МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем

«До захисту в ЕК»		**«До захисту допущено»**	
Директор інституту(декан факультету)		Завідувач кафедри	
_____ (підпис)	Андрій Форсюк (ім'я та прізвище)	_____ (підпис)	Сергій Чумаченко (ім'я та прізвище)
«___» _____ 20__р.		«___» _____ 20__р.	

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Інформаційні управляючі системи та технології
на тему: Розроблення інформаційної системи моніторингу відвідуваності працівників підприємства

Виконав: здобувач 2 курсу, групи ІС-2-3М

Мохонько Іван Олександрович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Костіков Микола Павлович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Володимир Овчарук _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач. _____
(підпис)

Київ - 2022 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _____Автоматизації і комп'ютерних систем_____
Кафедра _____Інформаційних систем_____
Освітній ступінь_магістр_____
Спеціальність_122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма ____Інформаційні управляючі системи_____
та технології_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем

__Чумаченко С.М._____
"11" листопада 2022 року_____

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Мохонька Івана Олександровича_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Розроблення інформаційної системи моніторингу_____ відвідуваності працівників підприємства_____
керівник роботи_Костіков Микола Павлович к.т.н. доц._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 11 листопада 2021 року № 884-кс
2. Строк подання здобувачем роботи___4 лютого 2022 р._____
3. Вихідні дані до роботи_дані про працівників, поточні встановлені системи для контролю доступу та моніторингу, інформація про підприємство_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Основні положення та складнощі під час реалізації моніторингу для підприємства_____
Розділ 2. Огляд поточних рішень для моніторингу відвідуваності працівників підприємства_____
Розділ 3. Приклад реалізації системи моніторингу відвідуваності працівників підприємства_____
5. Перелік графічного матеріалу формули, схеми, модель бази даних, скріншоти інтерфейсу та приклади коду_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Костіков М. П., доц.	11.11.2021	12.11.2021
2.	Костіков М. П., доц.	25.11.2021	25.11.2021
3.	Костіков М. П., доц.	30.01.2022	02.02.2022

7. Дата видачі завдання___11.11.2021_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Розгляд і аналіз теоретичних основ	12.11.2021-24.11.2021	
2	Дослідження та аналіз схожих систем аналогів	24.11.2021-07.12.2021	
3	Дослідження та аналіз технологій для розробки системи	07.12.2021-28.12.2021	
4	Розроблення системи моніторингу відвідуваності працівників підприємства	28.12.2021 - 12.01.2022	
5	Оформлення тексту пояснювальної записки	12.01.2022-24.01.2022	
6	Оформлення автореферату	24.01.2022-29.01.2022	
7	Оформлення презентації	29.01.2022-07.02.2022	

Здобувач _____ Мохонько І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ доц. Костіков М. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Прізвище та ініціали здобувача: Мохонько І.О.

Назва магістерської роботи: «Розроблення інформаційної системи моніторингу відвідуваності працівників підприємства»

Магістерська робота містить 63 сторінки, 13 рисунків, 57 літературних джерел.

Основною метою є створення системи моніторингу працівників підприємста що дозволить покращити процеси пов’язані з менеджементом відвідування та самого персоналу. Основним завдання цієї системи є впровадження процесів відстежування працівників під час робочого дня та відправлення їм повідомлень у разі потреби. Також надати інтуїтивний інтерфейс для роботи з цими даними як для менеджерів так і для самого персоналу. Дозволити строїти графіки відвідуваності, переміщення по підприємству та формувати деяку статистику для подальшого покращення процесів які з цим пов’язані.

КЛЮЧОВІ СЛОВА: ВІЗУАЛІЗАЦІЯ ДАНИХ, VUE, JAVASCRIPT, HTML, CSS, NODE.JS, GRAPHQL

ANNOTATION

Surname and initials of the applicant: Mokhonko I.O.

The title of master's work: “Employees monitoring information system development".

The master's thesis contains 63 pages, 13 pictures, 57 literary sources.

The main purpose of this work is to develop a system for monitoring employees. Those improvements will improve and ease processes related to management of the personnel. The main goal of this system is to implement employees tracking during their work time and sending messages if needed. Also to introduce intuitive user interface for working with this data for managers and employees itself. Provide support for rendering attendance charts, map of employee movements inside the facility or office based on gathered data and build useful statistics for further improvements of related processes.

KEYWORDS: DATA VISUALIZATION, VUE, JAVASCRIPT, HTML, CSS, NODE.JS, GRAPHQL

ЗМІСТ

ВСТУП

Актуальність теми. Наразі найбільш популярним способом моніторнигу відувідування є турнікети де кожен працівник відмічається при заході та виході, що дає деяку інформацію про час його перебування, також якщо є двері, які можуть бути встановлені по всьому офісу чи підприємству та мають таку ж саму пропускну систему. Вони надають можливість дивитися в який час та ким було здійснено вхід в якусь кімнату. Цей спосіб моніторингу не є поганим, та деяким підприємствам він є цілком задовільним, але більшість таких системи не мають якогось стандартизованого програмного забезпечення для перегляду цих даних або зовсім не надають таку інформацію. Також для цього потрібно встановлювати обладнання що несе за собою фінансові витрати.

Дана робота є прикладом розкриття сутності проблеми моніторингу працівників на підприємствах. В ці часи пандемії та періоду після неї більшість працівників були вимушені працювати з дому, але деякі випадки не дозволяють цього робити, наприклад ті які повинні забезпечувати продукцією постійно, незалежно від ситуації. В таких випадках присутність працівників є вимогою та не може бути змінена. Створення такої системи яка б змогла надати достатній рівень моніторингу та статистики є не тільки впровадженням покращень, а й деякою необхідністю для розвитку.

В наші часи майже кожна людина при собі має смартфон, в свою чергу цей девайс дозволяє встановлювати та розроблювати майже будь-яки додатки та реализовувати будь який функціонал, якщо він постійно підключений до мережі Інтернет це дає дуже багато можливостей. Наприклад, система геолокації, яка дозволяє бачити позицію телефону на карті та відстежувати її зміну в реальному часі з точністю до метрів. Знаючи про це ми можемо використати це замість звичного турнікета. Система дозволяє при приході на роботу відмічатися та автоматично через геолокацію перевіряти місцезнаходження працівника. Одним тільки цим функціоналом підприємство

може економити на встановленню та технічному підтриманні турнікетів. З іншого боку якщо роль турнікета є більше це дозвіл на вхід тільки працівникам підприємста то повністю відовитися від нього буде неможливо, в такому випадку потрібно буде мати можливість підключити ці турнікети до системи та записувати кожен вхід та вихід. Навідміну від звичайних турнікетів система дозволяє дуже тонкі налаштування та дає простір для його розширення що може бути виконано під кожні потреби в індивідуальному порядку.

Якщо підприємство це велика територія, використовуючи геопозицію яка в реальному часі відстежується можливо буде відразу знайти ту людину яка потрібна, або швидко сповістити когось про те що хтось зайшов в зону в яку йому не дозволено заходити. І все це в смартфоні!

Мета дослідження. Розробити проектні рішення та концепцію для системи моніторингу відвідуваності працівників підприємства.

Завдання дослідження:

1. Проаналізувати основні проблеми під час створення подібних систем для моніторингу
2. Провести аналіз існуючих рішень та зробити порівняльну характеристику.
3. Виявити недоліки та переваги існуючих рішень або систем.
4. Створити робочий прототип такої системи та запропонувати концепії її подальшого розвитку.

Методи дослідження. Основним методом дослідження був аналіз існуючих проблем, що виникають при впровадженні подібних систем. Порівняльний аналіз існуючих систем був використаний для формування та отримання загальної оцінки стану цієї проблеми. Був проведений критичний аналіз джерел по цій тематиці.

Наукова новизна одержаних результатів. У даній роботі основною науковою новизною є концепція створення однієї системи яка включає в себе всі необхідні інструменти та функціонал для моніторингу працівників підприємства, починаючи від простого пропуску через турнікет та закінчуючи

збором даних про зміну геопозиції працівника під час робочого дня та подальше використання зібраних даних в цілях покращеннях аспектів кофморту праці з урахуванням специфіки конкретного підприємства.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій. Наразі в Україні поступово займаються переведенням всіх послуг в онлайн простір, де не потрібно стояти в чергах та приносити роздруковані документи. Всі ці дії є природньою еволюцією розвитку цивілізованих держав. Система моніторингу працівників не є виключенням, вона також автоматизує один із тих процесів які десятиліттями стояв на одному місці. Тому вона буде одним із тих покращень яке дозволить зробити життя простішим не тільким працівникам, а й самому підприємству.

Всі ці можливості є великою перевагою перед устарівшими методами. Вони дозволяють вивести підприємство на більший рівень технологічності та діджиталізувати процеси які до цього були майже неможливі.

Наукове значення роботи. Основною метою є створення системи моніторингу працівників підприємста що дозволить покращити процеси пов’язані з менеджементом відвідування та самого персоналу. Основним завдання цієї системи є впровадження процесів трекінгу працівників під час робочого дня та відправлення їм повідомлень у разі потреби. Також надати інтуїтивний інтерфейс для роботи з цими даними як для менеджерів так і для самого персоналу. Дозволити строїти графіки відвдуваності, переміщення по підприємству та формувати деяку статистику для подальшого покращення процесів які з цим пов’язані.

Одним із плюсів такого підходу є можливість накопичення даних про переміщення працівників під час робочого дня. На основі цього можливо створювати карти де відмічені місця найбільного скупчення людей або місця де більше за все люди проводять робочого часу. Наприклад, під час пандемії, ми можемо відслідковувати як в реальному часі проходять скупчення людей, або переглянути коли і хто був в такому скупченні. Маючи таку статистику буде можливість попередити, тих хто приймав у цьому участь, у великому ризику

зараженні, оскільки хтось був на момент скупчення вже хворий. Після цього ризик подальшого поширення хвороби буде менший. Або, ми можемо автоматично відправляти нагадування про те що потрібно одягнути маску та сфотографуватися в ній тим хто зараз знаходиться в цьому скупченні.

