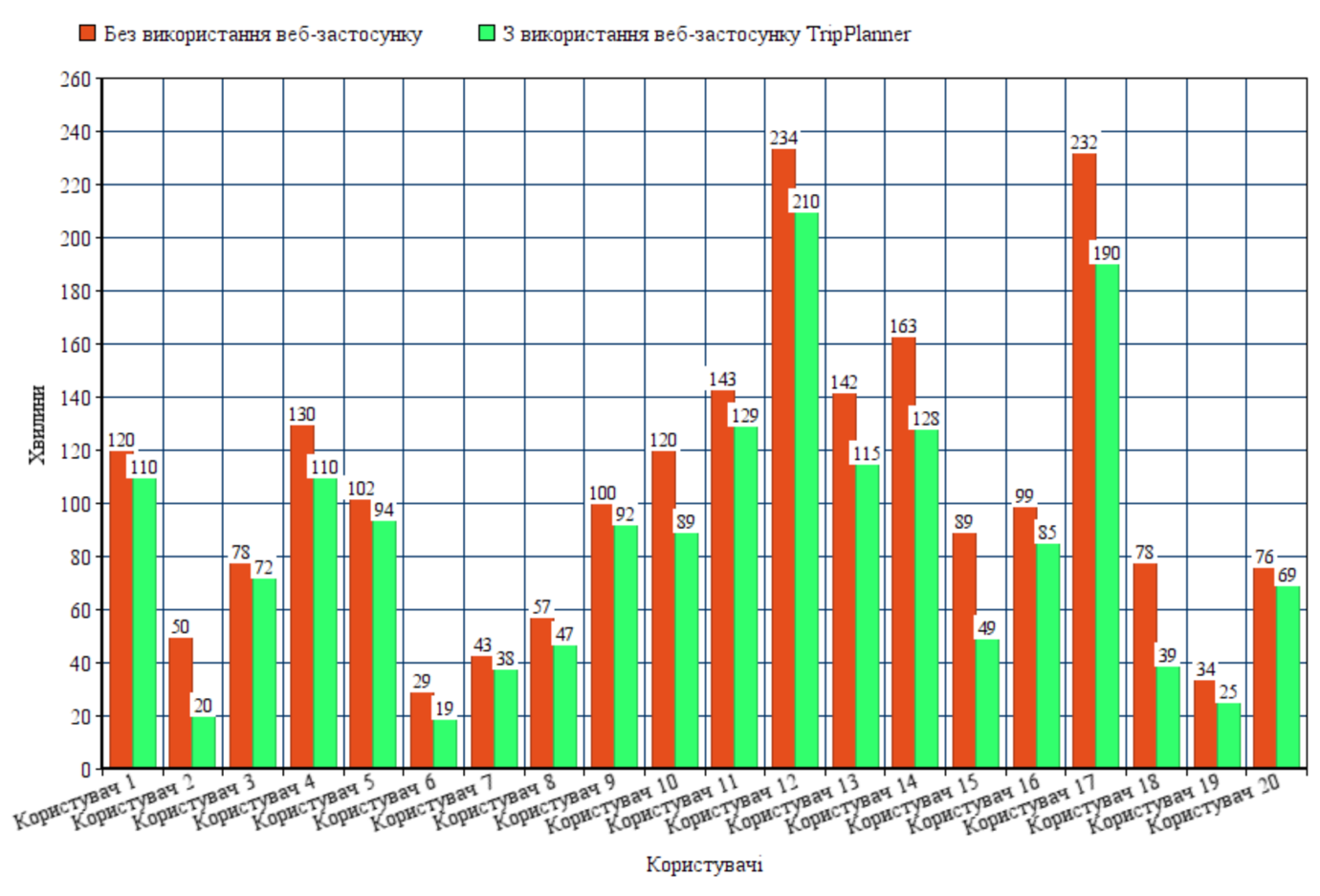


Рисунок 6.1 – Діаграма порівняння витраченого часу на планування подорожі

Середній час, витрачений на планування подорожі без веб-застосунку, становив 106 хвилин на одну людину.

Середній час, витрачений на планування подорожі з веб-застосунком, становив 87 хвилини.

Отже, середній час планування подорожі, завдяки веб-застосування скоротився на 18%

# **7 ОХОРОНА ПРАЦІ**

Охорона праці[21] - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Безпека — це такі умови, в яких перебуває складна система, коли дія зовнішніх факторiв і внутрішніх чинників не призводить до процесів, що вважаються негативними по відношенню до даної складної системи у відповідності до наявних, на даному етапі, потреб, знань та уявлень.

Оскільки дипломний проект, присвячений̆ « Веб-застосунок для планування заходів під час подорожі на базі методів побудови навігаційних маршрутів », розробку якого здійснює інженер фірми з інформаційних послуг. Таким чином для розгляду питань охорони праці оберемо робоче місце інженера з веб-розробки, яке знаходиться у офісному приміщенні.

Аналіз способів зробити працю безпечніше та менш шкідливою наведено в Додатку Б, він розкриває наступні питання:

* Визначення основних параметрів умов на робочому місці інженера-програміста
* Розрахунок освітлення на робочому місці
* Визначення категорії пожежонебезпечних приміщень
* Заходи з охорони праці
* Безпека у суспільстві в умовах загрози COVID – 19.

