МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
В.о. завідувача кафедри
інтелектуальних інформаційних
систем, канд. техн. наук, доц.
_____Є. В. Сіденко
«____»_____2023 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА МОНІТОРИНГУ ПЕРЕМІЩЕНЬ ОСІБ, НАЛЕЖНИХ ДО СОЦІАЛЬНО НЕЗАХИЩЕНИХ ГРУП, В УМОВАХ ВОЄННОГО СТАНУ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.1710101

Виконав студент 6-го курсу, групи 601
_____**А. *Ю. Агарков***
«___» лютого 2023 р.

Керівник: канд. фіз. мат. наук, доцент
_____***І. В. Кулаковська***
«___» лютого 2023 р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**

Галузь знань **12 «Інформаційні технології»**

(шифр і назва)

Спеціальність **122 «Комп'ютерні науки»**

(шифр і назва)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доц.

_____ Є. В. Сіденко

« » 20 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Агаркову Артему Юрійович

(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи «Система моніторингу переміщень осіб, належних до соціально незахищених груп, в умовах воєнного стану».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз. мат. наук, доц.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 199

2. Строк подання студентом роботи 16 лютого 2023 р.

3. Вхідні (початкові) дані до роботи: gps-моніторинг, мова програмування Java, ос Android.

Очікуваний результат: система моніторингу переміщень осіб з можливістю прокласти найкоротший шлях до найближчого пункту допомоги.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

- аналіз актуальності розробки додатку;
- огляд та аналіз існуючих аналогів;
- огляд технологій та програмних засобів;
- розробка системи моніторингу.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: опис основних питань охорони праці пов'язаних з професійною діяльністю та використання комп'ютерів для роботи в офісі

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	Григор'єва Л.І. докт. біол. наук, професор	
Методична частина	Кулаковська І.В. канд. фіз. мат. наук, доц.	

Керівник роботи __канд. фіз. мат. наук, доц. Кулаковська І. В._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання _____Агарков А. Ю._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «07» _листопада_ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Система моніторингу переміщень осіб, належних до соціально незахищених груп в умовах воєнного стану.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	Виконано
2	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	Виконано
3	Складання календарного плану на період виконання МКР	11.11.2022	15.11.2022	Виконано
4	Огляд літератури за темою дослідження	16.11.2022	27.11.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.11.2022	18.12.2022	Виконано
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	19.12.2022	12.01.2023	Виконано
7	Опис фахової частини МКР, зокрема дослідження публікацій щодо переміщень осіб належних до соціально незахищених груп, огляд існуючих аналогів розроблюваного додатку для вирішення поставленої задачі, реалізація обраних технологій з аналізом отриманих результатів	13.01.2023	25.01.2023	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	Виконано
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	Виконано
10	Корегування роботи за результатами попереднього захисту	04.02.2023	06.02.2023	Виконано
11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2023	09.02.2023	Виконано
12	Подання МКР рецензенту	09.02.2023	10.02.2023	Виконано
13	Рецензування МКР	11.02.2023	12.02.2023	Виконано
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	Виконано
15	Захист МКР перед екзаменаційною комісією (ЕК)	22.02.2023	23.02.2023	Виконано

Розробив студент _____Агарков А. Ю._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи __канд. фіз. мат. наук, доц Кулаковська І. В.____ _____
(*наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«12» листопада 2022 р.

АНОТАЦІЯ

до магістерської кваліфікаційної роботи
студента групи 601 ЧНУ ім. Петра Могили

Агаркова Артема Юрійовича

на тему: **“СИСТЕМА МОНІТОРИНГУ ПЕРЕМІЩЕНЬ ОСІБ, НАЛЕЖНИХ ДО СОЦІАЛЬНО НЕЗАХИЩЕНИХ ГРУП, В УМОВАХ ВОЄННОГО СТАНУ”**

Актуальність даного дослідження полягає у необхідності створення додатку що допоможе відстежувати переміщення осіб належних до соціально незахищених груп в умовах воєнного часу. Це дозволить зробити переміщення людей, що належать до соціально незахищених груп більше безпечним та контрольованим.

Об’єктом дослідження є процеси позиціонування за допомогою GPS трекінгу.

Предметом дослідження є методи та засоби GPS трекінгу.

Метою дослідження є вдосконалення методів та засобів комп’ютеризованого моніторингу переміщень осіб, що належать до соціально незахищених груп та поліпшення системи контролю над ними з використанням засобів та методів GPS - трекінгу.

В результаті роботи отримано повноцінну функціонуючу систему GPS-трекінгу, яка здатна контролювати переміщення членів домашньої групи зі збереженням їх маршруту та створювати найкоротші маршрути між «точками безпеки» на карті за рахунок зберігання даних точок в базі даних та реалізації алгоритму Прима.

Дана робота складається з чотирьох розділів. Кожен розділ відповідно присвячений: аналізу предметної області, проектуванню інформаційної системи, проектуванню системи, аналізу отриманих результатів, охороні праці, методичній частині магістерської роботи. Загальний обсяг роботи – 103 сторінок. Магістерська кваліфікаційна робота містить один додаток, 12 рисунків, 5 таблицю і посилання на 30 літературних джерел.

Ключові слова: веб-розробка. Android, веб-додаток, GPS-трекінг.

ABSTRACT

to the master's qualification work by the student of the group 601 of Petro Mohyla Black Sea National University

Aharkov Artem

“DEVELOPMENT OF A SYSTEM FOR MONITORING MOVMENTS OF PERSONS BELONGING TO SOCIALLY UNPROTECTED GROUPS UNDER THE CONDITIONS OF THE STATE OF MARTIAL”

The relevance of this study is the need to create an application that will help track the movement of persons belonging to socially vulnerable groups in wartime conditions. This will make the movement of people belonging to socially vulnerable groups safer and more controllable.

The object of the work is the process of researching the process of positioning with the help of GPS tracking.

A subject of research is the methods and means of GPS tracking.

A purpose of the study is improve methods and means of computerized monitoring of movements of persons belonging to socially vulnerable groups and to improve the system of control over them using means and methods of GPS tracking.

As a result of the work, a fully functioning GPS-tracking system was obtained, which is able to control the movement of members of the home group while saving their route and create the shortest routes between "security points" on the map due to the storage of point data in the database and the implementation of Prima algorithm.

This work consists of four sections. Each section is respectively devoted to: analysis of the subject area, design of information systems, system design, analysis of the obtained results, labor protection, methodical part of the master's work. The total amount of work - 103 pages Master's thesis contains one appendix, 12 drawings, 5 table and references to 30 literary sources.

Key words: web development. Android, web app, GPS tracking.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ — Програмне забезпечення

GPS — Global Positioning System

WEB — World Wide Web

HTTPS — HyperText Transfer Protocol Secure

HTTP — HyperText Transfer Protocol

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«СИСТЕМА МОНІТОРИНГУ ПЕРЕМІЩЕНЬ ОСІБ, НАЛЕЖНИХ ДО СОЦІАЛЬНО НЕЗАХИЩЕНИХ ГРУП, В УМОВАХ ВОЄННОГО СТАНУ»

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.1710101

Виконав студент 6-го курсу, групи 601

_____**А. *Ю. Агарков***

«___» лютого 2023 р.

Керівник: канд. фіз. мат. наук, доцент

_____***І. В. Кулаковська***

«___» лютого 2023 р.

Миколаїв – 2023

ВСТУП

В сучасному світі діджиталізація і науково-технічний прогрес розвиваються швидше за будь-які інші галузі. Кожного дня з’являються нові програмні продукти і додатки.

Одним з популярних на сьогоднішній день напрямків є поняття GPS-трекінгу. Дане поняття розуміє під собою виявлення фізичного місця положення девайсу з наступною обробкою.

Дані технології широко використовуються в найрізноманітніших галузях, починаючи від контролю місцезнаходження дитини до трекінгу транспорту, наприклад, на базі служби таксі.

Виходячи з усього вищесказаного, можна зробити висновок про високу актуальність явища GPS-трекінгу.

Об’єктом є процес дослідження процесу позіціонування з допомогою GPS трекінгу.

Предметом дослідження є методи та засоби GPS трекінгу.

Метою дослідження є реалізація системи моніторингу переміщень осіб, що належать до соціально незахищених груп та поліпшення системи контролю над ними з використанням засобів та методів GPS - трекінгу.

Для досягнення поставленої мети слід виконати наступні завдання:

- провести огляд предметної області:
    a) проаналізувати поняття прикладного програмного забезпечення;
    b) провести аналіз поняття GPS-трекінгу;
    c) провести огляд існуючих рішень;
- провести проектування системи:
    a) обрати архітектуру системи;
    b) спроектувати базу даних системи;
    c) спроектувати внутрішню будову системи;
- провести розробку програмного забезпечення:

a) провести огляд інструментальних засобів розробки;
b) провести огляд використаних алгоритмів і їх реалізації;
c) розробити графічний інтерфейс користувача;

- провести тестування і огляд результатів:
  a) провести тестування системи;
  b) провести огляд отриманих результатів через порівняння з виявленими аналогами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Прикладне програмне забезпечення

Прикладне програмне забезпечення – це тип комп’ютерної програми, яка виконує певні функції. Ці функції, які виконує прикладне програмне забезпечення, можуть бути особистими, діловими та освітніми. Таким чином, прикладне програмне забезпечення також відоме як програмне забезпечення для кінцевого користувача або програмне забезпечення для продуктивності. Кожна програмна програма розроблена, щоб допомогти користувачам у певному процесі, пов’язаному з продуктивністю, ефективністю та спілкуванням. На відміну від системного програмного забезпечення, прикладне програмне забезпечення є специфічним для своєї функціональності та виконує завдання, для виконання якого воно розроблене. Більшість програм, які ми бачимо на наших смартфонах, є прикладами програмного забезпечення.

Прикладне програмне забезпечення Програми розроблено для виконання різноманітних ролей. Функції не обмежуються, але залежать від потреб користувача. Деякі з найпоширеніших функцій прикладного програмного забезпечення:

- маніпулювання даними;
- управління інформацією;
- обчислення фігур;
- побудова візуалів;
- координаційні ресурси;
- написання звітів;
- створення електронних таблиць;
- ведення документації;
- розробка сайтів;

– розрахунок витрат.

Тип прикладного програмного забезпечення, яке вам потрібно використовувати, залежить від ваших потреб. Ось список програмного забезпечення або, можна сказати, типи програмного забезпечення.

Програмне забезпечення для обробки текстів використовується для форматування, прикрашання та обробки тексту. Він допускає такі функції, як синоніми та антоніми. Ви можете змінювати шрифти, кольори та стиль відповідно до свого вибору за допомогою функції Word Art. У ньому також доступні параметри перевірки помилок, граматики та орфографії. Microsoft Word є найкращим прикладом програмного забезпечення для обробки текстів.

Програмне забезпечення електронних таблиць в основному використовується для зберігання даних у форматі таблиці та виконання обчислень. Пересічні комірки даються в електронній таблиці для зберігання різних полів даних, таких як час, дата, текст і числа. Користувачі можуть виконувати обчислення за допомогою формул і функцій. Найкращим прикладом програмного забезпечення для роботи з електронними таблицями є Microsoft Excel.

Програмне забезпечення для презентацій дозволяє вам викладати свої думки та ідеї у вигляді візуальної інформації. Потім ви можете представити цю інформацію у вигляді слайдів. Ви можете зробити свої слайди інтерактивними та інформативними, додавши відео, тексти, діаграми, графіки та зображення. Найкращим прикладом програмного забезпечення для презентацій є Microsoft PowerPoint.

Мультимедійне програмне забезпечення дозволяє створювати або записувати відео, аудіо та файли зображень. Таке програмне забезпечення використовується для редагування відео, графіки та анімації. Типовими прикладами мультимедійного програмного забезпечення є програвач VLC, MX Player і Windows Media Player.

Веб-браузери и використовуються для перегляду в Інтернеті. Вони дозволяють знаходити та отримувати дані з Інтернету. Найпопулярнішими веб-браузерами є Chrome і Firefox.