Маючи такі можливості по контролю ми не тільки можемо зменшувати ризики розповсюдження хвороб, а й зменшити ризики таких спалахів у майбутньому. І це тільки один випадок їх використання, маючи ці дані ми можемо створюти будь-які передбачення або визначати деякі тендеції. Також система відкрита для розширення, та дозволяє будь-яким розробникам створювати свої або інтегрувати вже існуючі програмні рішенні на основі цих даних.

Практичне значення отриманих результатів. Результати які ми отримаємо після використання системи не тільки впроваджують покращення для підприємства яке її використовує, а й для людства та природи вцілому, оскільки на основі цих даних ми можемо відстежувати такі складні та випадкові процеси як розповсюдження хвороби.

Особистий внесок здобувача. Дослідження невирішених проблем в цій області, аналіз сучасних засобів для її вирішення та створення нових концепцій та архітектури системи моніторингу відвідуваності працівників були проведені здобувачем особисто.

Публікації. Тези доповіді з теми магістерської роботи було опубліковано у збірнику матеріалів студентської науково-практичної конференції “Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами” в Національному університеті харчових технологій.

ОСНОВНІ ПОЛОЖЕННЯ ТА СКЛАДНОСЩІ ПІД ЧАС РЕАЛІЗАЦІЇ МОНІТОРИНГУ ДЛЯ ПІДПРИЄМСТВ

В сучасному світі все більше аспектів людської діяльності переходять в сферу інформаційних технологій. Банки ведуть свою діяльність по більшій мірі в інтернеті, надаючи послуги через свої веб-сайти, а підприємства зберігають свою бухгалтерію використовуючи спеціальне програмне забезпечення замість друку паперу із записами. Але деякі аспекти все ще залишаються такими як і були в часи свого зародження, моніторинг працівників на підприємства не є виключенням.

Такий важливий аспект як моніторинг працівників є дуже важливою частиною правильного функціонування самого підприємства. Не приймаючи ніяких мір по менеджменту та контролю за працівниками швидко приведе до поганих наслідків для бізнесу. Наразі майже кожна фірма яка має офіс чи філію має деяку систему пропуску або ідентифікації працівників які в ній працюють для виключення випадків несанкіонованого доступу до приміщень посторонніх осіб.

Але такі системи використовуються не тільки для ідентифікації своїх працівників, а і для будь-кого. Наприклад, метро, де для доступу до поїздів спочатку потрібно оплатити вхід на турнікеті, в іншому випадку у доступі буде відмовлено.

Отже, при проектуванній та розробці засобів для менджменту та моніторингу потрібно враховувати особливості конкретного підприємства на якому вона буде впроваджена. Також потрібно дозволити інтегрувати рішення які вже впровадженні для поступового переходу або частичного використання створеної системи для максимізації можливостей які будуть надані після впровадження.

Система позитивно впливає на різні сторони роботи підприємства або компанії, насамперед на підвищення продуктивності та безпеки. Завдяки

функціям, які дозволяють відслідковувати активність працівників (наприклад, відпрацьований час, кількість перерв, залучення до роботи) можна бути впевненим, що команда продуктивно працює. Такі функції, як відстеження використання Інтернету, сайтів та програм, допомагають переконатися, що команда не переглядає контент, який не належить до роботи.

Хто отримує вигоду від запровадження:

- Власник (керівник підприємства);
- Керівники відділів безпеки/служба охорони;
- Керівники ІТ;
- Начальник відділу кадрів;
- Бухгалтер із розрахунку ЗП.

Основною технологією яка використовується для моніторингу працівників є відстежування геопозиції телефона, яка дозволяє в реальному часі бачити позицію на карті.

ОГЛЯД ПОТОЧНИХ РІШЕНЬ ДЛЯ МОНІТОРИНГУ ВІДВІДУВАНОСТІ ПРАЦІВНИКІВ ПІДПРИЄМТСВА

Проаналізуємо рішення для моніторингу відвідуваності які наразі доступні на ринку за їхнім функціоналом та оцінимо їх плюси та мінуси.

Візьмемо для прикладу систему від компанії “Салютем Україна”. Їхні рішення дозволяють встановлювати систему контролю доступу та обліку робочого часу яку можно використовувати в офісах, бізнес центрах та на підприємствах. Також їхні системи дозволяють проводити тестування на вміст алкоголю в крові, біометричний контроль, управління ліфтами та холодними приміщеннями.

Компанія пропонує:

- формування ТЗ під конкретний випадок;
- формування специфікації та підбору обладнання;
- документування проекту;
- монтаж та надання документів на використання;
- уавчання персоналу та налаштування системи;
- технічну підтримку системи та обладнання.

Розглянема для прикладу систему обліку робочого часу. Одним з основних мотивів її використання в організації є потреба керівників у контролі та обліку робочого часу працівників. Інформація про проходи, через обладнані точки доступу, зберігається в базі даних системи і в подальшому може бути використана для створення різних звітів, що дозволяє вести повний облік та аналіз робочого часу, задавати різні критерії пошуку - запізнення, переробки, з можливістю поділу за підрозділами за будь-який проміжок часу. В стандартний пакет послуг входять:

Вихід з робочого місця раніше

Дозволяє виявити наявність або відсутність випадків виходу працівників раніше встановленого часу, що дає змогу оперативно виявляти факти

порушення трудової дисципліни. Звіт будується за період одного місяця. При налаштуванні даного типу звітів є можливість вказувати допустимий період виходу раніше часу (у хвилинах), який не буде зарахований як порушення та відображений у звіті.

Звіт по віздхиленням

Дозволяє отримати узагальнену інформацію щодо всіх порушень трудової дисципліни за вказаний період часу (день, тиждень, місяць). Звіт відображає відсутність пари вхід-вихід, запізнення, виходу раніше часу, переробки, відсутності, виходу протягом робочого дня (якщо людина перевищила період зазначеної нештрафованої відсутності, але повернулася на робоче місце до кінця робочого часу). Для цього типу звітів можуть застосовуватися параметри нормалізації. Звіт використовується керівниками для повного моніторингу за дотриманням працівниками трудової дисципліни.

Вхід та вихід за місяць

По кожному співробітнику, згідно з вибудованим графіком, ведеться облік робочого часу за трьома показниками: час приходу, час виходу та відпрацьований час з урахуванням нормалізації. Нормалізація — приведення даних зафіксованих системою до виду придатного для розрахунку відпрацьованого часу (усунення конфліктів, неоднозначностей тощо). пари), є можливість задати політику, за якою розраховуватиметься загальний робочий час (перший вхід-останній вихід тощо) або можливість вказати заздалегідь час відсутності, що не штрафується.

Звіт по запізненням

Відображає дані (у хвилинах) про запізнення на роботу за вибраний звітний період.

Звіт по відвідуванню

Цей звіт дозволяє проводити облік присутності працівників на робочому місці за період призначеного робочого часу. Звіт відображає присутність працівників із можливістю деталізації отриманих подій з обладнання та відображення подій для розрахунку робочого часу. У звіті також

відображається фактична відсутність співробітника та всі інші події згідно з поправками до робочого часу (відрядження, відпустка, лікарняний тощо). Звіт можна сформувати за поточний чи будь-який інший обліковий місяць.

Табель обліку робочого часу

Табель протягом місяця є добре знайомий бухгалтерам табель обліку робочого дня формою Т-13. Обґрунтовує нарахування працівникам заробітної плати відповідно до кількості відпрацьованих днів та годин за місяць.

Табель обліку робочого часу (за неділю)

Спрощена форма табеля за місяць, що вказує на основні моменти: час приходу та догляду, загальна кількість відпрацьованих годин, використовується для моніторингу трудової дисципліни працівників за тижневий період.

Диференціальний звіт

Диференціальний звіт дозволяє оцінити ефективність використання робочого часу працівника шляхом визначення того, яку частину цього часу він провів на території підприємства в принципі, і яку частину цього часу — безпосередньо на робочому місці.

Завдяки вбудованій системі поправок у відповідальних осіб існує можливість звести до мінімуму можливість виникнення похибок у разі непередбаченого відрядження, незапланованої відпустки, лікарняного тощо. Так, наприклад, при обліку робочого часу працівника, який відсутній на місці через хворобу, можна вручну внести відповідні коригування в систему, які будуть враховані при формуванні (місячного) табеля обліку робочого часу.

Незважаючи на те, що базовий функціонал досить великий, у ньому передбачена можливість редагування та доопрацювання існуючих звітів під індивідуальні завдання та переваги кінцевого користувача.

Переваги

- Розрахунок зарплати, кадровий облік та управління доступом в одній програмі.
- Отримання найточнішої інформації з трудової дисципліни.

- Відсутність дублювання даних у різних системах.
- Значне скорочення витрат на складання табелів робочого часу та розрахунок зарплати.
- Можливість аналізу ефективності роботи офісних працівників.
- Узгодженість організації доступу співробітників до різних приміщень та кадрових змін.