**Висновок**

У даному розділі було розглянуто управління та організацію охорони праці на робочому місці інженера-програміста, визначено основні параметри умов праці згідно індивідуального завдання.

Проведено розрахунок індивідуального завдання на тему - освітлення та сформовані рекомендації щодо нього. Згідно СНІП II – 4-79 природне і штучне освітлення (зміни БСТ №8,10-1986). Розряд зорової роботи інженера програміста – lV. Найменший розмір об’єкту 0.5-1мм. Найменший рівень освітлення на робочому місці програміста становить 300-500 люкс.

Проведено розрахунок індивідуального завдання на тему - визначення категорії пожежонебезпечних приміщень. Отримане питомий пожежне навантаження знаходиться в діапазоні 1 ≤ 62,8 ≤ 180 тому пожежонебезпечна приміщення можна віднести до категорії В4.

Надано рекомендації щодо безпеки у суспільстві в умовах загрози COVID – 19.

# **ВИСНОВКИ**

У даній роботі було спроектовано та розроблено веб-застосування для планування заходів під час подорожі та досягнута мета у скороченні часу, який витрачався на розробку маршруту. В результаті дослідження встановлено, що розробка веб-застосування значно полегшить процес планування подорожі. На практиці доведено, що час планування подорожі скорочується на 18%.

Враховуючи функціональні вимоги, була побудована діаграма варіантів використання і описані сценарії̈ до них, а також визначені нефункціональні вимоги. Виконано планування проекту, у ході якого була визначена тривалість повної̈ розробки, розроблено план виконання робіт, ідентифіковано ризики та розглянуті реакції̈ на найбільш імовірні ризики. Створено архітектурний̆ проект системи, на якому зображені всі підсистеми та зв’язки між ними. Під час розробки застосування було обрано та обґрунтовано технології̈ розробки, які дозволяють реалізувати архітектуру засобу та створити інтуїтивно зрозумілий інтефейс користувача. р

В представленій̆ роботі розглянуто методи побудови навігаційних маршрутів.

Проведено розрахунок освітлення та сформовані рекомендації щодо нього. Найменший рівень освітлення на робочому місці програміста становить 300-500 люкс.

Визначено категорії пожежонебезпечних приміщень. Отримане питомий пожежне навантаження знаходиться в діапазоні 1 ≤ 62,8 ≤ 180 тому пожежонебезпечна приміщення можна віднести до категорії В4.

Надано рекомендації щодо безпеки у суспільстві в умовах загрози COVID – 19.

# **СПИСОК ЛІТЕРАТУРИ**

1. Оберт Т.Г. ТЕОРИЯ КОНКУРЕНЦИИ учебное пособие для студентов экономических специальностей. 2008. 15 с.
2. Матанцев А.Н. Анализ рынка. Настольная книга маркетолога. 2017. С. 67 — 120
3. PostgreSQL : The world's most advanced open source database [Електронний курс]. — Режим доступу: https://www.postgresql.org/. — Загол. з екрану
4. Галактионов В.А., Пестун М.В. Алгоритмы построения и распознавания навигационных описаний маршрутов для картографических компьютерных систем. 2012. C. 62-80
5. [David Tuffley](https://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Dr+David+Tuffley&text=Dr+David+Tuffley&sort=relevancerank&search-alias=books) Software Requirements Specifications: A How To Guide for Project Staff. 2010. — 89 c.
6. Грицюк Ю.І. Аналіз вимог до програмного забезпечення. 2018. — 456 с.
7. Google Maps Api [Електронний ресурс]. – Режим доступу https://developers.google.com/maps/documentation . – Загол. з екрану.
8. Google Places Api [Електронний ресурс]. – Режим доступу https://loud.google.com/maps-platform/places/ . – Загол. з екрану.
9. Буч, Г. Язык UML. Руководство пользователя . 2006. – 496 с.
10. Web Speech API [Електронний ресурс]. – Режим доступу <https://wicg.github.io/speech-api/>
11. Wallace Clark, Henry Gantt. [The Gantt chart, a working tool of management](https://archive.org/details/ganttchartworkin00claruoft). 1922. — 230 c.
12. James Rumbaugh, Ivar Jacobson, Grady Booch. The unified modeling language reference manual. 1999. — 584 c.
13. Современный учебник JavaScript [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/. – Загол. з екрану.
14. Kyle Simpson You Don't Know JS. 2015. — 88 c.
15. Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/uk/. – Загол. з екрану.
16. React a JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://reactjs.org/. – Загол. з екрану.
17. Express – фреймворк веб-приложений Node.js [Електронний ресурс]. – Режим доступу: https://expressjs.com/ru/. – Загол. з экрану.
18. Passport.js [Електронний ресурс]. – Режим доступу: https://passportjs.org/. – Загол. з екрану.
19. Cem Kaner, Hung Q. Nguyen, Jack Falk Testing Computer Software 2012. — 512 c.
20. Пригожев А.С. Анализ данных при тестировании программного обеспечения // Т.А. Таран Интеллектуальный анализ информации. 2016. С. 174 – 178
21. Гандзюк М.П. Основи охорони праці: Підручник / М.П. Гандзюк, Є.П. Желібо, М. О. Халімовський. - Львів: Новий світ-2000, 2003. - 22 с