Ці типи **прикладного програмного забезпечення** називаються академічним програмним забезпеченням, оскільки вони спеціально розроблені для полегшення навчання. У нього включено всі види навчального програмного забезпечення. Прикладами освітнього програмного забезпечення є EDX, MindPlay і Kid Pix.

Графічне програмне забезпечення використовується для внесення змін у візуальні дані, зображення й анімацію. Він містить різне редакційне програмне забезпечення. Прикладами графічного програмного забезпечення є Adobe Photoshop, Unity 3d і PaintShop.

Безкоштовне програмне забезпечення, це тип програмного забезпечення доступний безкоштовно. Тому ви можете завантажити та встановити їх безкоштовно. Однак вам заборонено вносити будь-які зміни у вихідний код. Skype є прикладом безкоштовного програмного забезпечення.

Умовно-безкоштовні програми розповсюджуються серед користувачів на пробній основі. Потім, якщо користувачам це подобається і вони хочуть продовжити, вони повинні заплатити за це програмне забезпечення. Прикладом умовно-безкоштовного програмного забезпечення є Winzip.

Програмне забезпечення моделювання – це програма моніторингу, яка дозволяє користувачеві спостерігати за операцією, не виконуючи її. Таке програмне забезпечення корисне, коли робота існуючої системи не дуже точна, передбачувана або небезпечна. Він широко використовується в техніці, робототехніці, системах польоту, прогнозі погоди, тестуванні, освіті та відеоіграх. MATLAB є найкращим прикладом програмного забезпечення для моделювання.

Програмне забезпечення з відкритим **кодом** доступне з вихідним кодом і правами для будь-кого перевіряти, змінювати та покращувати його. Крім того, більшість програм з відкритим вихідним кодом доступна безкоштовно, тоді як лише небагато є платними на такому умовному рівні.

Програмне забезпечення із закритим вихідним кодом є прямою протилежністю програмного забезпечення з відкритим кодом. Вони є платним програмним забезпеченням і мають права інтелектуальної власності або терпіння над вихідним кодом. Зазвичай він супроводжується деякими обмеженнями, а також умовами.

Бізнес-прикладне програмне забезпечення — це підмножина прикладного програмного забезпечення, яке організація використовує переважно для бізнес-цілей. Це програмне забезпечення розроблено спеціально для полегшення певних бізнес-функцій. Серед суттєвих переваг прикладного програмного забезпечення для бізнесу є підвищена продуктивність, ефективність, точність і періодичні звіти для бізнес-аналізу. Сьогодні кожна нова та швидкозростаюча бізнес-організація використовує програмне забезпечення для бізнесу.

Програмне забезпечення для управління взаємовідносинами з клієнтами керує взаємодією компанії з поточними та потенційними клієнтами. Крім того, це допомагає збирати, аналізувати та розробляти стратегію великої кількості даних про клієнтів для розвитку бізнесу.

Планування ресурсів підприємства — це програмне забезпечення та система, яка керує всіма основними видами діяльності та іншими бізнес-процесами організації. ERP може автоматизувати та спростити такі бізнес-діяльності, як закупівлі та бухгалтерський облік, управління ризиками, управління проектами, комплаєнс та керування ланцюгом поставок.

Програмне забезпечення для керування проектами – це прикладне програмне забезпечення, яке використовується для планування проектів, керування змінами, планування та розподілу ресурсів. Він також допомагає користувачам виконувати такі функції, як керування бюджетом і витратами, документування прогресу, звітування про результати та призначення завдань.

Програмне забезпечення бази даних, яке також називають системою керування базами даних (СУБД), призначене для створення бази даних і зберігання, пошуку, обробки та вилучення важливих даних організації.

Програмне забезпечення для керування бізнес-процесами, також відоме як програмне забезпечення BPM, є інструментом автоматизації процесів. Він визначає, звітує та автоматизує процеси, призначені для оптимізації бізнесу організації.

Програмне забезпечення для керування ресурсами – це тип прикладного програмного забезпечення, яке допомагає планувати ресурси, капітал і персонал для безперешкодного завершення проекту. Це також допомагає керувати кількома проектами та гарантує, що все розподіляється в режимі реального часу.

Навчальне програмне забезпечення відноситься до прикладного програмного забезпечення, розробленого з освітньою метою. Освітнє програмне забезпечення полегшує вивчення та викладання освітнього змісту, концепцій і процесів.

Програмне забезпечення продуктивності допомагає користувачам виконувати свої завдання більш ефективно та вчасно. Його категорії включають співпрацю, управління часом, управління базами даних і створення документів. Це програми, які організації використовують для підвищення загальної продуктивності.

Спеціально розроблене програмне забезпечення – це прикладне програмне забезпечення, створене виключно для організації чи окремого користувача відповідно до їхніх бізнес-потреб. Це одне з найпопулярніших і широко використовуваних прикладних програм для бізнесу. Він може виконувати завдання, яке ви бажаєте, і бути спроектованим і розробленим відповідно до процесу вашої організації. У результаті ви можете отримати власне програмне забезпечення для автоматизації, інтеграції та керування всіма своїми завданнями.

1.2 GPS-трекінг

Пристрій відстеження GPS, пристрій географічного відстеження або просто трекер — це навігаційний пристрій, зазвичай на транспортному засобі, об’єкті, людині чи тварині, який використовує систему глобального позиціонування (GPS) для визначення свого руху та визначення свого географічного розташування WGS84 UTM (географічне відстеження), щоб визначити його місцезнаходження [1]. Пристрої відстеження GPS посилають спеціальні супутникові сигнали, які обробляються приймачами.

Місцезнаходження зберігається в пристрої відстеження або передається на пристрої, підключені до Інтернету, через стільникову мережу (GSM/GPRS/CDMA/LTE або SMS), вбудований радіопристрій або супутниковий модем або Wi-Fi у всьому світі.

Розмір GPS-антени обмежує розмір трекера, який зазвичай менше півдолара (30,61 мм у діаметрі). До 2020 року відстеження обійдеться в 2 мільярди доларів плюс військові витрати. Під час війни в Перській затоці 10% або більше цілей використовували трекери. Майже кожен мобільний телефон відстежує кожен його рух. Стежка може бути нанесена на карту в режимі реального часу за допомогою програмного забезпечення для відстеження GPS і пристроїв із підтримкою GPS [2].

GPS «Track Me» по суті містить модуль GPS, який приймає сигнали GPS і обчислює координати. Для реєстраторів даних він містить велику пам'ять для зберігання координат. Передавач даних також містить модем GSM/GPRS/CDMA/LTE для передачі цієї інформації в IP-пакетах через SMS або GPRS на центральний комп’ютер. Пристрої супутникового відстеження GPS можна використовувати в будь-якій точці світу за допомогою супутникових технологій, таких як GlobalStar або Iridium. Вони не вимагають стільникового зв'язку.

Існує три типи GPS-трекерів, хоча більшість телефонів із GPS можуть працювати в будь-якому з цих режимів залежно від встановленого мобільного додатка:

GPS-реєстратор періодично записує місцезнаходження пристрою у внутрішню пам'ять. Логери GPS можуть мати слот для карти пам’яті або внутрішню флеш-карту та USB-порт. Деякі діють як флеш-накопичувачі USB, дозволяючи завантажувати дані журналу трасування для подальшого аналізу на вашому комп’ютері. Списки треків або POI можуть бути у форматах GPX, KML, NMEA або інших форматах.

Більшість цифрових камер економлять час під час фотографування. Якщо годинник камери достатньо точний або використовує GPS як джерело часу, цей час можна співвіднести з даними журналу GPS, щоб забезпечити точне місцезнаходження. Це можна додати до метаданих Exif у файлі зображення. Камери з вбудованими GPS-приймачами можуть безпосередньо створювати такі фотографії з геотегами. У деяких випадках приватного розслідування реєстратори даних використовуються для відстеження цільових транспортних засобів. Приватним детективам не потрібно надто ретельно стежити за цілями, і вони завжди мають резервне джерело даних.

Пускачі даних є найпоширенішим типом пристрою відстеження GPS, який використовується для відстеження активів, особистого відстеження та відстеження системи. стеження за транспортними засобами. Практично кожен телефон перебуває в цьому режимі згідно з угодою користувача, навіть якщо зберігання даних для майбутніх передач вимкнено або вимкнено.

Цей тип пристрою, також відомий як «GPS-маяк», періодично надсилає місцезнаходження пристрою разом із іншою інформацією, такою як швидкість або висота, на призначений сервер, де дані можна зберігати та негайно аналізувати.

Пристрій GPS-навігації та телефон знаходяться поруч в одній коробці та живляться від однієї батареї. Телефон періодично надсилає текстові повідомлення через SMS або GPRS, що містять дані з GPS-приймача. Новіші смартфони з

вбудованим GPS і програмним забезпеченням GPS-відстеження можуть перетворити ваш телефон на пристрій передачі даних (або реєстратор). Починаючи з 2009 року, програми з відкритим кодом і пропрієтарні програми доступні для телефонів з функціями Java ME, iPhone, Android, Windows Mobile і Symbian [3, 4, 5].

Більшість GPS-трекерів 21-го століття пропонують технологію передачі даних, яка дозволяє здійснювати складне GPS-відстеження в комерційних середовищах, особливо в організаціях з мобільною робочою силою, як-от комерційні автопарки. Типова система відстеження GPS, яка використовується в комерційному управлінні автопарком, складається з двох основних частин: апаратного забезпечення позиціонування (або пристрою відстеження) та програмного забезпечення відстеження. Цю комбінацію часто називають автоматичною системою позиціонування автомобіля. Пристрої стеження найчастіше встановлюються в автомобілях, підключаються до шини CAN, замку запалювання та акумулятора. Це дозволяє збирати додаткові дані, які згодом передаються на сервер відстеження GPS. У більшості випадків це можна переглянути на веб-сайті, доступному через Інтернет, а діяльність автопарку можна переглянути в реальному часі або за попередні періоди за допомогою цифрових карт і звітів.

Системи відстеження GPS, які використовуються в комерційних автопарках, зазвичай налаштовані на передачу вхідних даних про місцезнаходження та телеметрію з певною частотою оновлення або коли подія (відкриті/зачинені двері, увімкнення/вимкнення допоміжного пристрою, перетин кордону геозони) запускає пристрій передачі даних. GPS-відстеження в режимі реального часу, яке використовується в комерційних автопарках, зазвичай відноситься до систем, які періодично оновлюються з інтервалом в одну, дві або п’ять хвилин, коли запалювання ввімкнено. Деякі системи відстеження поєднують заплановані оновлення з оновленнями, викликаними зміною назви. Великі комерційні автострахові компанії використовують рішення GPS-відстеження, такі як

Telematics 2.0, телематичну технологію на основі Інтернету речей для автомобільної промисловості.

GPS-пристрої для збору даних також називають «GPS-транспондерами». Ці пристрої завжди ввімкнено, і їх можна опитувати скільки завгодно (технологія Pull), на відміну від пристроїв даних, які періодично надсилають інформацію про своє місцезнаходження (технологія Push). Ця технологія не має широкого поширення, але прикладом такого пристрою є комп'ютер, підключений до Інтернету, на якому працює gpsd.

Коли лише час від часу необхідно знати місцезнаходження трекера (наприклад, розміщення на майні, яке може бути вкрадене, Або пристрої, які не мають постійного джерела живлення, яке періодично надсилає дані, наприклад вантажі чи контейнери.)

Екстрактори даних стають все більш поширеними у формі пристроїв, що містять GPS-приймачі та мобільні телефони, які відповідають повідомленнями із зазначенням свого місцезнаходження, коли надсилаються спеціальні SMS-повідомлення.

Приховані GPS-трекери містять ту саму електроніку, що й звичайні GPS-трекери, але розроблені так, щоб вони виглядали як повсякденні речі. Приховані GPS-трекери можна використовувати для захисту електроінструментів; ці пристрої можна сховати в ящиках для електроінструментів і відстежувати в разі крадіжки.