Недоліки

- Відносно висока вартість.
- Постійна технічна підтримка обладнання.
- Неможливість швидкої заміни всього обладнання.
- Неможливість визначення точного місцезнаходження працівників

Розглянему систему контролю доступу та моніторингу працівників від компанії “Вал Тек”. Рішення цієї компанії надає майже всі послуги які надавала поперядня компанія, та надає більше покращену систему для ведення бази персоналу та відвідувачів. Також присутня можливість інтеграції вже з існуючими системами безпеки або моніторнигу, наприклад:

- з системою відеоспостереження для поєднання архівів подій систем, передачі системі відеоспостереження сповіщень про необхідність
- з системою охоронної сигналізації (СОС), для обмеження доступу в приміщення, що стоять на охороні, або для автоматичного зняття і постановки приміщень на охорону.
- з системою пожежної сигналізації (СПС) для отримання інформації про стан пожежних сповіщувачів, автоматичного розблокування евакуаційних виходів і закривання протипожежних дверей у разі пожежної тривоги.
- стартувати запис, повернути камеру для запису наслідків зафіксованої підозрілої події;

У мережевій СКУД всі контролери з'єднані з центральним сервером. Мережеві системи зручні для великих об'єктів (офіси, виробничі підприємства), оскільки керувати навіть десятком дверей, на яких встановлені автономні системи, стає надзвичайно важко. Незамінні мережеві системи в наступних випадках:

- якщо необхідна інформація про події, що відбулися раніше або потрібен додатковий контроль в реальному режимі часу. Наприклад, в мережевій системі існує функція фотоверифікації: на прохідній при піднесенні людиною, що увійшла, ідентифікатора до зчитувача, службовець (вахтер, охоронець) може на екрані монітора бачити фотографію людини, якій в базі даних привласнений даний ідентифікатор, і порівняти із зовнішністю минаючого, що підстраховує від передачі карток іншим людям;
- якщо необхідно організувати облік робочого часу і контроль трудової дисципліни;
- якщо необхідно забезпечити взаємодію (інтеграцію) з іншими підсистемами безпеки, наприклад, відеоспостереженням або пожежною сигналізацією.

У мережній системі з одного місця можна не тільки контролювати події на всій території, що охороняється, а й централізовано керувати правами користувачів, вести базу даних. Мережеві системи дозволяють організувати кілька робочих місць, розділивши функції управління між різними співробітниками і службами підприємства.

У мережевих системах контролю доступу можуть застосовуватися бездротові технології (радіоканали). Використання бездротових мереж найчастіше визначається конкретними ситуаціями: складно або неможливо прокласти дротові комунікації між об'єктами, скорочення фінансових витрат на монтаж точки проходу і т.д. Існує велика кількість варіантів радіоканалів, проте в СКУД використовуються тільки деякі з них.

- **Bluetooth.** Даний вид бездротового пристрою передачі даних являє собою аналог Ethernet. Його особливість полягає в тому, що відпадає необхідність прокладати паралельні комунікації для об'єднання компонентів при використанні інтерфейсу RS-485.
- **Wi-Fi.** Основна перевага даного радіоканалу полягає у великій дальності зв'язку, яка здатна досягати декількох сотень метрів. Це особливо необхідно для з'єднання між собою об'єктів на великих відстанях. При цьому скорочуються як тимчасові, так і фінансові витрати на прокладку вуличних комунікацій.
- **ZigBee.** Спочатку сферою застосування даного радіоканалу була система охоронної та пожежної сигналізації. Технології не стоять на місці і активно розвиваються, тому ZigBee може використовуватися і в системах контролю доступу. Дана бездротова технологія працює в неліцензованому діапазоні 2,45 ГГц.
- **GSM.** Перевага використання даного бездротового каналу зв'язку & nbsp; - практично суцільне покриття. До основних методів передачі інформації в розглянутій мережі відносяться GPRS, SMS і голосовий канал.

Автономні СКУД.

Автономні системи дешевше, простіше в експлуатації, не вимагають прокладки сотень метрів кабелю, використання пристроїв сполучення з сервером, самого сервера. При цьому до мінусів таких систем відноситься неможливість створювати звіти, вести облік робочого часу, передавати й узагальнювати інформацію про події, управлятися дистанційно. При виборі автономної системи з високими вимогами щодо безпеки рекомендується звернути увагу на наступне:

- Зчитувач повинен бути відділений від контролера, щоб дроти, по яких можливо відкривання замку, були недоступні зовні.

- Контролер повинен мати резервне джерело живлення на випадок відключення електроживлення. Переважно використовувати зчитувач в вандалозахисному корпусі.

- У складі автономної системи контролю доступу використовуються також електронні замки, передають інформацію по бездротових каналах зв'язку: в двері встановлюється механічний замок з електронним управлінням і вбудованим зчитувачем.

- Замок по радіоканалу пов'язаний з хабом, який вже по проводах обмінюється інформацією з робочою станцією, на якій встановлено програмне забезпечення.

У таких системах можно виділити такі пристрої:

Перегороджуючі пристрої

Встановлюються на двері:

- Електрозщіпки — найменш захищені від злому, тому їх зазвичай встановлюють на внутрішні двері (внутрішньоофісні і т. п.) Електрозащіпки, як і інші типи замків, бувають електричні, що відкриваються (тобто двері відкриваються при подачі напруги живлення на замок), і електрично закриваються (відкриваються, як тільки з них знімається напруга живлення, тому рекомендовані для використання пожежною інспекцією).
- Електромагнітні замки - практично всі замикаються при подачі на них електроживлення, тобто придатні для установки на шляхах евакуації при пожежі.
- Електромеханічні замки - досить стійкі до злому (якщо замок міцний механічно), багато хто має механічний перевзвод (це означає, що якщо на

замок подали відкриваючий імпульс, він буде розблокований до тих пір, поки двері не відкриють).

Встановлюються на проходах / проїздах:

- Шлюзові кабіни —використовуються в банках, на режимних об'єктах (на підприємствах з підвищеними вимогами до безпеки).
- Ворота і шлагбауми - в основному, встановлюються на в'їздах на територію підприємства, на автомобільних парковках і автостоянках, на в'їздах на прибудинкову територію, у двори житлових будівель. Основна вимога - стійкість до кліматичних умов і можливість автоматизованого управління (за допомогою системи контролю доступу). Коли мова йде про організацію контролю доступу проїзду, до системи пред'являються додаткові вимоги - підвищена дальність зчитування міток, розпізнавання автомобільних номерів (у разі інтеграції з системою відеоспостереження).

Ідентифікатори

Основні типи виконання - карточка, мітка. Є базовим елементом системи контролю доступу, оскільки зберігає код, який служить для визначення прав («ідентифікації») власника. Це може бути Touch memory, безконтактна карта (наприклад, RFID -мітки), або застріваючий тип карт із магнітною смугою. В якості ідентифікатора може виступати так само код, що вводиться на клавіатурі, а також окремі біометричні ознаки людини - відбиток пальця, малюнок сітківки або райдужної оболонки ока, тривимірне зображення особи, малюнок капілярних ліній долоні.

Надійність (стійкість до злому) системи контролю доступу в значній мірі визначається типом використовуваного ідентифікатора: наприклад, найбільш поширені безконтактні карти proximity можуть підробляти в майстернях з виготовлення ключів на обладнанні, що є у вільному продажу. Тому для об'єктів, що вимагають більш високого рівня захисту, подібні ідентифікатори

не підходять. Принципово вищий рівень захищеності забезпечують RFID-мітки, в яких код карти зберігається в захищеній області та шифрується.

Контролери

Це ключовий елемент системи: саме контролер визначає, пропустити чи ні власника ідентифікатора через точку проходу, оскільки зберігає коди ідентифікаторів зі списком прав доступу кожного з них. Коли людина пред'являє (підносить до зчитувального пристрою) ідентифікатор, зчитаний з нього код порівнюється з зберігаються в базі, на підставі чого приймається рішення про відкриття точки проходу. Контролер для своєї роботи вимагає електроживлення, тому контролери, як правило, мають власний акумулятор, який підтримує його працездатність від декількох годин до декількох діб на випадок аварії електромережі.

Зчитувачі

Це пристрій, який отримує («зчитує») код ідентифікатора і передає його в контролер. Варіанти виконання зчитувача залежать від типу ідентифікатора: для «таблетки» - це два електричних контакта (у вигляді «лузи»), для proximity-карти - це електронна плата з антеною в корпусі, а для зчитування, наприклад, малюнка райдужної оболонки очі до складу зчитувача повинна входити телевізійна камера. Якщо зчитувач встановлюється на вулиці (ворота, зовнішні двері будівлі, проїзд на територію автостоянки), то він повинен витримувати кліматичні навантаження - перепади температур, опади - особливо, якщо мова йде про об'єкти в районах з суворими кліматичними умовами. А якщо існує загроза вандалізму, необхідна ще і механічна міцність (сталевий корпус). Окремо можна виділити зчитувачі для дальньої ідентифікації об'єктів (з відстанню ідентифікації до 50 м.). Такі системи зручні на автомобільних проїздах, парковках, на в'їздах на платні дороги і т.п. Ідентифікатори (мітки) для таких зчитувачів, як правило, активні (містять вбудовану батарейку).

Варіанти виконання пристроїв залежать від типу ідентифікатора. Так, для зчитування безконтактних карт обладнання виконується найчастіше з ABS пластику, стійкого до ультрафіолету, з електронною платою та антеною всередині корпусу. Обладнання, що монтується зовні приміщення, виконується з матеріалів, здатних витримувати температурні перепади та вплив опадів. При загрозі вандалізму зчитувачі мають міцний корпус, виконаний з нержавіючої сталі. Також є зчитувачі далекого радіусу дії, призначені для ідентифікації об'єктів на відстані до 12 метрів. Подібні пристрої зручні у використанні зі шлагбаумами для парковок та автопроїздів тощо.

Для біометричних ідентифікаторів (відбиток пальця, венозний малюнок, геометрія обличчя) потрібні біометричні зчитувачі. Біометричні технології засновані на «читанні» певних тілесних характеристик користувача. Дані перетворюються на цифровий код, який потім надходить у контролер або термінал, і базу даних програмного забезпечення.

Варто звернути увагу, що біометричний ідентифікатор, наприклад, відбиток пальця, зберігається в основі як унікальний цифровий код, з якого не можна відновити папілярний малюнок відбитка пальця. Ця технологія дозволить дотримуватися закону про захист персональних даних.

Використовуючи біометричні технології в системах контролю та управління доступом, ви отримуєте ряд переваг перед традиційними системами. Наприклад, RFID ключ може бути втрачений співробітником і ключем може скористатися зловмисник, щоб проникнути на об'єкт. Якщо на підприємстві запроваджено облік робочого часу і несумлінний співробітник вирішив попросити свого колегу відзначити його прихід на роботу за допомогою безконтактної картки, то у разі використання біометричних ідентифікаторів такі ситуації не можуть повторитись. Плюсом використання таких систем є швидкість додавання нового співробітника до бази даних та здешевлення адміністрування такої системи, оскільки немає необхідності відновлювати втрачені карти та заводити ключі для нових користувачів.

Переваги

- Гарна система для менедженту персоналу та їх доступів.
- Можливість гнучно налаштовувати систему як для персоналу так і для менеджерів які нею будуть керувати.

Недоліки

- Постійна технічна підтримка обладнання.
- Неможливість швидкої заміни всього обладнання.

Для таких систем також потрібні картки для кожного працівника, вони є тим ключем який дозволяє здійснювати прохід через деякий замок або турнікет, дана система має ряд недоліків, таких як неможливість швидкої заміни, та сама потреба в носінні цієї картки. Але є і багато переваг таких як:

- Висока надійність, внаслідок простоти пристрою; • Неможливість підробки пластикової карти, внаслідок відсутності інформації про склад провідників; • Висока стійкість пластикової карти до зовнішніх впливів: для того, щоб зіпсувати карту, необхідно ее зламати.
- Зчитувачі безконтактних карток (proximity card) дозволяють дистанційно ідентифікувати особу по персональному коду, записаного на картках.
- Процес зчитування забезпечує їх стійку роботу і зручність використання, високу пропускну здатність. Зчитувач генерує електромагнітне випромінювання певної частоти і, при внесенні карти в зону дії зчитувача, це випромінювання через вбудовану в карті антену живить чіп карти. Отримавши необхідну енергію для роботи, карта пересилає на зчитувач свій ідентифікаційний номер за допомогою електромагнітного імпульсу певної форми і частоти.
- Найвища пропускна здатність; зчитувачі можна досить ефективно захистити від вандалізму, встановивши, наприклад, їх усередині стіни, оскільки прямого контакту з ними не потрібно.

- Зчитувачі можна досить ефективно використовувати для контролю проїзду автотранспорту, розмістивши антену всередині полотна дороги, завдяки чому від водія не потрібно залишати салон для пред'явлення документів.

Кнопки виходу

Будь-яка система контролю доступу не може обійтися без простого, але важливого елемента, як кнопка виходу. Кнопки виходу використовуються для відкриття дверей, шлагбауму, воріт тощо. Типи кнопок виходу:

- Нормально відкриті (NO) – даний тип кнопок працює на замикання і найчастіше використовується з електромеханічними замками, клямками, контролерами.
- Нормально закриті (NC) – такі кнопки працюють на розмикання та використовуються під час роботи з магнітними замками.
- Універсальні (NC/NO) – це найбільш популярний тип кнопок, оскільки цей вид може працювати на замикання та розмикання. Цей тип підійде до будь-якої системи контролю доступу.

Важливим фактором є спосіб натискання кнопки. Найбільш звичним та поширеним є механічний спосіб натискання. У разі сенсорної кнопки необхідно лише злегка доторкнутися до провідної поверхні, щоб змінилося опір контролера і відбулося відкриття дверей. Безконтактна кнопка виходу має інфрачервоний датчик, який реагує на наближення руки. Плюсом цього рішення є гігієнічність: безконтактної кнопки з інфрачервоним сенсором не потрібно торкатися, досить просто змахнути перед пристроєм рукою – і двері відчиняться.

Ще одним із підтипів таких систем є електромеханічний замок. Електромеханічний замок нічим не відрізняється від звичайного замка. Але, звичайно ж, має свій невеликий секрет — захована електроніку в самому механізмі. Для такого замку необхідне підключення до мережі. Решта механізму роботи досить простий — комплексна робота електромагнітної котушки та пружини.

Залежно від конкретних цілей, а також технічних характеристик електромеханічні замки бувають різні. Почнемо з основної класифікації.

Електромеханічний замок. Принцип роботи цього виду пристрою полягає в механічному впливі на основний механізм. Важливою перевагою електромеханічного замку є його інтеграція з електроприводом. Це досить просто, що також дозволяє підключити до системи інші охоронні пристрої.

Електромагнітний замок. Це корпус, який складається з електромагніту та спеціальної планки. Планка має дуже високі показники магнітної проникності. Відповідно принцип роботи даного замку полягає в дії магніту: щоб двері були замкнені, магнітне поле взаємодіє з електрикою.

Електроригельний замок. Так само, як і будь-який інший вид, універсальний у своєму використанні. Його можна монтувати як у дерев'яні, так і в металопластикові двері, і навіть на скляні двері. Сам замок працює за допомогою ригельного механізму, звідси й назва. Ригель спонукає подачі електроживлення. Але, що не мало важливо, електроригельним замком можна керувати дистанційно з будь-якої точки світу. Тобто, якщо ви знаходитесь вдома, Ви можете відкрити офісні двері за допомогою спеціальної карти.

Електромеханічна засувка. Це досить універсальний пристрій. Вона відмінно підійде до будь-якого матеріалу дверей: чи то дерево чи металопластик. Засувка - це образно кажучи "відповідь". Монтується в дверний отвір за допомогою спеціального фіксатора для замку. Основна перевага електромеханічних засувок - це їхня простота. Вони досить легко монтуються, а потім ще легше використовуються. Електромеханічна клямка має одну функцію - утримувати.

Врізний електромеханічний замок. Найпоширеніший варіант - оскільки підходять для більшості дверей, і не треба виконувати додаткові пильні роботи. Як уже можна було припустити — серцевина врізного замку повністю вставляється в дверне полотно. Відкрити замок можна як за допомогою звичайного ключа, так і за допомогою картки.

Накладний електромеханічний замок. З накладним так само, як і з врізним, немає жодних складнощів. Монтується зверху на дверне полотно і вуаля – двері зачинені! Накладний замк має одну особливість в управлінні. Зовні двері можна відкрити двома способами: за допомогою ключа, вставленого у дверну свердловину або за допомогою блока живлення/контролера. Щоб вийти із замкненого приміщення, потрібно лише натиснути на спеціальну кнопку, яка вмонтована всередині.

Зараз, всі ці дії можно робити за допомогою телефонів, наразі майже кожна модель телефону підтримує технології які дозволяють проводити зчитування якоїсь інформації, та можуть бути використані замість пластикових карток. Такий спосід дозволить зменшити ціну за систему та її підтримку в майбутньому.

На цей день, схожі рішення які дозволяють робити деякий моніторинг це або прості турнікети які не мають ніяких панелей керування де можна буде отримати повну інформацію про працівників які через нього проходили або рішення які створені конкретно під потреби окремого підприємства або компанії. Тому дуже складно зробити повний аналіз та виокремити конкретні напрями для покращення. Найбільш правильним рішенням буде знайти той функціонал який є найбільш схожим в усіх цих системах та розвивати його надаючи можливість його розширення під конкретні потреби, що дозволить надати набільш гнучку екосистему для будь-якого підприємства.

Наразі немає ніяких аналогів таких універсальних систем, тому питанння розвитку цього напрямку є відкритим та не має ніяких загальних рішень які можно покращити, всі рішення будуються з нуля або заточені під конкретні потреби та є повним відображенням бачення розробників про те як все це повино бути влаштовано. Тому з цього всьго випливає велика необхідність у створенні та дослідженні принципів створення таких систем.

Принцип роботи систем відстежування геопозиції

Для реалізації функціоналу відстеження скористаємося системою GPS. Дана система підтримується майже будь-яким сучасним смартфоном, та дозволяє дуже точно показувати місцезнаходження.

GPS (англ. **Global Positioning System** — система глобального позиціонування, читається Джі Пі Ес, також ГПС (глобальна позиціонуюча система)) — супутникова система навігації, що забезпечує вимірювання відстані, часу та визначає місце розташування у всесвітній системі координат WGS 84. Дозволяє майже за будь-якої погоди визначати місце розташування будь-де на Землі (крім приполярні області) і навколоземного космічного простору. Система розроблена, реалізована та експлуатується Міністерством оборони США, при цьому в даний час доступна для використання в цивільних цілях – потрібен лише навігатор або інший апарат (наприклад, смартфон) із GPS-приймачем.

Основний принцип використання системи - визначення місця розташування шляхом вимірювання моментів часу прийому синхронізованого сигналу від навігаційних супутників антеною користувача. Для визначення тривимірних координат GPS-приймачу потрібно мати чотири рівняння: «відстань дорівнює добутку швидкості світла на різницю моментів прийому сигналу споживачем та моменту його синхронного випромінювання від супутників»:

$$|\mathbf{r} - \mathbf{a}_j| = c(t_j - \tau)$$

Рис. 2. 1. Приклад формули для визначення координат

GPS складається з трьох основних сегментів: космічного, керуючого та користувальницького. Супутники GPS транслюють сигнал із космосу, і всі приймачі GPS використовують цей сигнал для обчислення свого положення у просторі за трьома координатами в режимі реального часу.

Космічний сегмент складається з 32 супутників, що обертаються на середній орбіті Землі. Керуючий сегмент є головною керуючою станцією і декількома додатковими станціями, а також наземними антенами та станціями моніторингу, ресурси деяких із згаданих є спільними з іншими проектами. Користувальницький сегмент представлений приймачами GPS.

Незважаючи на те, що спочатку проект GPS був спрямований на військову мету, сьогодні GPS широко використовуються в цивільних цілях. GPS-приймачі продають у багатьох магазинах, що торгують електронікою, їх вбудовують у мобільні телефони, смартфони та наручні електроні годинники. Користувачам також пропонуються різні пристрої та програмні продукти, що дають змогу бачити своє місцезнаходження на електронній карті; мають можливість прокладати маршрути з урахуванням дорожніх знаків, дозволених поворотів і навіть пробок; шукати на карті конкретні будинки та вулиці, пам'ятки, кафе, лікарні, автозаправки та інші об'єкти інфраструктури.