У серпні 2010 року бразильська компанія Unilever провела незвичайну акцію, яка включила GPS-трекери в контейнери для прального порошку Omo. Потім команди відстежують покупців, які купили коробку миючого засобу, до свого дому, де вони отримають покупку. Компанія також запустила веб-сайт (португальською мовою), на якому показано приблизне розташування будинків переможців [30].

1.3 Аналіз існуючих рішень

Одним з найвідоміших аналогів є Skyriver GPS, який представляє собою GPS-додаток на смартфон, який дозволяє відстежувати пристрій в реальному часі.

Інтерфейс додатку зображено на рисунках 1.1., 1.2.

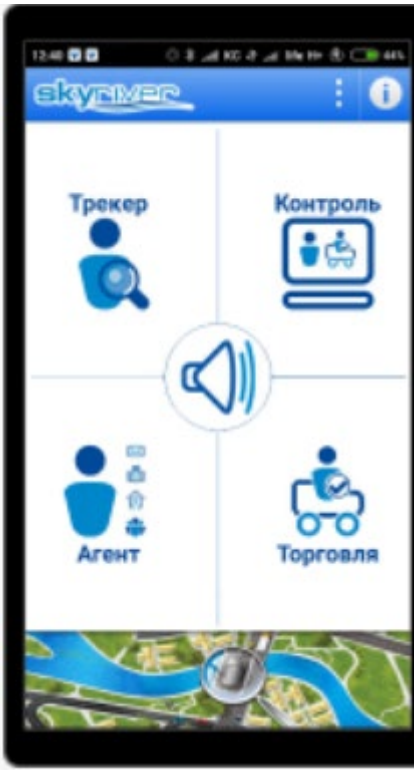

Рисунок 1.1 — Інтерфейс Skyriver

Рисунок 1.2 — Інтерфейс Skyriver

Застосунок Find My Kids - це сімейний GPS-трекер для дітей та їх батьків, створений з єдиною метою - зберегти життя та здоров'я вашої дитини. Безпека визначається тим, як додаток збирає та кому передає ваші дані. Способи забезпечення конфіденційності й захисту даних можуть різнитися залежно від використання додатка, регіону та віку користувача.

Інтерфейс додатку зображено на рисунку 1.3.

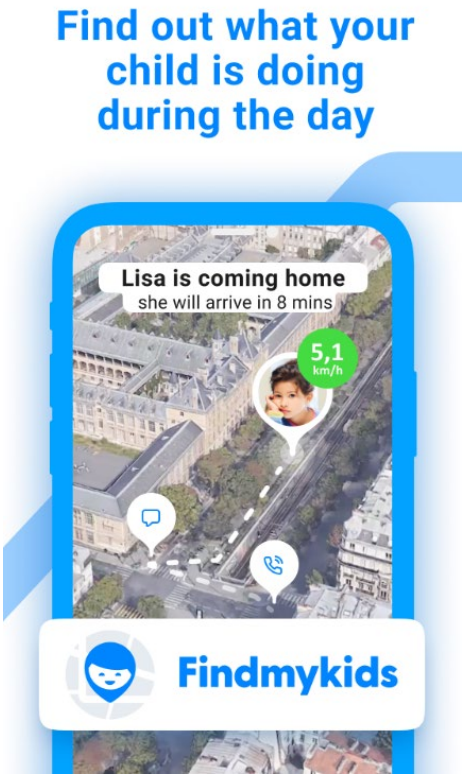

Рисунок 1.3 — Інтерфейс Find My Kids

Family Locator дозволяє залишатися на зв'язку з членами вашої родини протягом дня. Відстеження місцезнаходження сім'ї використовує вбудований GPS-трекер вашого телефону, щоб забезпечити безпеку сім'ї, навіть якщо вони далеко. Користувач може отримувати сповіщення, коли відстежені члени родини досягають пунктів призначення. Обмін геоданими GPS з членами родини на сімейній карті.

Інтерфейс додатку зображено на рисунку 1.4.

Рисунок 1.4 — Інтерфейс Family Locator

Сімейний GPS трекер KidsControl Circles створений для безпеки дітей та сім'ї. Користувач зможе ділитися своїм місцезнаходженням та отримувати автоматичні повідомлення про пересування дитини, знатимете, коли він вийшов зі школи і приїхав додому, чи не розрядилася в нього батарея і чи довго ще добиратися до місця.

Інтерфейс додатку зображено на рисунку 1.5.

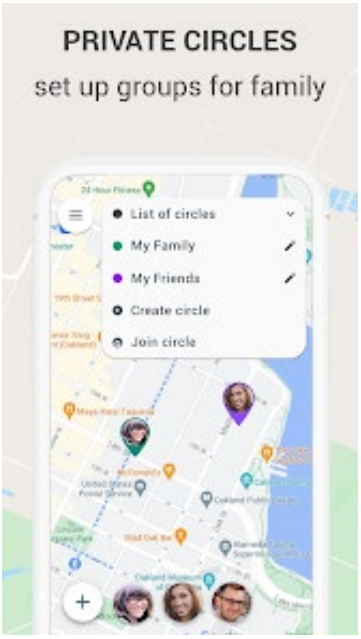

Рисунок 1.5 — Інтерфейс KidsControl Circles

1.4 Постановка задачі

Актуальність даного дослідження полягає у необхідності створення додатку що допоможе відстежувати переміщення осіб належних до соціально незахищених груп в умовах воєнного часу. Це дозволить зробити переміщення людей, що належать до соціально незахищених груп більше безпечним та контрольованим.

Об’єктом дослідження є поняття GPS трекінгу.

Предметом дослідження є методи та засоби GPS трекінгу.

Метою дослідження є реалізація системи моніторингу переміщень осіб, що належать до соціально незахищених груп та поліпшення системи контролю над ними з використанням засобів та методів GPS - трекінгу.

Для досягнення поставленої мети має бути розроблена власна програмна реалізація системи GPS-трекінгу, яка повинна мати наступний функціонал:

- клієнтський додаток, для передачі даних;
- серверна частина для обробки даних;
- наявність механізмів реєстрації користувачів;
- наявність механізмів авторизації користувачів;
- можливість перегляду і внесення змін до інформації щодо активних девайсів;
- можливість перегляду останнього онлайну девайсу на карті світу.

Висновки до розділу 1

Після написання даного розділу, в якому здійснено аналіз за основними поняттями порушеного питання, а саме створення теоретичної бази стосовно GPS трекінгу, подібних додатків, методів і засобів їх розробки, дала змогу оцінити ситуацію та сформувати базову картину та план, щодо створення власної реалізації GPS трекеру. Таким чином, перший розділ магістерської роботи дав змогу зрозуміти та ознайомитись з наявною теоретичною інформацією стосовно предметної області створення GPS трекерів.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір архітектури

Model-View-Controller (MVC) — це шаблон проектування програмного забезпечення [1], який зазвичай використовується для розробки інтерфейсів користувача, який розділяє відповідну логіку програмного забезпечення на три взаємопов’язані елементи. Це робиться для того, щоб відокремити внутрішнє представлення інформації від того, як інформація подається та отримується користувачем [2, 3].

Традиційно використовується для настільних графічних інтерфейсів користувача (GUI), цей шаблон став популярним у розробці веб-додатків [4]. Популярні мови програмування мають фреймворки MVC, які полегшують реалізацію шаблонів.

Як одна з фундаментальних ідей у ранній розробці графічних інтерфейсів користувача, MVC став одним із перших способів опису та впровадження програмних структур відповідно до їхніх обов’язків [5].

Тригве Ренскауг створив MVC, працюючи над Smalltalk-79 як запрошений вчений у Xerox Palo Alto Research Center (PARC) наприкінці 1970-х років [6, 7, 8]. Він хотів створити схему, яку можна було б використовувати для побудови. Будь-яка програма, де користувачі взаємодіють із великими та складними наборами даних. Спочатку він складався з чотирьох частин: модель, вигляд, предмет і редактор. Після обговорень з іншими розробниками Smalltalk він та решта команди зупинилися на моделі, представленні та контролері [6].

У своєму остаточному дизайні моделі чітко та інтуїтивно зрозуміло представляють частини програми. Представлення — це візуальне представлення моделі, яке отримує дані від моделі для відображення користувачеві та передає запити між користувачем і моделлю. Контролер — це організуюча частина інтерфейсу користувача, яка розміщує та координує кілька подання на екрані,

отримує введені користувачем дані та надсилає відповідні повідомлення до своїх базових поданнях. Ця конструкція також включає редактор як особливий тип контролера для зміни певного представлення, створеного з цим видом [6].

Smalltalk-80 підтримує версію MVC, яка розвинулася з нього [6]. Він надає абстрактні класи View і Controller, а також різні конкретні підкласи кожного абстрактного класу, що представляють різні загальні віджети. У цьому сценарії подання представляє певний спосіб відображення інформації для користувача, а контролер представляє конкретний спосіб взаємодії користувача з поданням. Представлення також пов'язане з об'єктом моделі, але структура цього об'єкта залежить від програміста. Smalltalk-80 також включає MVC Inspector, інструмент розробки для перегляду даної моделі, представлення та структури контролера поруч [9].

У 1988 році в Journal of Object Technology (JOT) двоє колишніх співробітників PARC представили MVC як загальну «парадигму та метод програмування» для розробників Smalltalk-80. Однак їхня схема відрізняється від схеми Reenskaug та інших і схеми, представленої в посібнику Smalltalk-80. Вони визначають вигляд як такий, що містить будь-які графічні проблеми, тоді як контролер є більш абстрактним, зазвичай невидимим об'єктом, Він отримує дані користувача та взаємодіє з одним або декількома представленнями даних і має лише одну модель [10].

Згодом шаблон MVC розвинувся [11], що призвело до таких моделей, як ієрархічна модель-перегляд-контролер (HMVC), модель-перегляд-адаптер (MVA), модель-перегляд-рендерер (MVP), модель-перегляд-перегляд тощо Варіанти. модель (MVVM).) та інші, які застосовують MVC до різних контекстів.

Використання шаблону MVC у веб-додатках збільшилося після представлення NeXT WebObjects у 1996 році, спочатку написаного на Objective-C (значною мірою запозиченого з Smalltalk), і допомагає запровадити принципи MVC. Шаблон MVC пізніше став популярним серед розробників Java, коли

WebObjects було перенесено на Java. Подальші фреймворки Java, такі як Spring (випущений у жовтні 2002 року), продовжили міцний зв’язок між Java та MVC.

У 2003 році Мартін Фаулер опублікував «Патерни архітектури корпоративних додатків», у яких представлено MVC як шаблон, у якому «контролери введення» отримують запити, надсилають відповідні повідомлення об’єктам моделі, отримують відповіді від об’єктів моделі та передають відповіді [8]: 56 Це близький до підходу, який використовується фреймворком Ruby on Rails (серпень 2004), де клієнт надсилає запит на сервер через перегляд браузера, який потім його обробляє. Потім контролер зв’язується з відповідним об’єктом моделі [12]. Фреймворк Django (липень 2005, для Python) пропонує подібний шаблон "MTV" (Model Template View), де представлення отримує дані від моделі та передає їх шаблону для відображення [13]. Дебют як Rails, так і Django з сильним акцентом на швидкому розгортанні підвищив популярність MVC за межами його давньої популярності в традиційних корпоративних середовищах.

Елементи шаблону

Модель. Основний компонент шаблона. Це динамічна структура даних програми, незалежна від інтерфейсу користувача [14]. Вона безпосередньо керує даними програми, логікою та правилами.

Вид. Будь-яке представлення інформації, наприклад діаграма, графік або таблиця. Можливі декілька переглядів тієї самої інформації, наприклад гістограма для керівництва та таблиця для бухгалтерів.

Команди, які приймають вхідні дані та перетворюють їх на модель або подання [15].

Окрім поділу програми на ці компоненти, конструкція модель-представлення-контролер визначає взаємодію між ними [16].

- модель відповідає за управління даними програми. Він отримує дані користувача від контролера;
- подання відображає подання моделі в певному форматі;

- контролер реагує на введення користувача та здійснює взаємодії з об’єктами моделі даних. Контролер отримує вхідні дані, за бажанням перевіряє їх і потім передає вхідні дані моделі.