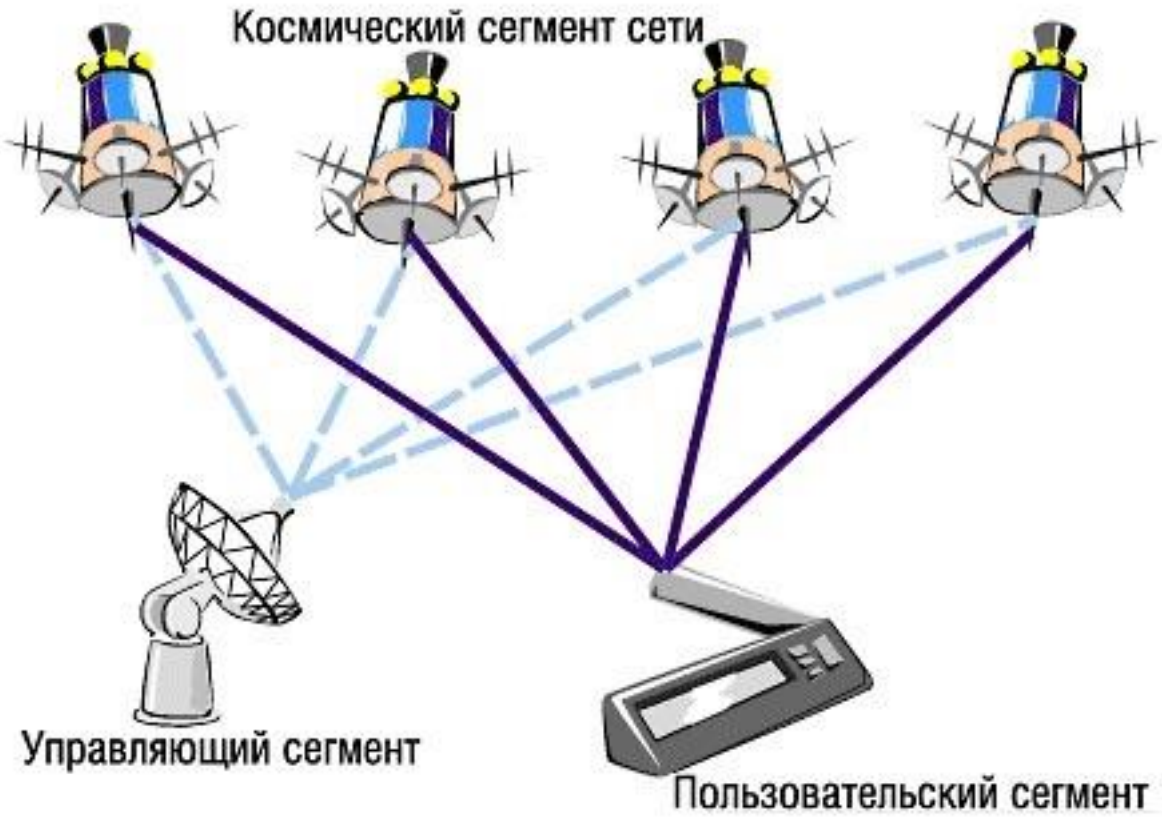


Рис. 2.2. Приклад системи GPS

Навігатори – один із найбільш затребуваних товарів на ринку гаджетів, за популярністю їх обганяють лише смартфони. Але і смартфони виробники вбудовують чіпи GPS, щоб пристрій міг виконувати функції навігатора. Однак тут користувача може підстерігати пастка, тому що в гонитві за прибутком виробники допускають навмисні або випадкові неточності в описі свого товару, дозволяючи покупцям переплутати технології GPS та AGPS.

У приймачів GPS є невеликі мінуси: вони працюють лише на вулиці, а через погану погоду можуть виникнути проблеми з прийомом сигналу від супутника, але ці недоліки вирішили за допомогою технології A-GPS (не плутати з AGPS). Суть у тому, що сигнал від приймача перенаправляють на сервер, на якому міститься вся інформація про становище супутників, тому

проблем із прийомом сигналу не існує. A-GPS використовують усі сучасні автомобільні навігатори.

Але існує також стільникова навігація AGPS – вона працює тільки в зоні покриття стільникової мережі та визначає місцезнаходження з точністю до 500 м. Вона менш точна в порівнянні з GPS, дає загальне уявлення про місце, де ви знаходитесь, зате пропонує супутникову карту околиць. Важливо, щоб було підключено послугу мобільного інтернету, а на рахунку залишалися гроші. Із сервісом AGPS працюють Google Maps. Найчастіше можливостей стільникової навігації достатньо, але її все одно не варто плутати з точною та безкоштовною системою GPS.

ПРИКЛАД РЕАЛІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ ВІДВІДУВАНОСТІ ПРАЦІВНИКІВ ПІДПРИЄМСТВА

Для початку ми повинні придумати структури бази даних яка буде використовуватися в системі:

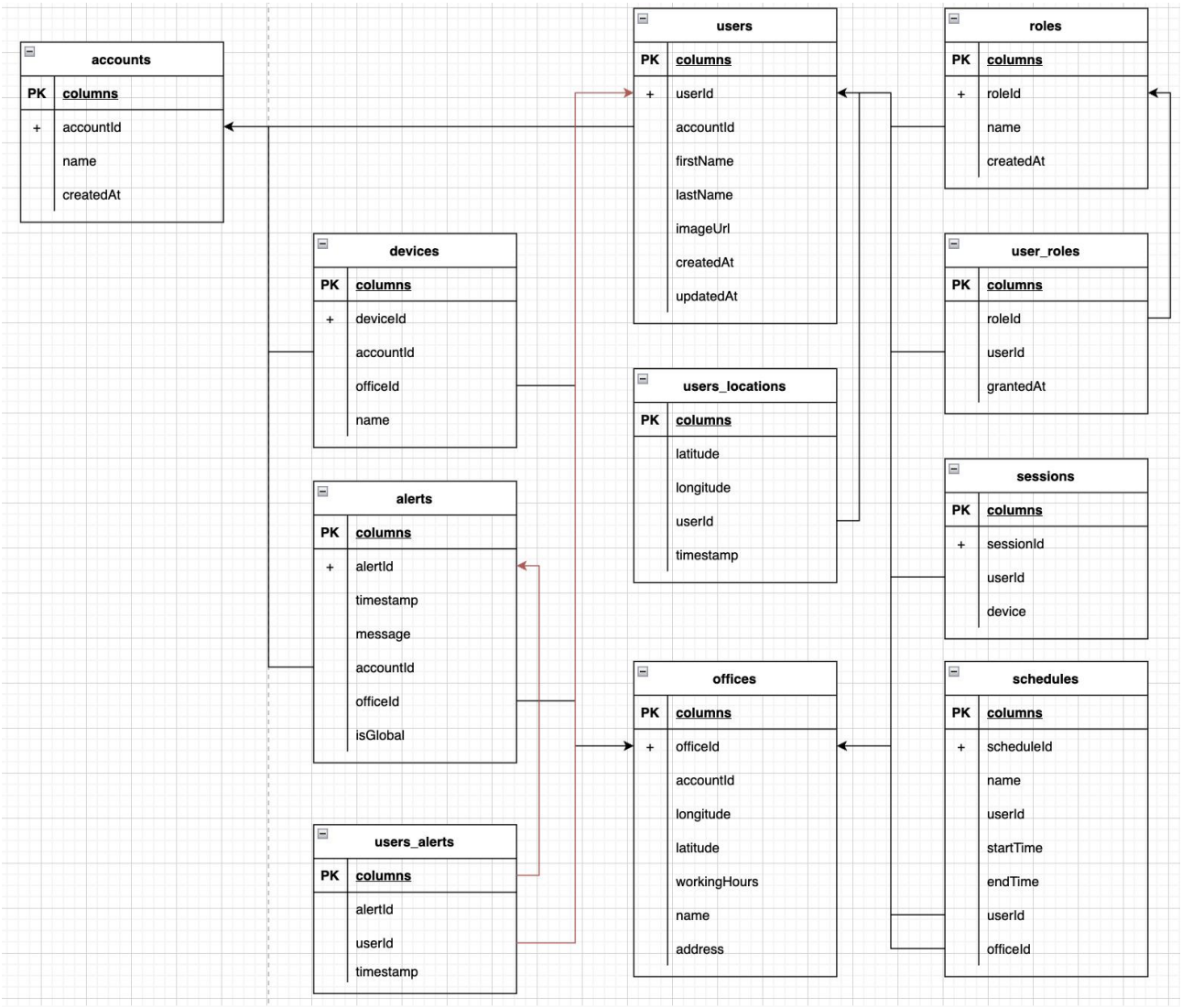


Рис. 3.1. Схема бази даних для системи моніторингу

Розберемо кожну таблицю:

accounts

Таблиця створена для збереження акаунтів клієнтів та зберігається в компанії яка цю систему буде підтримувати. Призначеня для обліку клієнтів та ведення бухгалтерії. Включає в себе такі поля як:

- **accountId** - унікальний ідетфикатор клієнта.
- **name** - назва клієнту
- **createdAt** - дата приєднання клієнта до системи

Дана таблиця включає в себе тільки три поля, в майбутньому, якщо буде потреба в розширенні або доданні нових таблиць пов’язаних з нею це буде зробити без складощів. Наприклад, таблиця з даними про використання серверних потужностей (якщо клієнт не має своїх серверів для системи) та звітів про оплату за послуги.

users

Таблиця призначена для збереження користувачів системи для конкретного підприємства. Може зберігатися як на серверах підприємства так і на серверах компанії яка цю систему створила. Включає в себе такі поля як:

- **userId** - унікальний ідентифікатор працівника чи користувача системи. Повинен бути завжди унікальним серед всіх користувачів системи серед всіх клієнтів якщо підприємство використовує таблицу “для всіх”. При використанні таблиці у базі даних самого підприємства таких обмежень немає.
- **accountId** - унікальний ідентифікатор акаунта клієнта серед всіх клієнтів які користуються цією системою.
- firstName - ім’я
- lastName - прізвище
- **imageUrl** - посилання на зображення для аватару користувача чи працівника. Зберігається на зовнішньому сховищі даних для файлів.
- **createdAt** - дата створення користувача у системі.

- **updatedAt** - дата останної зміни даних про користувача в системі.

Користувач є однією з найважливіших одиниць даних в базі даних, оскільки кожен запис ідентифікує якогось користувача чи працівника в системі та дозволяє на основі його створювати записи в інших таблицях які прив'язані до нього та створювати різні графіки та статистики.

roles

Таблиця створена для збереження ролей користувачів або працівників. Суть ролі полягає в тому що їх можна ставити для деяких користувачів та забороняти заходити в деякі приміщення, якщо роль користувача який хочу туди зайти не дозволяє це зробити. Також для менеджменту всередині системи, наприклад роль менеджера яка дозволяє бачити та редагувати дані про працівника в системі навідміну від ролі звичайного працівника який може тільки переглядати дозволені йому дані. Включає в себе такі поля як:

- **roleId** - унікальний ідентифкатор ролі в системи.
- **name** - назва ролі
- **createdAt** - дата створення ролі

Сама таблиця не несе в собі ніяк обмежень чи доступів, вона відповідає тільки за збереження цих ролей. Реалізацію доступів буде реалізована вже в коді самої системи.

user_roles

Таблиця створеня для збереження інформації про те які ролі були надані користувачу. Оскільки один користувач може мати багато ролей, наприклад (Працівник та менеджер), а одна роль може бути більше чим в одного користувача, то такий тип зв'язку повинен бути багато до багатьох. Включає в себе такі поля як:

- **userId** - ідентифікатор користувача якому присвоєна роль.
- **roleId** - ідентифікатор ролі.
- **grantedAt** - дата додавання ролі користувачу.

sessions

Таблиця зберігає в собі інформацію про поточні сесії користувача який увійшов у систему, оскільки система може бути відкрита як на телефоні так і на комп'ютері для одного користувача, тому ми повинні дозволяти мати записи про ці активні сесії в таблиці. Таблиця дозволяє моментально відключиті всі девайси користувача або відключити якийся один потрібний, наприклад, у разі загублення девайсу на якому користуває є авторизованим у системі для більшої безпеки, або після звільнення відразу закрити всі доступи до системи. Включає в себе такі поля як:

- **sessionId** - унікальний ідентифікатор поточної сессії.
- **userId** - ідентифікатор користувача.
- **deviceId** - ідентифікатор девайсу з якого користувач авторизувавсвя в системі.

Наразі були описані всі поточні таблиці які відносяться до менеджменнту користувачі або працівників. Таблиці можливо розширювати новими полями, або додавати зв'язані таблиці з іншою інформацією до них. Що є великим плюсом для сторонніх розробників які потрібно додавати якусь конкретну логіку специфічну для їхнього підприємства.

Розглянемо таблиці які потрібні для функціонування доступів на підприємстві.

offices

Таблиця представляє із себе дані про офіси підприємства в яких працюють працівники. Їхні координати та адресу для подальшої перевірки координат. Включає в себе такі поля як

- **officeId** - унікальний ідентифікатор офісу.
- **accountId** - ідентифікатор акаунту клієнта який використовує систему.
- **longitude** - довгота де знаходиться офіс чи підприємство.
- **latitude** - широта де знаходиться офіс чи підприємство.

- **workingHours** - часи роботи офіси (коли він відкритий на вхід та вихід).
- **name** - назва офісу.
- **address** - адреса офісу.

Тепер підприємство має інформації про її офіси та може додавати графіки роботи для працівників з прив’язкою до офісу. Також таблицю легко розширити новими полями якщо специфіка підприємства цього потребує.

Також якщо офіс має в собі якісь інши системи пропуску та вони можуть бути інтегровані до поточної системи то для цього існує таблиця під назвою **devices**, вона включає в себе такі поля:

- **deviceId** - унікальний ідентифікатор девайсу чи обладнання яке встановлене.
- **accountId** - ідентифікатор акаунті клієнта який використовує систему.
- **officeId** - ідентифікатор офісу в якому встановлений цей девайс.
- **name** - назва девайсу.

Також, якщо девайся дозволяють надавати якусь іншу інформацію, наприклад, статус роботи, то ці дані також можливо додати шляхом розширення таблиці, або створення нової таблиці зі зв’язком до **deviceId**.

Тепер маючи дані про офіси чи філії підприємства та працівників ми можемо формувати графікі для них. Для цього було створена таблиця під назвою schedules, яка включає в себе про графік роботи працівників. Вона включає в себе такі поля:

- **scheduleId** - унікальний ідентифкатор графіку.
- **userId** - ідентикатор користувача якому цей графік встановлено.
- **name** - назва графіку (опціонально).
- **startTime** - час початку роботи.
- **endTime** - час закінчення роботи.
- **officeId** - офіс (опціонально, якщо не вказати то працівник може працювати по цьому графіку в будь-якому офісі).

В даному прикладі ми бачимо що під кожного користувача створюється окремий запис із графіком. Також для реалізації таког зв’язку можно використовувати зв’язок багато-до-багатьох, але в даному випадкий зв’язок один до багатьох також підходить. У раз потреби цей функціонал може бути доданий, оскільки система гнучка для такого роду змін.

Після формування графіку ми можемо почати стеження за позиціє працівника під час робочого дня. Для цього була створена таблиця **users_locations**. Вона включає в себе такі поля:

- **longitude** - довгота.
- **latitude** - широта.
- **userId** - ідентифікатор користувача за яким був створений запис про місцезнаходження.
- **timestamp** - дата та час отримання координат.

Тепер ми можемо реалізовувати логіку яка буде час від часу перевіряти місцезнаходження працівника. Також система дозволяє налаштовувати як буде починатися стеження, чи після того як працівник пройшов турнікет та система записала його прихід, чи після того як користувач в додатку на телефоні натиснув кнопку про те що він прийшов на роботу.

Також ми можемо розширяти таблицю іншими метриками, наприклад якщо кожне приміщення має свій Wi-Fi та користувач до нього підключений то ми можемо знати з точністю до кімнати де зараз він знаходиться, для цього потрібно буде створити таблицю з кімнатами які є на підприємстві чи офісі та записувати ці дані. Також можемо міксувати це з координатами для максимальної точності.

На основі цих даних ми вже можемо створювати звіти про працівників, та їх знаходження на робочому місці. Строїти за допомогою штучночного інтелекту деякі передбачення про поведінку на роботі та інше.

У часи пандемії та для велики підприємств де працює велика кілкість людей дуже важливим постає питання дотримання всіх мір захисту проти поширення вірусу, такими методами є дотримання дистанції між людьми та

заборона на зібрання великих кількостей людей в закритих або маленьких приміщеннях, де ризики зараження стають дуже високими. Оскільки система дозволяє бачити місцезнаходження працівника по координатах ми можемо створити деяку систему яка буде слідкувати за кількістю працівників в одному місці та дистанцією між ними в реальному часі та надсилати попередження про ризики для здоров'я. Для цього ми повинні створити ще дві таблиці, перша буде називатися **alerts**, вона буде включати в себе повідомлення різного роду які будуть надсилатися працівникам. Вона включає в себе такі поля:

- **alertId** - ідентифікатор попередження.
- **timestamp** - час в який була чи буде створена відправка.
- **message** - текстове повідомлення яке отримають працівники.
- **accountId** - ідентифікатор акаунту клієта який користується системою.
- **officeId** - ідентифікатор офісу в якому буде створена розсилка.
- **isGlobal** - булеве значення, якщо воно true то розсилка буде здійснена для всіх працівників офісу, якщо офіс не був указаний то для всіх працівників підприємства.

Тепер система буде слідкувати за координатами людей, та у разі скупчення або недотримання дистанції буде надсилати повідомлення, але постає проблема з тим що повідомлення будуть відправлятися всім працівникам у офісі, тому ми можемо покращити додавши таблицю, яка буде відповідати за те які повідомлення відправляти яким працівникам, тоді у нас буде можливість відправки конкретним працівникам які порушили правила дистанції чи скупчення. Створена таблиця буде називатися **users_alerts**, та буде включати в себе такі поля:

- **alertId** - ідентифікатор повідомлення яке буде відправлене.
- **userId** - ідентифікатор працівника який отримає повідомлення.
- **timestamp** - час в який буде відправлено чи було відправлено.

Оскільки система геолокації не надає постійно точні координати, то для більшої надійності система буде слідкувати деякий час за кординатами, та якщо вони не змінюються протягом тривалого часи (який також можна налаштувати

в панелі керування) і після цього тільки надсилати повідомлення. Якщо система вирішить що даних недостатньо вона зробить запис про можливий ризик, який працівник зможе перевірити в системі.

Ми можемо розширювати даний функціонал додавши таблицю з хворими які зараз хворіють та на основі їхніх переміщень та контакту з іншими особами надсилати попередження про можливі ризики зараження та поперджувати розповсюдження хвороби у разі її стрімкого поширення між працівниками.

Розробку візуального інтерфейсу частини системи для працівника та менеджера розробимо за допомогою мові розмітки HTML та CSS, логіка клієнтської частини буде імплементована за допомогою JavaScript. Серверна частина буде створена також на мові JavaScript, яка буде запускатися в середовищі Node.js.