Як і інші шаблони програмного забезпечення, MVC виражає «основне рішення» проблеми, що робить його застосовним до кожної системи [17]. Конкретний дизайн MVC може суттєво відрізнятися від традиційних описів тут [18].

Хоча MVC спочатку був розроблений для робочого столу, він був широко прийнятий основними мовами програмування для розробки додатків World Wide Web. Для реалізації цього шаблону було створено кілька веб-фреймворків. Інтерпретації цих програмних інфраструктур відрізняються, головним чином у тому, як обов’язки MVC розподіляються між клієнтом і сервером [20].

Деякі веб-платформи MVC використовують підхід тонкого клієнта, розміщуючи майже всю логіку моделі, представлення та контролера на сервері. У цьому підході клієнт надсилає гіперпосилання або запит форми до контролера, який потім отримує повну та оновлену веб-сторінку (або інший документ) із представлення; модель повністю живе на сервері [20].

2.2 Аналіз варіантів використання

Для більш детального розуміння функціонування системи необхідно проаналізувати варіанти використання системи.

Оскільки клієнтська частина представляє собою просто сервіс з однією кнопкою, то діаграма варіантів використання представлена виключно для серверної частини.

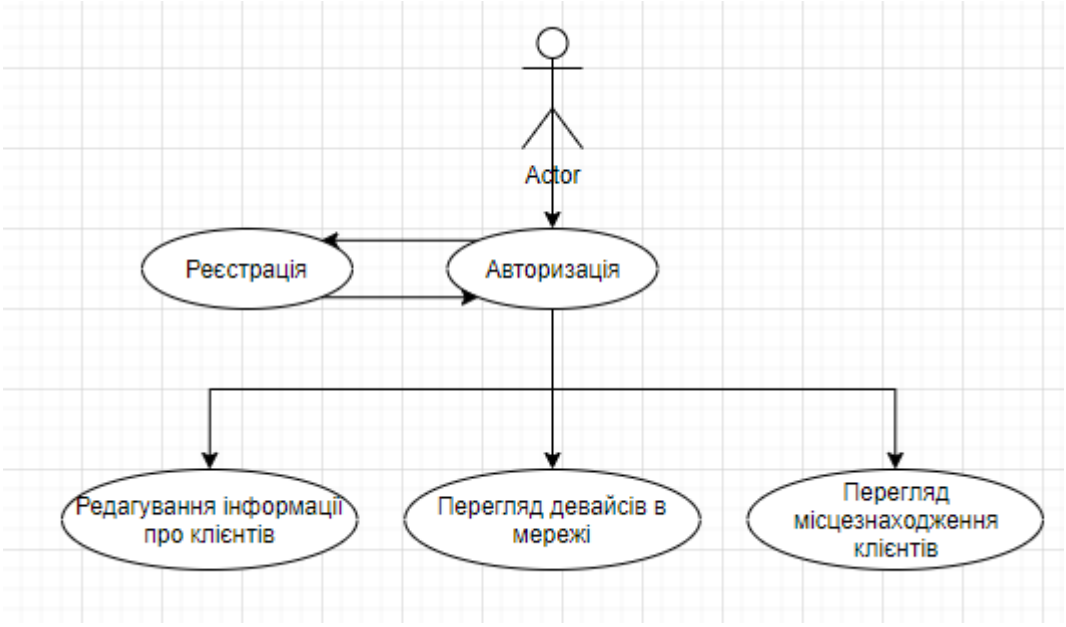

Рисунок 2.1 — Діаграма варіантів використання

2.3 Проектування бази даних

База даних складається з трьох сутностей, які описують основні дані системи. Сутність персони зберігає в собі унікальний ідентифікатор, повне ім’я, логін і пароль. Сутність дозволів ілюструє дозволи одній персоні на моніторинг переміщень іншої, зберігаючи унікальний ідентифікатор дозволу, ідентифікатор наглядача і суб’єкта моніторингу. Сутність мітки, в свою чергу, зберігає час, дату, широту і довготу, ідентифікатор персони, до якої відноситься даний маркер.

Схема бази даних зображена на рисунку 2.2.

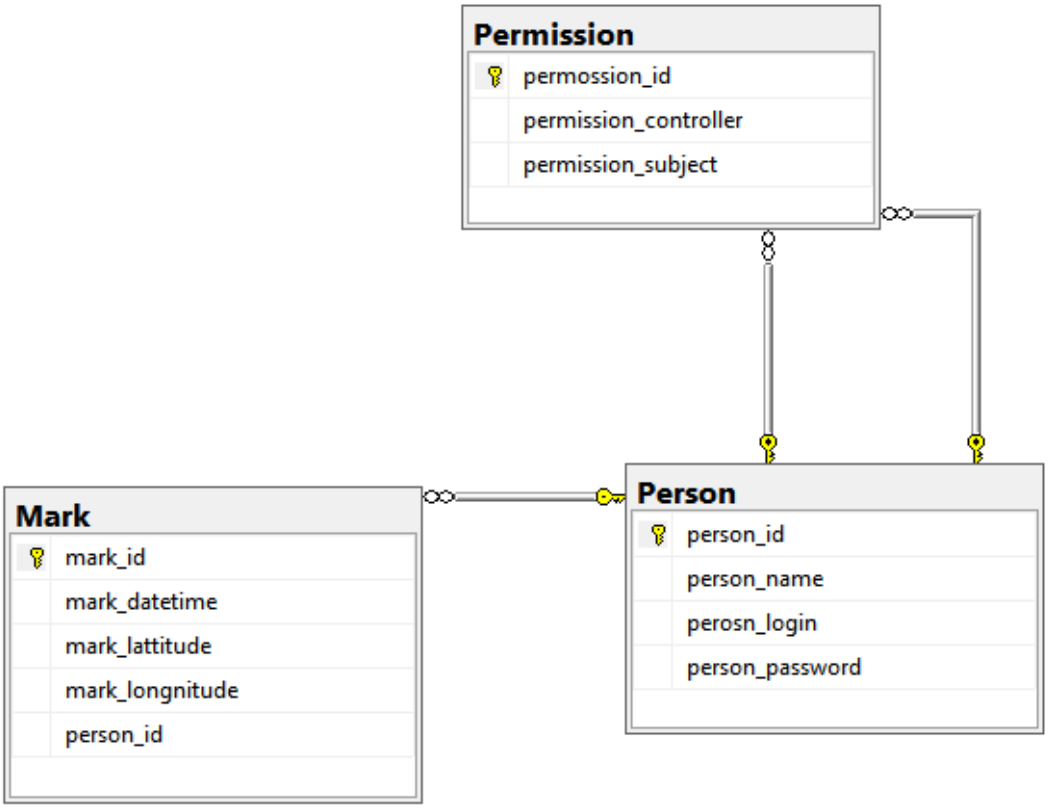

Рисунок 2.2 — Схема бази даних

2.4 Проектування внутрішньої будови

Внутрішня будова розуміє під собою склад класів, їх методів і полів, оскільки клас є основною будівельною одиницею програми.

Діаграма класів зображена на рисунку 2.3.

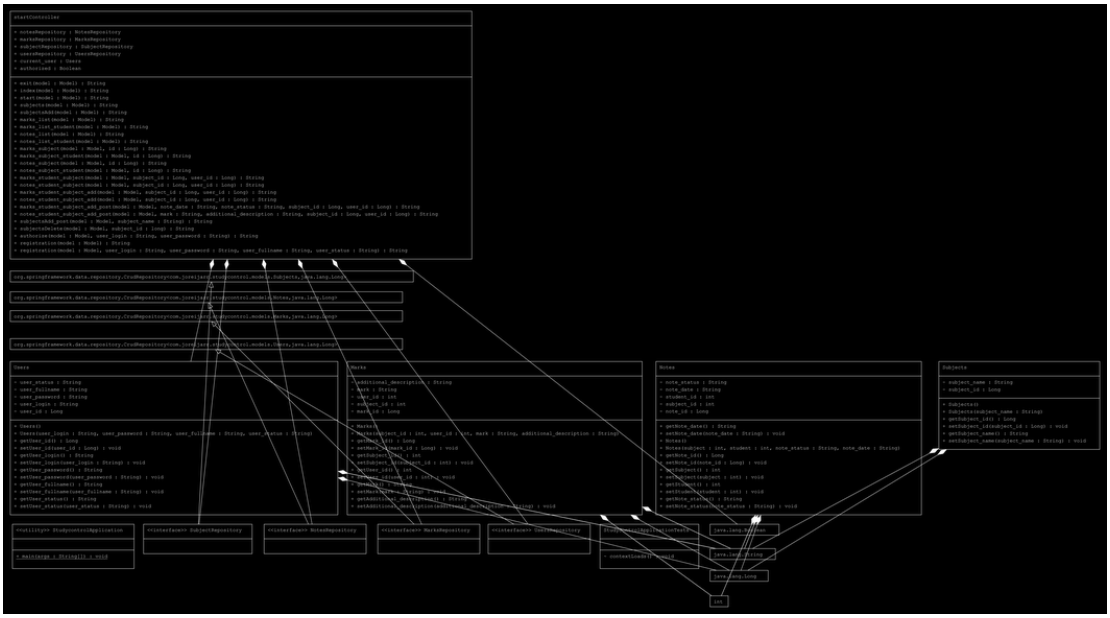

Рисунок 2.3 — Діаграма класів

Висновки до розділу 2

Мета даного розділу полягала у виборі архітектури для реалізації проекту, та проектуванні бази даних. В ході написання другого розділу було обрано архітектуру, проаналізовані варіанти використання та сформовано базу даних. Таким чином, другий розділ магістерської роботи дав змогу виробити підхід до розробки майбутньої системи і засоби, якими вона буде реалізована.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Огляд інструментальних засобів розробки

Мова програмування Java була розроблена Sun Microsystems на початку 1990-х років. Незважаючи на те, що Java в основному використовується для Інтернет-програм, вона є простою, ефективною мовою загального призначення. Java спочатку була розроблена для вбудованих мережевих програм, що працюють на кількох платформах. Це портативна, об'єктно-орієнтована, інтерпретована мова.

Java надзвичайно портативна. Той самий Java-додаток працюватиме однаково на будь-якому комп'ютері, незалежно від апаратного забезпечення чи операційної системи, якщо він має інтерпретатор Java. Окрім портативності, ще однією з ключових переваг Java є її набір функцій безпеки, які захищають ПК, на якому запущено програму Java, не лише від проблем, викликаних помилковим кодом, але й від шкідливих програм (таких як віруси). Ви можете безпечно запускати аплет Java, завантажений з Інтернету, оскільки функції безпеки Java запобігають доступу цих типів аплетів до жорсткого диска комп'ютера або мережевих з'єднань. Аплет – це, як правило, невелика програма Java, вбудована в сторінку HTML.

Java можна вважати як скомпільованою, так і інтерпретованою мовою, оскільки її вихідний код спочатку компілюється в двійковий байт-код. Цей байт-код працює на віртуальній машині Java (JVM), яка зазвичай є програмним інтерпретатором. Використання скомпільованого байт-коду дозволяє інтерпретатору (віртуальній машині) бути малим і ефективним (і майже таким же швидким, як центральний процесор, що виконує власний скомпільований код). Крім того, цей байт-код надає Java її переносимість: вона працюватиме на будь-якій JVM, яка правильно реалізована, незалежно від комп'ютерної апаратної чи програмної конфігурації. Більшість веб-браузерів (таких як Microsoft Internet Explorer або Netscape Communicator) містять JVM для запуску Java-аплетів.