Для збереження даних будемо використовувати найпростішу базу даних, у форматі key-value. База даних “Ключ-значення“ це парадигма зберігання даних, призначена для зберігання, вилучення та управління асоціативними масивами, структура даних, більш відома сьогодні як словник або хеш-таблиця. Словники містять колекцію об'єктів чи записів, які, своєю чергою, містять безліч різних полів, кожне містить дані. Ці записи зберігаються та вилучаються з використанням ключа, який однозначно ідентифікує запис та використовується для швидкого пошуку даних у базі даних. Бази даних «ключ-значення» працюють зовсім інакше, ніж відомі реляційні бази даних (РБД). У РБД попередньо визначають структуру даних у базі даних як послідовність таблиць, що містять поля з чітко визначеними типами даних. Експонування типів даних у базі даних дозволяє застосувати низку оптимізації. Навпаки, системи "ключ-значення" обробляють дані як одну непрозору колекцію, яка може мати різні поля для кожного запису. Це забезпечує значну гнучкість і більш точно слідує сучасним концепціям, таким як об'єктно-орієнтоване програмування. Оскільки необов'язкові значення не представлені заповнювачами або вхідними параметрами, як у більшості РБД, бази даних «ключ-значення» часто використовують набагато менше пам'яті для зберігання

однієї й тієї ж бази даних, що може призвести до значного збільшення продуктивності за певних робочих навантажень.

Оскільки така база даних не підтримує можливість зв’язування даних, як це можливо в реляційних база даних, тому таку логіку ми повинні імплементувати самі. Зокрема, вже існують готові рішення, які дозволять спростити написання логіки. Одним із таких інструментів є GraphQL.

GraphQL — це мова запитів та маніпуляції даними з відкритим кодом для API та среда виконання для обслуговування запитів із наявних даних. GraphQL розробила компанія Facebook (наразі Meta) у 2012, а публічний реліз відбувся 2015 року. 7 листопада 2018, GraphQL було переведено від Facebook до новоутвореної GraphQL Foundation, яку прихистила неприбуткова Linux Foundation. GraphQL надає підхід розробки веб API та його можна порівнювати та протиставляти REST та іншим архітектурам веб-сервісів. Він дозволяє клієнтам визначати структуру потрібних даних і таку саму структуру повертає сервер, таким чином запобігаючи передачі надлишкових даних, але це впливає на дієвість веб-кешування результатів запитів. Гнучкість і багатість мови запитів, що може бути не потрібна для простих API. Він складається з системи типів, мови запитів та семантики виконання, статичної валідації та інтроспекції.

Ми можемо описати структуру нашої бази даних використовуючи спеціальний синтаксис GraphQL для написання схем БД. Приклад схеми для сутності працівника:

```
interface Entity {
      id: ID!
      createdAt: Int!
      updatedAt: Int
}

type User implements Entity {
      id: ID!
      login: String!
      password: String!
      firstName: String
```

```
        lastName: String
        imageUrl: String
        accountId: ID!
        createdAt: Int!
        updatedAt: Int

        loginSessions: [LoginSession]!
}
```

Спочатку створимо інтерфейс під назвою *Entity*, він буде слугувати інтерфейсом для будь-якої сутності, та включає в себе поле id, яке повинне бути реалізоване в кожній сутності, воно буде слугувати як *primary key* як реляційних базах даних. По ньому ми зможемо ідентифікувати будь-яку сутність всередині таблиці.

Далі створюємо схему для працівника, в нашому випадку сутність буде називатися *User*, оскільки це буде загальне поняття в базі даних, яке в перспективі буде відображати не тільки працівника підприємства а й просто користувача системи який може бути не зв'язаним з підприємством. Сутність має такі поля як і схеми бази даних для таблиці **users**. В таблиці ми бачимо що назва поля для primary key називається як *userId*, а в нашій схемі як *id*, але це не є проблемою, оскільки на етапі реалізації логіки вибору з бази даних ми можемо змінювати назви полів. Поле accountId, буде слугувати посиланням на ідентифікатор акаунту який буде створений в таблиці accounts, тому потрібно створити схему для цієї таблиці:

```
type Account implements Entity {
        id: ID!
        name: String
        createdAt: Int!
        updatedAt: Int!
}
```

Як бачимо дана сутність також повинна імплементувати інтерфейс Entity, оскільки також повинна мати унікальний ідентифікатор. Тепер ми можемо

указувати ідентифікатор аккаунта для користувача в якому він був створений для позначення зв’язку.

Далі подивившись на схему сутності користувача, ми бачимо поле під назвою loginSessions якого немає в таблиці. Це поле ніяким чином не буде зберігатися в таблиці users, оскільки є повністю створеним полемо GraphQL, воно створенне для відображення поточний сесій входу корисстувача в системі та є деяким посилання на таблицу під назвою sessions, для цього нам знову потрібно створити схему для сутності сесії:

```
type LoginSession implements Entity {
      id: ID!
      userAgent: String!
      userId: ID!
}
```

Тепер ми маємо сутність яка відображає поточні сесії. Як бачимо її назва LoginSession, тепер ми можемо використовувати її як тип (схоже з ООП), та вказувати посилання на неї в схемах для інших сутностей. Указавши цей тип для поля, під час запиту, ми отримаємо пов’язану інформацію з таблиці sessions для користувача для якого був зроблений запит. Отже, ми отримали аналог функції JOIN в реляційних базах даних. Великим плюсом є те, що дани приходять в такому форматі в якому був створений запит.

Для того щоб сформувати запит ми повинні описати в схемі сутності які запити можно робити для неї та в якому форматі будуть отримані відповіді. Для цього GraphQL дозволяє створювати схеми для запитів, одна сутність може мати багато схем для запиту, створимо таку схему для сутності користувача:

```
type Query {
  User(id: ID!): User
  Users(params: FilterParams): [User]!
}
```

Для цього потрібно сформувати назву запиту, в даному прикладі маємо два запити на отримання одного користувача по його ідентифікатору, та отриманні колекції користувачів вказавши параметри для фільтрації які описані в інтерфейсі *FilterParams*:

```
input FilterParams {
      ids: [ID!]
      limit: Int
      offset: Int
      sort: SORT
      q: String
}
```

В ньому описані поля за якими буде йти пошук або фільрація. Вони є аналогом умови WHERE в реляційних базах даних. Результатом запиту ми отримаємо або об’єкт зі знайденим користувачем, або для отримання колекції масив з користувачами. Якщо нічого не буде знайдено ми отримаємо або нічого або пустий масив відповідно.

Для того щоб ми змогли запустити запит ми повинні налаштувати логіку по якій ці дані будуть витягуватися з бази даних, для цього нам потрібно реалізувати це, GraphQL дозволяє описувати це в окремому файлі. Приклад коду:

```
User: {
      async loginSessions(_, __, context) {
            return await UserAPIAdapter.getUserSessions({ authorization:
            context.req.headers.authorization });
      }
},

Query: {
      async User(_, args, context) {
            const { id } = args;

            return await UserAPIAdapter.getUserById(id, { authorization:
            context.req.headers.authorization });
      },
```

```
        Users(_, args) {
            // …
        }
},
```

Приклад того як проходять запити описано на малюнках нижче.