Порівняно з C++ (іншою об’єктно-орієнтованою мовою), код Java працює трохи повільніше (через JVM), але він більш портативний і має набагато кращі функції безпеки. Віртуальна машина забезпечує ізоляцію між ненадійною програмою Java і ПК, на якому запущено програмне забезпечення. Синтаксис Java подібний до C++, але мови досить різні. Наприклад, Java не дозволяє програмістам реалізувати перевантаження операторів, тоді як C++ дозволяє. Крім того, Java є динамічною мовою, де ви можете безпечно змінювати програму під час її роботи, тоді як C++ це не дозволяє. Це особливо важливо для мережевих програм, які не можуть дозволити собі будь-які простої. Крім того, усі базові типи даних Java є попередньо визначеними та не залежать від платформи, тоді як деякі типи даних можуть змінюватися залежно від платформи, що використовується в C або C++ (наприклад, тип int).

Програми Java структуровані краще, ніж еквіваленти C++. Усі функції (або методи Java) і виконувані оператори в Java повинні знаходитися в межах класу, тоді як C++ дозволяє визначенням функцій і рядкам коду існувати поза класами (як у програмах у стилі C). Глобальні дані та методи не можуть перебувати поза класом у Java, тоді як C++ це дозволяє. Ці обмеження, хоч часом і обтяжливі, допомагають підтримувати цілісність і безпеку програм Java і змушують їх бути повністю об’єктно-орієнтованими.

Іншою ключовою особливістю Java є те, що це відкритий стандарт із загальнодоступним вихідним кодом. Sun Microsystems контролює мову Java та пов’язані з нею продукти, але ліберальна ліцензійна політика Sun сприяла тому, що Інтернет-спільнота прийняла Java як стандарт. Ви можете безкоштовно завантажити всі інструменти, необхідні для розробки та запуску Java-аплетів і додатків, із веб-сайту Sun Java (http://java.sun.com).

Мова розмітки гіпертексту або HTML є стандартною мовою розмітки для документів, що відображаються у веб-браузері. У цьому можуть допомогти такі технології, як каскадні таблиці стилів (CSS) і мови сценаріїв, такі як JavaScript.

Веб-браузер отримує документи HTML з веб-сервера або локального сховища та перетворює документи на мультимедійні веб-сторінки. HTML семантично описує структуру веб-сторінки і спочатку містить підказки про те, як повинен виглядати документ.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою структури HTML зображення та інші об'єкти (наприклад, інтерактивні форми) можуть бути вбудовані в скопійовані сторінки. HTML надає інструменти для створення структурованих документів, які представляють структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, посилання та інші елементи. Елементи HTML розділені тегами, записаними в кутових дужках. Такі теги, як <img /> і <input /> вносять вміст безпосередньо на сторінку. Інші теги, такі як <p>, оточують та надають інформацію про текст документа, а також можуть включати інші теги як дочірні елементи. Замість відображення тегів HTML браузери використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати скрипти, такі як JavaScript, які впливають на поведінку та вміст веб-сторінок. Увімкнення CSS визначає зовнішній вигляд і макет вмісту. Консорціум World Wide Web Consortium (W3C), колишній розпорядник HTML і поточний розпорядник стандартів CSS, заохочує використання CSS замість явного представлення HTML з 1997 року [2]. Форми HTML, відомі як HTML5, використовуються для відображення відео та аудіо, в основному за допомогою елемента <canvas> у поєднанні з javascript.

Розмітка HTML складається з кількох ключових компонентів, включаючи ті, які називаються тегами (та їх атрибутами), типи даних на основі символів, посилання на символи та посилання на сутність. Теги HTML найчастіше зустрічаються парами, наприклад <h1> і </h1>, хоча деякі теги представляють порожні елементи і тому є дивними, наприклад <img>. Першим тегом у такій парі є початковий тег, а другий — кінцевий (також його називають початковим і кінцевим тегом).

Іншим важливим компонентом є оголошення типу документа HTML, яке запускає відтворення стандартного режиму.

Нижче наведено приклад класичного «Hello, World!». програма:

```
<!DOCTYPE html>
<html>
  <header>
    <title> Це ім'я</title>
  </head>
  <текст>
    <div>
       <p>Привіт, світ! </p>
    </div>
  </text>
</html>
```

Текст між <html> і </html> описує веб-сторінку, а текст між <body> і </body> є видимим вмістом сторінки. Текст тегу <title> <title> визначає назву сторінки браузера, яка з'являється на вкладках і заголовках вікна браузера, а тег визначає поділ сторінки, що використовується для спрощення дизайну. Ви можете використовувати елемент <meta> між <head> і </head>, щоб визначити метадані сторінки.

Оголошення doctype <!DOCTYPE html> призначено для HTML5. Різні браузери повертатимуться до «чудового» режиму, якщо рекламу не ввімкнено.

Документи HTML мають на увазі структуру вкладених елементів HTML. Вони представлені в документі тегами HTML, укладеними в кутові дужки наступним чином: .[71]У простому загальному випадку область дії елемента визначається парою тегів: «відкриваючий тег» і «завершальний тег» . Текстовий вміст елемента (якщо є) розміщується між цими тегами.

Мітки також можуть містити додаткові теги міток між початком і кінцем, включаючи поєднання міток і тексту. Це вказує наступні (вкладені) елементи як дочірні елементи батьківського елемента.

Початковий тег також може включати атрибути елементів всередині тегу. Вони вказують додаткову інформацію, таку як ідентифікатор розділу в документі, ідентифікатор, який використовується для прив’язки інформації про стиль до представлення документа, а для деяких тегів, наприклад <img> для вбудовування зображень, посилання на ресурси зображень у таких документах. : <img src="example.com/example.jpg">

Деякі елементи, такі як розриви рядків або
, не дозволяють вставляти будь-який вміст, текст чи інші теги. Вони вимагають лише порожнього тега (як початкового тегу) і не використовують закриваючий тег.

Багато тегів, включаючи дуже поширений закриваючий тег елемента абзацу <p>, є необов’язковими. Браузер HTML або інший проксі-сервер може визначити кінець елемента відповідно до контекстних і структурних правил, визначених стандартом HTML. Ці правила складні, і більшість програмістів HTML не розуміють.

Тому загальна форма елемента HTML: <tag attribute1="value1" attribute2="value2"> " content " </tag>. Деякі елементи HTML визначаються як порожні елементи і мають форму <tag attribute1 = "value1" attribute2 = "value2"> . Порожні елементи можуть не містити вмісту, наприклад теги
 або вбудовані теги <img>. Назви елементів HTML – це імена, які використовуються в тегах. Зверніть увагу, що назви закриваючих тегів починаються з косої риски / , а в порожніх елементах закриваючі теги не є ані обов’язковими, ані дозволеними. Якщо атрибут не вказано, у кожному випадку використовується значення за замовчуванням.

Наприклад, <h2> Golf </h2> встановлює "Гольф" як заголовок другого рівня. Структурна розмітка не передбачає певного відтворення, але більшість веб-

браузерів мають стилі за замовчуванням для форматування елементів. Вміст можна додатково стилізувати за допомогою каскадних таблиць стилів (CSS).

Теги презентації вказують, як повинен виглядати текст незалежно від його призначення

Наприклад, жирний текст вказує на те, що пристрої візуального виводу мають відображати жирний текст, але мало вказує на те, які пристрої не повинні (наприклад, аудіопристрої, які читають текст вголос). У випадку жирного тексту і курсивного тексту, існують інші елементи, які можуть мати еквівалентні візуальні, але більш семантичні символи, наприклад сильний текст сильний >> і підкреслений текст відповідно. Легше зрозуміти, як слухові користувацькі агенти повинні інтерпретувати останні два елементи. Однак вони не еквівалентні своїм аналогам у презентації: наприклад, програми зчитування з екрана не очікують підкреслювати назви книг, але такі назви виділені на екрані курсивом. Відповідно до специфікації HTML 4.0 більшість елементів розмітки презентації застаріли на користь стилів за допомогою CSS.

Гіпертекстова розмітка перетворює частини документа на посилання на інші документи

Елемент прив’язки створює гіперпосилання в документі, а його атрибут href встановлює цільову URL-адресу посилання. Наприклад, тег HTML Вікіпедія відображатиме слово "Вікіпедія" як гіперпосилання. Щоб відобразити зображення як гіперпосилання, вставте елемент img як вміст всередині елемента a. Як і br, img є порожнім елементом з атрибутами, але без вмісту чи закриваючого тега. <img src="image.gif" alt="текст опису" width="50" height="50" border="0"> .

Більшість **атрибутів** елемента – це пари ім’я-значення, розділені знаком =, а початкова розмітка елемента записується після імені елемента. Значення можуть бути взяті в одинарні або подвійні лапки, хоча в HTML вони можуть бути не в лапках (але не в XHTML). Значення атрибутів без лапок вважаються небезпечними. На відміну від атрибутів пари ім'я-значення, є атрибути, які

впливають на елементи лише через їх присутність у початковій розмітці елемента [6], наприклад, атрибут ismap елемента img.

Існує кілька загальних атрибутів, які можуть з’являтися в багатьох елементах:

- атрибут id забезпечує унікальний ідентифікатор елемента для всього документа. Це використовується для ідентифікації елемента, щоб таблиці стилів могли змінювати його презентаційні властивості, а сценарії могли змінювати, анімувати або видаляти його вміст або презентацію. Доданий до URL-адреси сторінки, він надає глобальний унікальний ідентифікатор для елемента, як правило, підрозділу сторінки. Наприклад, ідентифікатор «Атрибути» в https://en.wikipedia.org/wiki/HTML#Attributes;
- атрибут class забезпечує спосіб класифікації подібних елементів. Це можна використовувати для семантичних або презентаційних цілей. Наприклад, документ HTML може семантично використовувати позначення <class="notation">, щоб вказати, що всі елементи з цим значенням класу підпорядковані основному тексту документа. У презентації такі елементи можуть бути зібрані разом і представлені у вигляді виносок на сторінці замість того, щоб з’являтися там, де вони зустрічаються у джерелі HTML. Атрибути класу використовуються семантично в мікроформатах. Можна вказати кілька значень класів; наприклад, <class="notation important"> поміщає елемент як до нотації, так і до важливих класів;
- автор може використовувати атрибут style для призначення властивостей презентації певному елементу. Вважається кращою практикою використовувати ідентифікатор елемента або атрибути класу для вибору елемента з таблиці стилів, хоча іноді це може бути занадто громіздким для простого, конкретного або спеціального стилю;
- атрибут title використовується для додавання підтекстового пояснення до елемента. У більшості браузерів цей атрибут відображається як підказка;

- атрибут lang визначає природну мову вмісту елемента, яка може відрізнятися від мови решти документа. Наприклад, в англомовному документі:

  Ну що ж, c'est la vie, як кажуть у Франції.

Семантичний HTML – це спосіб написання HTML, який підкреслює важливість закодованої інформації щодо її представлення (зовнішнього вигляду). HTML із самого початку включав семантичну розмітку, а також презентаційну розмітку, таку як теги <font>, <i> і . Існують також семантично нейтральні теги span і div. З кінця 1990-х років, коли каскадні таблиці стилів почали працювати в більшості браузерів, веб-авторам було рекомендовано уникати використання HTML-розмітки презентації для розділення презентації та вмісту.

Каскадні таблиці стилів, які люблять називати CSS, — це просто розроблена мова, призначена для спрощення процесу створення веб-сторінок, які виглядають презентабельно. CSS дозволяє застосовувати стилі до веб-сторінок. Що ще важливіше, CSS дозволяє робити це незалежно від HTML-коду, з якого складається кожна веб-сторінка. Він описує, як має виглядати веб-сторінка: він визначає кольори, шрифти, інтервали та багато іншого. Коротше кажучи, ви можете зробити так, щоб ваш веб-сайт виглядав як завгодно. CSS дозволяє розробникам і дизайнерам визначати його поведінку, зокрема те, як елементи розташовуються у браузері.

Тоді як html використовує теги, css використовує набори правил. CSS простий у вивченні та розумінні, але він забезпечує потужний контроль над представленням документа HTML.

CSS економить час: можна написати CSS один раз і повторно використовувати той самий аркуш на кількох HTML-сторінках.

Просте обслуговування: щоб внести глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках оновляться автоматично.

Пошукові системи: CSS вважається чистою технікою кодування, що означає, що пошуковим системам не доведеться намагатися «прочитати» його вміст.

Стилі кращі за HTML: CSS має набагато ширший набір атрибутів, ніж HTML, тому ви можете надати своїй HTML-сторінці набагато кращий вигляд порівняно з атрибутами HTML.

Офлайн-перегляд: CSS може зберігати веб-додатки локально за допомогою офлайн-кешу. За допомогою цього ми можемо переглядати офлайн-сайти.

Spring дозволяє легко створювати корпоративні програми Java. Він надає все необхідне для використання мови Java у корпоративному середовищі з підтримкою Groovy і Kotlin як альтернативних мов у JVM, а також з гнучкістю для створення багатьох типів архітектур залежно від потреб програми. Починаючи з версії Spring Framework 6.0, Spring вимагає Java 17+.

Spring підтримує широкий спектр сценаріїв застосування. У великому підприємстві програми часто існують тривалий час і мають працювати на JDK і сервері програм, цикл оновлення якого знаходиться поза контролем розробника. Інші можуть працювати як єдина банка з вбудованим сервером, можливо, у хмарному середовищі. Інші можуть бути автономними програмами (наприклад, пакетні або інтеграційні робочі навантаження), яким не потрібен сервер.

Spring є відкритим кодом. Він має велику та активну спільноту, яка надає безперервний зворотний зв’язок на основі різноманітних випадків використання в реальному світі. Це допомогло Spring успішно розвиватися протягом дуже тривалого часу.

Термін «Spring» означає різні речі в різних контекстах. Його можна використовувати для позначення самого проекту Spring Framework, з якого все почалося. Згодом інші проекти Spring були створені на основі Spring Framework. Найчастіше, коли говорять «Spring», мають на увазі всю сімейку проектів.

Spring Framework розділений на модулі. Програми можуть вибирати, які модулі їм потрібні. В основі — модулі основного контейнера, включаючи модель конфігурації та механізм впровадження залежностей. Крім того, Spring Framework забезпечує базову підтримку для різних архітектур додатків, включаючи обмін повідомленнями, транзакційні дані та постійність, а також веб. Він також включає

веб-фреймворк Spring MVC на основі Servlet і, паралельно, реактивний веб-фреймворк Spring WebFlux.

Примітка щодо модулів: фреймворк Spring дозволяє розгортати шлях модуля JDK 9 («Jigsaw»). Для використання в програмах із підтримкою Jigsaw, Spring Framework 5 jar постачається із записами маніфесту «Automatic-Module-Name», які визначають стабільні назви модулів на рівні мови («spring.core», «spring.context» тощо) незалежно від назви артефактів jar (банки мають той самий шаблон іменування з «-» замість «.», наприклад, «spring-core» та «spring-context»). Звичайно, фреймворк Spring продовжує добре працювати на шляху до класів як на JDK 8, так і на 9+.

Керівні принципи Spring Framework:

- забезпечити вибір на кожному рівні. Spring дозволяє відкласти рішення щодо дизайну якомога пізніше. Наприклад, ви можете змінити постачальників постійності через конфігурацію, не змінюючи свій код. Те саме стосується багатьох інших питань інфраструктури та інтеграції зі сторонніми API;
- врахувати різноманітні точки зору. Spring сприймає гнучкість і не має думок про те, як все має бути зроблено. Він підтримує широкий спектр потреб додатків з різними перспективами;
- підтримати сильну зворотну сумісність. Еволюція Spring була ретельно розроблена, щоб змусити кілька критичних змін між версіями. Spring підтримує ретельно підібраний діапазон версій JDK і бібліотек сторонніх розробників, щоб полегшити обслуговування програм і бібліотек, які залежать від Spring;
- подбати про дизайн API. Команда Spring вкладає багато думок і часу в створення інтуїтивно зрозумілих API, які витримають багато версій і багато років;
- встановити високі стандарти якості коду. Spring Framework робить сильний акцент на змістовному, актуальному та точному javadoc. Це один із

небагатьох проектів, які можуть претендувати на чисту структуру коду без циклічних залежностей між пакетами.

3.2 Огляд використаних алгоритмів та їх реалізації

В інформатиці алгоритм Прима (також відомий як алгоритм Ярніка) — це жадібний алгоритм, який знаходить мінімальне остовне дерево для зваженого неорієнтованого графа. Це означає, що він знаходить підмножину ребер, яка утворює дерево, що включає кожну вершину, де загальна вага всіх ребер у дереві мінімізована. Алгоритм працює, будуючи це дерево одну вершину за раз, з довільної початкової вершини, на кожному кроці додаючи найдешевший можливий зв’язок від дерева до іншої вершини.

Алгоритм був розроблений у 1930 році чеським математиком Войтехом Ярніком[1], а пізніше був перевідкритий і перевиданий комп’ютерними вченими Робертом С. Прімом у 1957 році[2] та Едсгером В. Дейкстрою у 1959 році.[3] Тому його також іноді називають алгоритмом Ярніка [4], алгоритмом Пріма–Ярніка [5], алгоритмом Пріма–Дейкстри [6] або алгоритмом DJP.[7]

Інші добре відомі алгоритми для цієї проблеми включають алгоритм Крускала та алгоритм Борувки.[8] Ці алгоритми знаходять мінімальний охоплюючий ліс у можливо відключеному графі; навпаки, найпростіша форма алгоритму Прима знаходить лише мінімальні остовні дерева в зв’язаних графах. Однак, запускаючи алгоритм Прима окремо для кожного зв’язаного компонента графа, його також можна використовувати для знаходження мінімального охоплюючого лісу.[9] З точки зору їх асимптотичної часової складності, ці три алгоритми однаково швидкі для розріджених графів, але повільніші, ніж інші більш складні алгоритми.[7, 6] Однак для досить щільних графів алгоритм Прима можна змусити працювати в лінійному часі, дотримуючись або покращуючи часові межі для інших алгоритмів.[10]

Програмна реалізація алгоритму, написаного мовою програмування Java, представлена нижче.

```
int minKey(int key[], Boolean mstSet[])
    {
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++)
            if (mstSet[v] == false && key[v] < min) {
                min = key[v];
                min_index = v;
            }

        return min_index;
    }

    void primMST(int graph[, ])
    {
        int parent[] = new int[V];

        int key[] = new int[V];

        Boolean mstSet[] = new Boolean[V];

        for (int i = 0; i < V; i++) {
            key[i] = Integer.MAX_VALUE;
            mstSet[i] = false;
        }

        key[0] = 0; // Make key 0 so that this vertex is
        // picked as first vertex
        parent[0] = -1; // First node is always root of MST
        for (int count = 0; count < V - 1; count++) {
            int u = minKey(key, mstSet);
            mstSet[u] = true;
            for (int v = 0; v < V; v++)
                if (graph[u, v] != 0 && mstSet[v] == false
                    && graph[u, v] < key[v]) {
                    parent[v] = u;
                    key[v] = graph[u, v];
                }
        }

    }
```

3.3 Розробка графічного інтерфейсу

Вся система складається з серверної частини і клієнтської частини. Клієнтська частина містить в собі один екран з однією кнопкою.

Серверна частина містить декілька функціональних сторінок. Найважливішою є сторінка показу останнього онлайн сеансу користувачів на карті. Окрім цього є сторінки додавання і редагування девайсів, сторінка підбору маршруту.

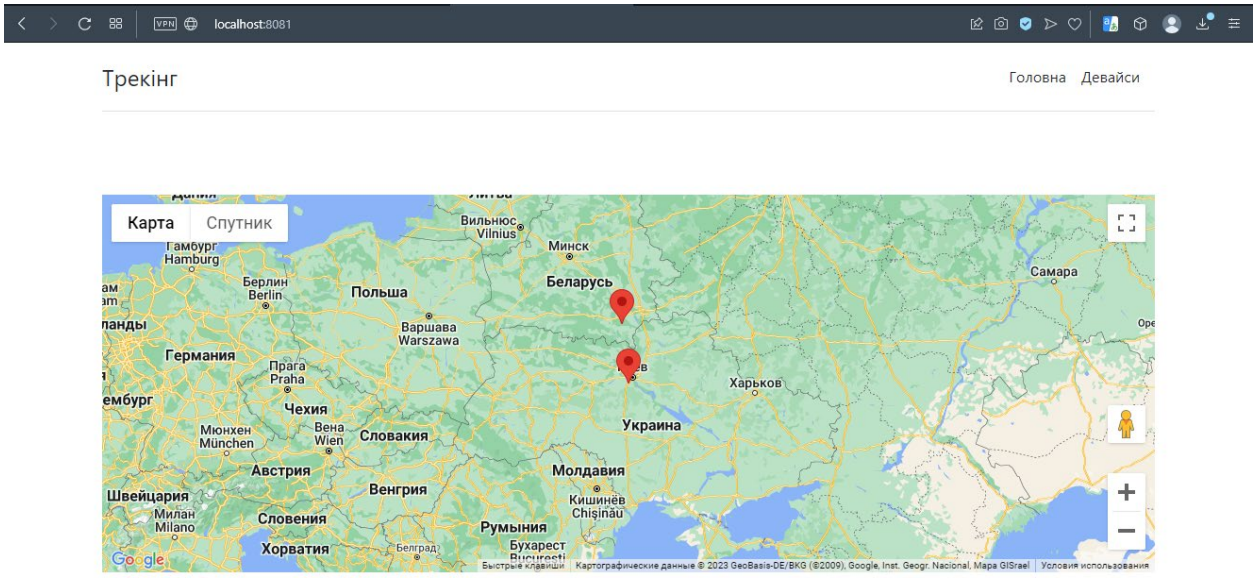

Рисунок 3.1 — Головна сторінка

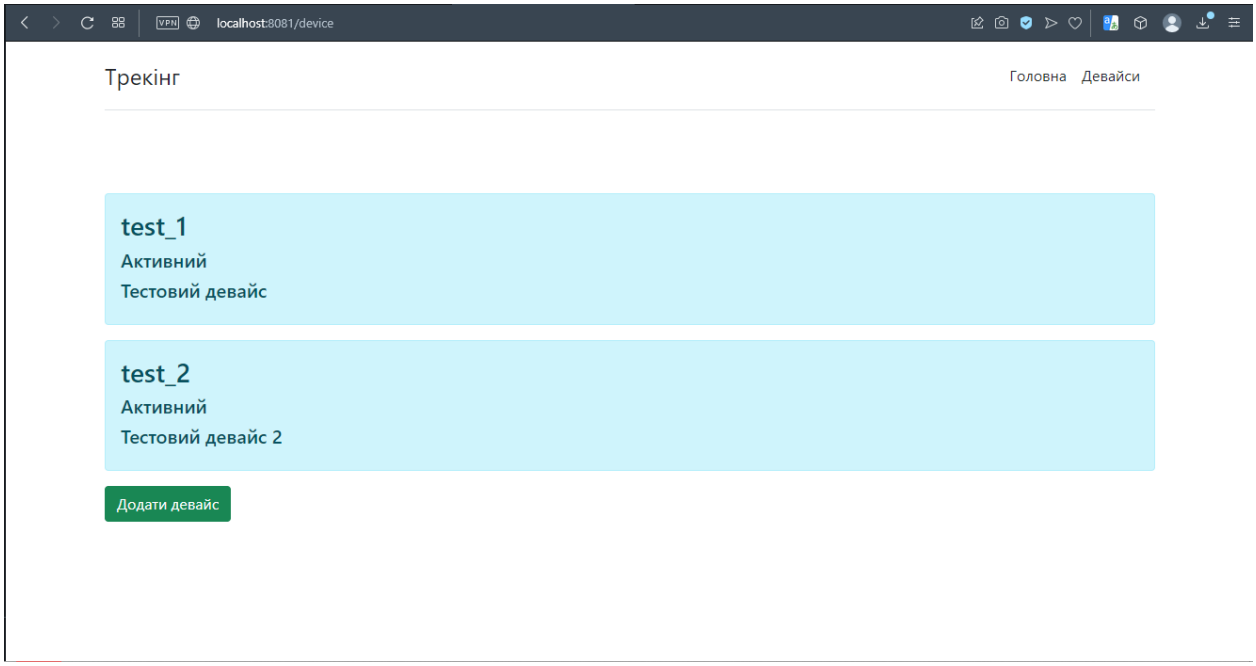

Рисунок 3.2 — Сторінка девайсів

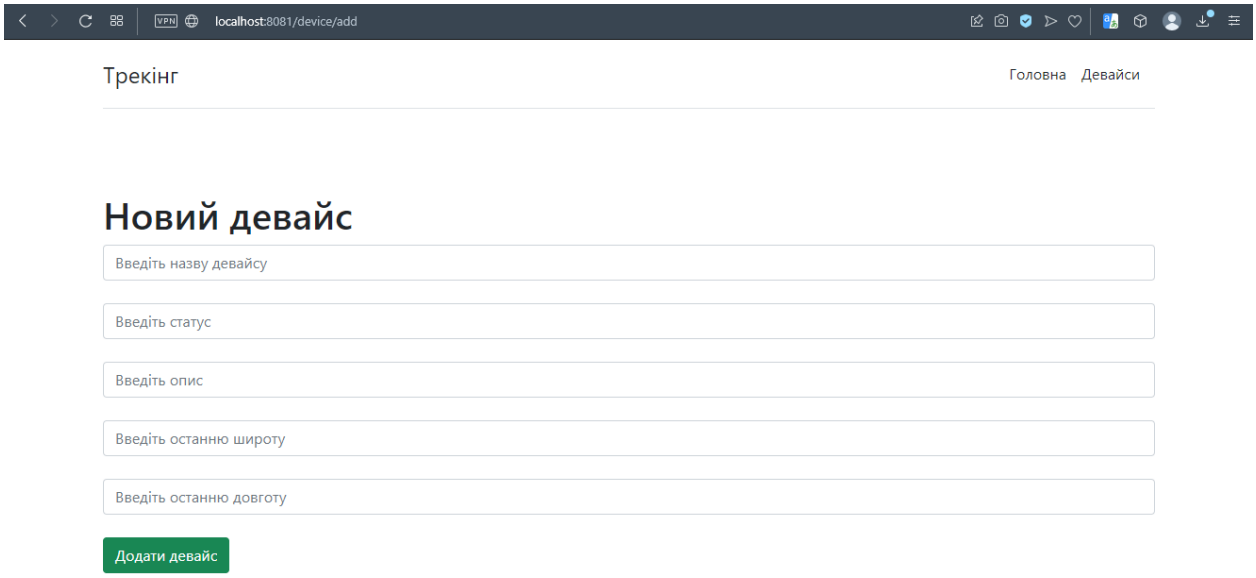

Рисунок 3.3 — Сторінка додавання девайсів

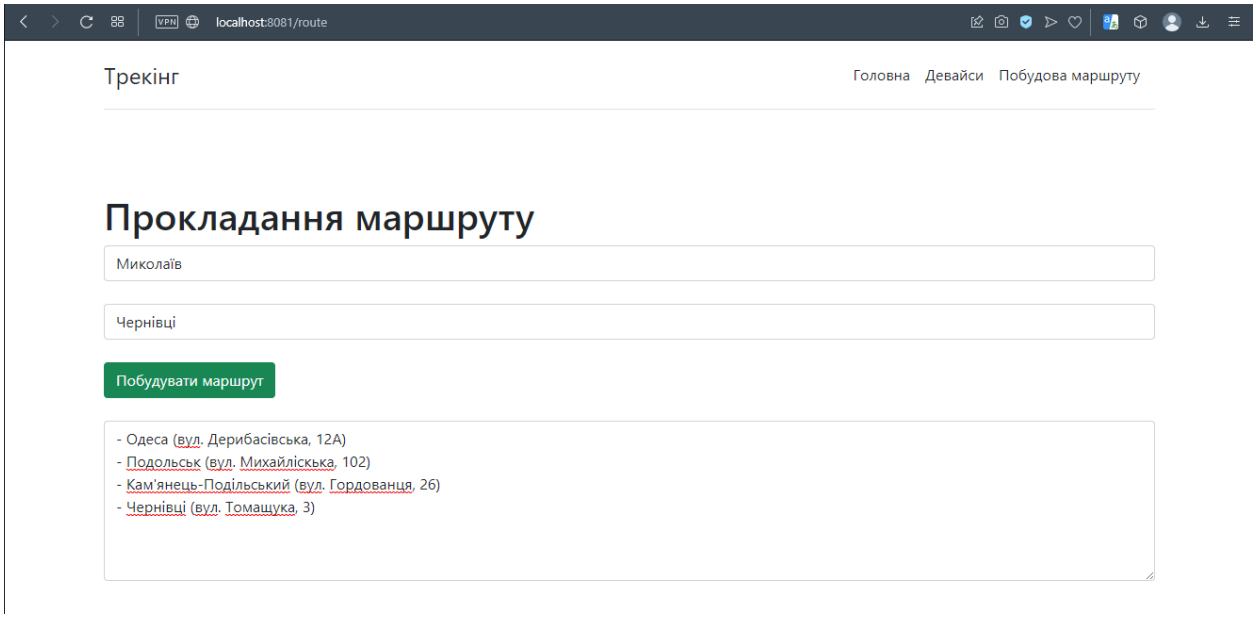

Рисунок 3.4 — Сторінка підбору маршрутів

Окрім цього система має мобільний додаток, який представлено інтерфейсом з єдиною кнопкою. Інтерфейс мобільного додатку зображено на рисунку 3.5.

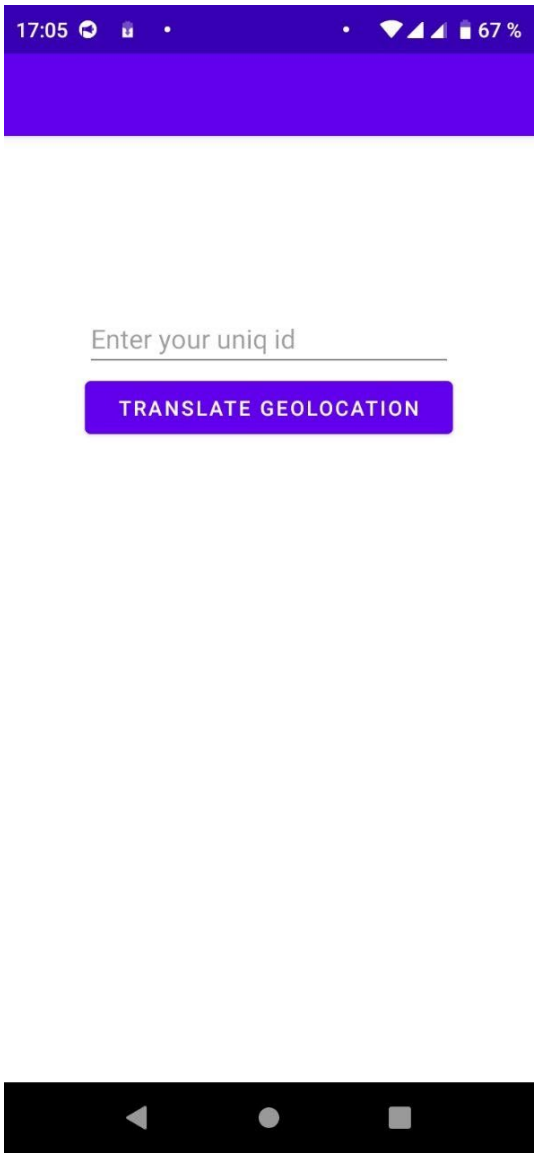

Рисунок 3.5 – Інтерфейс мобільного додатку

Висновки до розділу 3

В ході написання третього розділу проведені заходи щодо проектування та розробки власної реалізації додатку для GPS трекінгу. В ході проектування було створено UML-діаграми класів і варіантів використання, які ілюструють функціонал системи і його внутрішню будову. Окрім цього описано основні елементи кожної з діаграм для отримання загального розуміння загальної картини. Проведено огляд використаних алгоритмів та їх програмної реалізації, створено графічний інтерфейс користувача, обрано інструментальні засоби розробки системи.

4 ТЕСТУВАННЯ І АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Тестування

Тестування програмного забезпечення – це акт перевірки артефактів і поведінки програмного забезпечення, що тестується, шляхом перевірки та верифікації. Тестування програмного забезпечення також може забезпечити об’єктивне, незалежне уявлення про програмне забезпечення, щоб дозволити бізнесу оцінити та зрозуміти ризики впровадження програмного забезпечення. Методи тестування включають, але не обов’язково обмежуються:

- аналіз вимог до продукту щодо повноти та правильності в різних контекстах, таких як галузева перспектива, бізнес-перспектива, доцільність та життєздатність впровадження, зручність використання, продуктивність, безпека, міркування інфраструктури тощо.
- перегляд архітектури продукту та загального дизайну продукту
- робота з розробниками продуктів над удосконаленням техніки кодування, шаблонів проектування, тестів, які можна написати як частину коду на основі різних методів, таких як граничні умови тощо.
- виконання програми або програми з метою перевірки поведінки
- перевірка інфраструктури розгортання та пов’язаних скриптів та автоматизації
- брати участь у виробничій діяльності, використовуючи методи моніторингу та спостереження

Тестування програмного забезпечення може надати користувачам або спонсорам об’єктивну, незалежну інформацію про якість програмного забезпечення та ризик його збою.

Помилки програмного забезпечення виникають через наступний процес: Програміст робить помилку (помилку), що призводить до помилки (дефекту,

помилки) у вихідному коді програмного забезпечення. Якщо ця помилка виконана, у певних ситуаціях система дасть неправильні результати, що призведе до збою.

Не всі несправності обов'язково призведуть до збоїв. Наприклад, помилки в мертвому коді ніколи не призведуть до збоїв. Несправність, яка не виявила збоїв, може призвести до збою при зміні середовища. Приклади цих змін у середовищі включають програмне забезпечення, яке запускається на новій апаратній платформі комп’ютера, зміни вихідних даних або взаємодію з іншим програмним забезпеченням. Одна несправність може призвести до широкого спектру симптомів відмови.

Не всі помилки програмного забезпечення викликані помилками кодування. Одним із поширених джерел дорогих дефектів є прогалини у вимогах, тобто нерозпізнані вимоги, які призводять до помилок, пропущених розробником програми. Прогалини вимог часто можуть бути нефункціональними вимогами, такими як тестованість, масштабованість, ремонтопридатність, продуктивність та інші вимоги. безпеки.

У тестуванні програмного забезпечення існує **багато підходів**. Огляди, покрокові інструкції або перевірки називаються статичним тестуванням, тоді як виконання запрограмованого коду з заданим набором тестових випадків називається динамічним тестуванням.

Статичне тестування часто є неявним, як-от коректура, а також коли інструменти програмування/текстові редактори перевіряють структуру вихідного коду, а компілятори (попередні компілятори) перевіряють синтаксис і потік даних як статичний аналіз програми. Динамічне тестування відбувається під час запуску самої програми. Динамічне тестування може розпочатися до того, як програма буде на 100% завершена, щоб перевірити окремі розділи коду та застосувати до дискретних функцій або модулів. Типовими методами для них є або використання заглушок/драйверів, або виконання з середовища налагодження.

Статичне тестування передбачає перевірку, тоді як динамічне також передбачає перевірку.

Пасивне тестування означає перевірку поведінки системи без будь-якої взаємодії з програмним продуктом. На відміну від активного тестування, тестувальники не надають жодних тестових даних, а переглядають системні журнали та трасування. Вони шукають моделі та специфічну поведінку, щоб приймати якісь рішення. Це пов’язано з перевіркою часу виконання в автономному режимі та аналізом журналу.

Дослідницьке тестування — це підхід до тестування програмного забезпечення, який коротко описується як одночасне навчання, розробка тесту та виконання тесту. Джем Канер, який ввів цей термін у 1984 році,:2 визначає дослідницьке тестування як «стиль тестування програмного забезпечення, який підкреслює особисту свободу та відповідальність окремого тестувальника за постійну оптимізацію якості своєї роботи, розглядаючи тест- пов’язане навчання, дизайн тесту, виконання тесту та інтерпретація результатів тесту як взаємодопоміжні дії, які виконуються паралельно протягом усього проекту».

Методи тестування програмного забезпечення традиційно поділяються на **тестування білого та чорного ящиків**. Ці два підходи використовуються для опису точки зору, яку дотримується тестувальник під час розробки тестових випадків. До методології тестування програмного забезпечення також можна застосувати гібридний підхід, який називається тестуванням сірого ящика. Оскільки концепція тестування сірого ящика, яка розробляє тести на основі конкретних елементів дизайну, стає все більш популярною, це «довільне розходження» між тестуванням чорного та білого ящиків дещо зникло.

Таблиця 4.1 — Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Невірні данні при авторизації	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Пусті поля при авторизації	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Неспівпадаючі паролі при реєстрації	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.

Закінчення таблиці 4.1

№	Тест-кейс	Очікуваний результат	Отриманий результат
4	Пусті поля при реєстрації	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
5	Створення девайсу з пустими полями	При спробі створення девайсу з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення девайсу з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

4.2 Огляд отриманих результатів

Для проведення аналізу отриманих результатів і результативності розробки в цілому було вирішено порівняти розроблену систему з виявленими раніше аналогічними рішеннями.

Результати отриманого аналізу представлені в таблиці 4.2.

Таблиця — 4.2 Порівняльний аналіз аналогів

	Sky River	Find My Kids	Family Locator	Kids Control	Власне рішення
Простота інтерфейсу	5	8	6	7	9
Збереження маршруту	-	+	+		+
Наявність карти сховищ	-	-	-	-	+
Побудова безпечного маршруту	-	-	-	-	+
Realtime-трекінг	-	+	+	-	+
Безкоштовне використання	-	+	-	+	+

Висновки до розділу 4

В ході написання четвертого розділу було проведено тестування розробленої системи, яке дало розуміння про повну функціональність додатку. Окрім цього було проведено аналіз отриманих результатів, в ході якого було створено порівняльну таблицю, на базі якої можна зробити висновок, що розроблена система суттєво виграє у конкурентів в даній ніші. Фінальне тестування системи і порівняльний аналіз з наявними рішеннями довели адекватність функціональності системи й підтвердили вирішення проблем аналогів, що були виявлені в ході їх огляду.

ВИСНОВКИ

В ході написання першого розділу роботи було виявлено значення основних понять предметної області, таких як прикладне програмне забезпечення і GPS-трекінг, які в подальшому стануть основою для розуміння концепції розробки. Окрім цього проведено аналіз існуючих рішень з метою виявлення їх слабких сторін і вирішення знайдених проблем у власній реалізації.

Другий розділ роботи представляє собою проектування розроблюваної системи. В ході проектування було обрано архітектуру майбутньої системи, проведено проектування внутрішньої будови і структури бази даних.

Третій розділ містить в собі інформацію стосовно розробки програмного продукту, таку як опис використаних алгоритмів та їх реалізації, вибір інструментальних засобів розробки, створення графічного інтерфейсу користувача.

Четвертий розділ представляє собою тестування системи і аналіз отриманих результатів. Аналіз отриманих результаті представлено у вигляді зведеної порівняльної таблиці виявлених конкурентів і власної розробки. В результаті виявлено суттєву перевагу розробленої системи.

Мета дослідження полягала в створенні власної програмної реалізації системи GPS-трекінгу.

Для досягнення поставленої мети було виконано наступні завдання:

- проведено аналіз основних питань предметної області;
- проведено аналіз існуючих рішень;
- сформовано задачу для розробки;
- проведено аналіз варіантів використання системи;
- створено структуру бази даних;
- розроблено схему внутрішньої будови системи;

- обрано архітектуру системи;
- проведено аналіз інструментальних засобів розробки;
- розроблено графічний інтерфейс користувача;
- проведено тестування системи;
- проведено аналіз отриманих результатів.

Завдяки виконанню даних завдань в результаті роботи отримано повноцінну клієнт-серверну систему GPS-трекінгу, яка протестована, відлагоджена і готова до використання в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "WGS84". wiki-georef.
2. "GPS Cycle Computer v3". Axivo Inc. 7 November 2008. Retrieved 22 April 2014.
3. "Free Eriadne.org tracking system". Archived from the original on 9 January 2010. Retrieved 3 December 2010.
4. "open-gpstracker A GPS tracking Android App: Build to be extensible and Free". Retrieved 11 October 2009.
5. "Traccar Client - free open source Android tracker". Retrieved 15 August 2012.
6. "State budget committee rejects GPS monitoring expansion". Wisconsin State Journal. Retrieved 14 January 2014.
7. "Wisconsin Legislature 165.94". Wisconsin Legislature. Retrieved 14 January 2014.
8. Tovia Smith (29 August 2006). "Technology Lets Parents Track Kids' Every Move". NPR. Retrieved 25 April 2011.
9. Mark (5 September 2017). "GPS Tracking Device for Kids". ossaward. Retrieved 25 December 2017.
10. Karonis, George (24 September 2016). "How All Satellite Based GPS Trackers Work". liveviewgps.com. Retrieved 8 October 2017.
11. "How GPS helped Paul O'Connell win Six Nations player of the tournament at 35".
12. "The GPS tweak which Sale hope can get Tuilagi back with England".
13. "Rugby technology: How do England use GPS?". YouTube.
14. Rice, Clifford G. (2016). "Development of a system for remotely monitoring vaginal implant transmitters and fawn survival". Wildlife Biology. 22: 22–28. doi:10.2981/wlb.00177.

15. Stabach, Jared (1 April 2020). "Fantastic Beasts of the Great Plains". Smithsonian's National Zoo and Conservation Biology Institute. Retrieved 15 January 2021.
16. Brownings, Jeff (11 December 2020). "NSW GPS Legality Act 2007". Retrieved 8 March 2021.
17. Claburn, Thomas (4 March 2009). "Court Asked To Disallow Warrantless GPS Tracking". InformationWeek. Retrieved 18 March 2009.[permanent dead link]
18. "State of Tennessee PUBLIC CHAPTER NO. 169 HOUSE BILL NO. 457" (PDF). fas.org. State of Tennessee. Retrieved 1 January 2019.
19. "California Penal Code Section 637.7 - California Attorney Resources - California Laws". law.onecle.com. Retrieved 25 March 2017.
20. Koch, Wendy (12 February 2009). "Cheatin' hearts pump up economy on Valentine's Day". USA Today. Retrieved 5 April 2010.
21. Bello, Marisol (12 December 2008). "GPS, hidden cameras watching over Baby Jesus". USA Today. Retrieved 4 May 2010.
22. "Legis.ga.gov". Archived from the original on 15 February 2009.
23. "Georgia Bill Would Ban Hidden GPS Tracking Devices". WSB-TV. 29 January 2009. Archived from the original on 15 May 2011. Retrieved 8 April 2011.
24. "What is personal data? – A quick reference guide". ICO.ORG.UK. Information Commissioner's Office. Retrieved 15 September 2014.
25. "A brief guide to notification". ICO.GOV.UK. The Information Commissioner's Office. Retrieved 15 September 2014.
26. "Government officials track cars and trespass on private property, report shows". The Telegraph. 21 September 2009. Retrieved 14 September 2014.
27. "Stalking and Harassment". Crown Prosecution Service. Retrieved 15 September 2014.

28. "Animal campaigner claims car was bugged by Bernard Matthews". The Mirror Newspaper. 22 December 2011. Retrieved 14 September 2014.
29. "Code of practice for covert surveillance and property interference". GOV.UK. GOV.UK.
30. Wentz, Laurel (29 July 2010). "Is Your Detergent Stalking You?". Retrieved 7 August 2010.

Додаток А

Лістинг програмного коду

```
package com.joreijarr.studycontrol.controllers;



import com.joreijarr.studycontrol.models.Device;
import com.joreijarr.studycontrol.repo.DeviceRepository;
import org.springframework.web.bind.annotation.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

@Controller
public class startController {




    @Autowired
    DeviceRepository deviceRepository;

    @GetMapping("/")
    public String index(Model model)
    {
```

```
    model.addAttribute("title", "Головна");
    Iterable<Device> devices_it = deviceRepository.findAll();
    List<Device> dev = new ArrayList<>();
    devices_it.forEach(dev::add);
    model.addAttribute("markers", dev);
    return "index_user";
}



@GetMapping("/device")
public String device(Model model)
{
    model.addAttribute("title", "Девайси");
    Iterable<Device> devices_it = deviceRepository.findAll();
    List<Device> dev = new ArrayList<>();
    devices_it.forEach(dev::add);
    model.addAttribute("devices", dev);
    return "device_page";
}

@GetMapping("/device/add")
public String deviceAdd(Model model)
{
    model.addAttribute("title", "Новий девайс");
    return "device_add";
}
```

```
@PostMapping("/device/add")
public String deviceAdd_post(Model model,
                             @RequestParam String name,
                             @RequestParam String status,
                             @RequestParam String description,
                             @RequestParam String last_lattitude,
                             @RequestParam String last_longnittude)
{
  Device device = new Device(name, status, description,last_lattitude,last_longnittude);
  deviceRepository.save(device);
  return "redirect:/device";
}



@RequestMapping("/device/changeLatLng/{name}/{Lat}/{Lng}")
public String api(Model model, @PathVariable String name,
                              @PathVariable String Lat,
                              @PathVariable String Lng)
{
  String timeStamp = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(Calendar.getInstance().getTime());
  Device device = deviceRepository.findByName(name);
  device.setLast_lattitude(Lat);
  device.setLast_longnittude(Lng);
  deviceRepository.save(device);
```

```
        model.addAttribute("date", timeStamp);
        return "api_ok";
    }




}
package com.joreijarr.studycontrol.models;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Device {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long device_id;

    public String name, status, description, last_lattitude, last_longnittude;

    public Device() {
    }

    public Device(String name, String status, String description, String last_lattitude, String last_longnittude) {
        this.name = name;
```

```
        this.status = status;
        this.description = description;
        this.last_lattitude = last_lattitude;
        this.last_longnittude = last_longnittude;
    }

    public Long getDevice_id() {
        return device_id;
    }

    public void setDevice_id(Long device_id) {
        this.device_id = device_id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
```

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getLast_lattitude() {
        return last_lattitude;
    }

    public void setLast_lattitude(String last_lattitude) {
        this.last_lattitude = last_lattitude;
    }

    public String getLast_longnittude() {
        return last_longnittude;
    }

    public void setLast_longnittude(String last_longnittude) {
        this.last_longnittude = last_longnittude;
    }
}
package com.joreijarr.studycontrol;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class StudycontrolApplication {

	public static void main(String[] args) {
		SpringApplication.run(StudycontrolApplication.class, args);
	}

}
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title th:text="${title}"/>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
<header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
  <h1>Новий девайс</h1>
  <form method="post">
    <input type="text" name="name" placeholder="Введіть назву девайсу" class="form-control">
    <br>
    <input type="text" name="status" placeholder="Введіть статус" class="form-control">
```

```
            <br>
            <input type="text"  name="description" placeholder="Введіть опис" class="form-control">
            <br>
            <input type="text"  name="last_lattitude" placeholder="Введіть останню широту" class="form-control">
            <br>
            <input type="text"  name="last_longnittude" placeholder="Введіть останню довготу" class="form-control">
            <br>
            <button class="btn btn-success" type="submit">Додати девайс</button>
        </form>
    </div>
    </body>
    </html>
    <!DOCTYPE HTML>
    <html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title th:text="${title}"/>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
    </head>
    <body>
    <div class="container py-3">
        <header th:insert="blocks/header_user :: header_user"></header>
    </div>
    <div class="container mt-5">
```

```
    <div th:each="el: ${devices}" class="alert alert-info mt2">
        <h3 th:text="${el.name}"/>
        <h5 th:text="${el.status}"/>
        <h5 th:text="${el.description}"/>
    </div>

    <a href="/device/add" class="btn btn-success">Додати девайс</a>
</div>
</body>
</html>
```