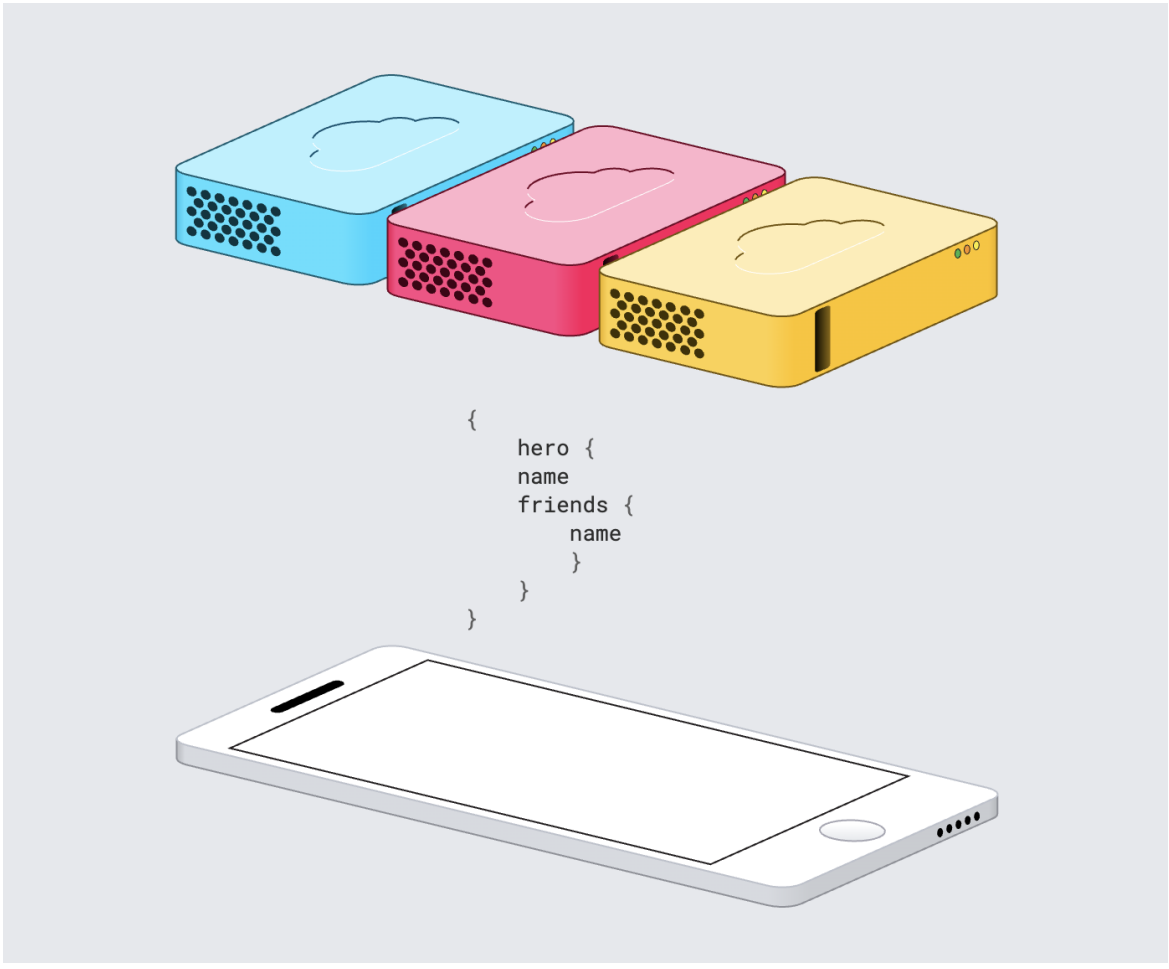


Рис. 3.2. Приклад запиту клієнта на сервер

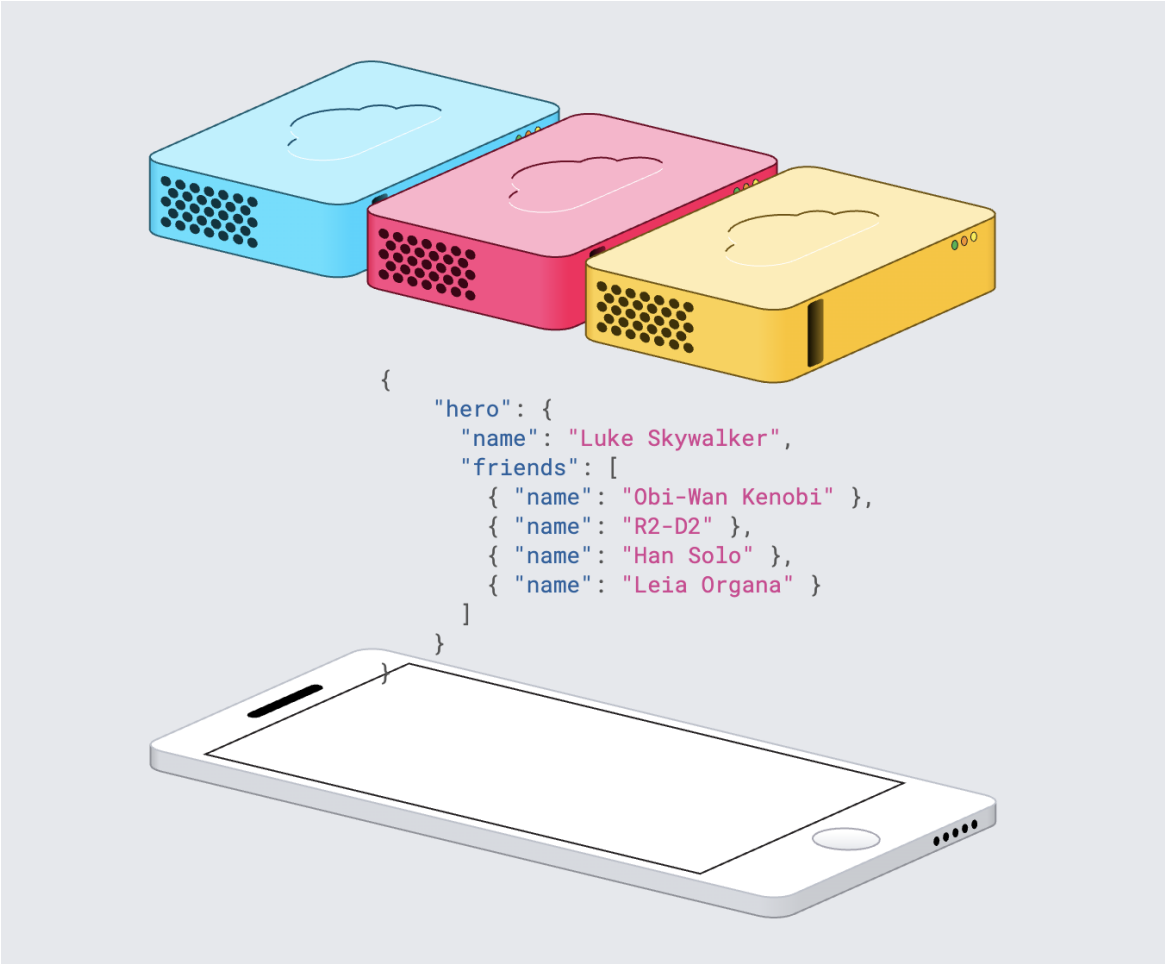


Рис. 3.3. Приклад відповіді до запиту

Приклад запиту для одержання користувача та поточних сесії буде виглядати так:

```
{
	Users {
		id
		firstName
		lastName
		accountId
		imageUrl

		loginSessions {
			userAgent

		}
	}
}
```

Отриманий результат буде в такому форматі:

```
{
      Users: [
            {
                  id: "dfa6d952-cf2e-4ca9-a4f3-0b185016e30a",
                  firstName: "Ivan",
                  lastName: "Mokhonko",
                  accountId: "1801d53f-76e2-4cff-a3a6-5d793c038595",
                  imageUrl: "",
                  loginSessions: [
                        {
                              userAgent:    "Mozilla/5.0    (Macintosh;
                              Intel     Mac     OS     X     10_15_7)
                              AppleWebKit/537.36  (KHTML,  like  Gecko)
                              Chrome/97.0.4692.99 Safari/537.36"
                        },
                        {
                              userAgent:  "Mozilla/5.0  (iPhone;  CPU
                              iPhone  OS  10_3_1  like  Mac  OS  X)
                              AppleWebKit/603.1.30 (KHTML, like Gecko)
                              Version/10.0 Mobile/14E304 Safari/602.1"
                        }

                  ]
            },
            {
                  id: "e0d331a9-71ff-4451-b751-1907f1f51fb8",
                  firstName: "Serhii",
                  lastName: "Mokhonko",
                  accountId: "1801d53f-76e2-4cff-a3a6-5d793c038595",
                  imageUrl: "",
                  loginSessions: [
                        {
                              userAgent: "Mozilla/5.0 (iPad; CPU OS
                              13_3 like Mac OS X) AppleWebKit/605.1.15
                              (KHTML,  like  Gecko)  CriOS/87.0.4280.77
                              Mobile/15E148 Safari/604.1'"
```

```
                        }
                    ]
                }

            ]
        }
```

Переглянувши запит, ми бачимо що отримали інформацію про користувачів та їхні поточні сесії, для користувача Ivan Mokhonko, існує дві сесії, одна з ноутбука, інша з телефона. Створивши запит на видалення сесії для телефону ми зможемо інвалідувати доступ, у разі потреби, з цього девайсу та виконати вихід користувача з системи, що дозволить підвищити безпеку.

Для модифікації даних також GraphQL надає можливість створювати схеми для запитів. Приклад схеми для запитів:

```
type Mutation {
  SignIn(login: String!, password: String!, userAgent: String): TokensPair
  SignOut: Boolean

  VerifyToken(token: String!): Boolean
  RefreshToken(refreshToken: String!): TokensPair

  CreateUser(
    login: String!,
    password: String!,
    firstName: String,
    lastName: String,
    avatarUrl: String,
    accountId: ID!,
  ): User
  UpdateUser(
    id: ID!,
    newLogin: String,
    newfirstName: String,
    newLastName: String,
    newAvatarUrl: String
  ): User
  DeactivateUser(userId: ID!, accountId: ID!): User
```

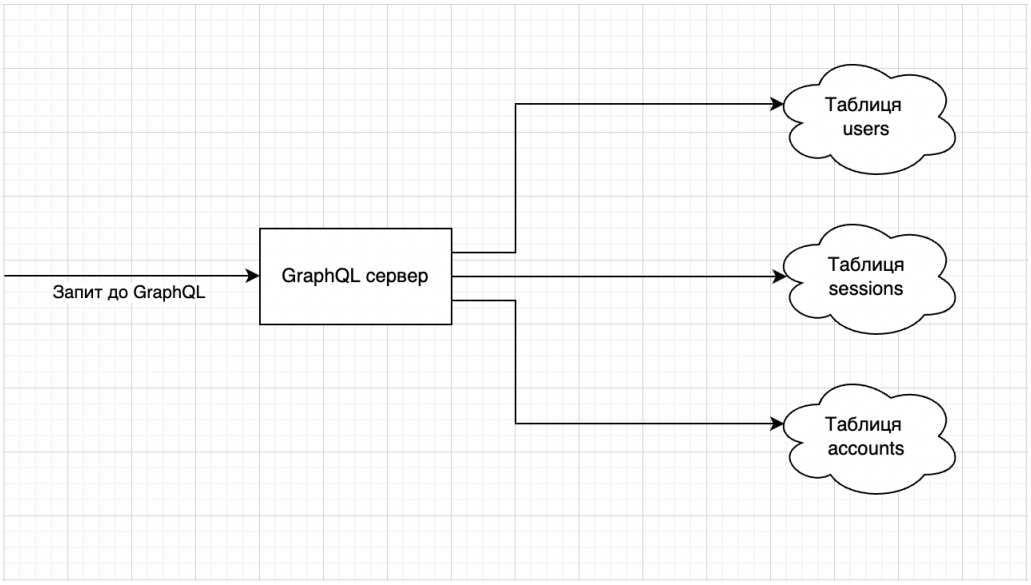
```
  DeleteUser(id: ID!): Boolean
}
```

Кожен запит також має поля які він отримує та запише до бази даних та фомат відповіді на запит. Також ми можемо створювати запити які не тільки виконують операції над поточною сутністю, а й деякі з нею пов’язні, наприклад додавання запишу до таблицій поточних сесій користувача при авторизації в системі. Реалізації логіки запитів на зміну:

```
Mutation: {
    async CreateUser(_, args) {
      const {
        login,
        password,
        firstName,
        lastName,
        avatarUrl,
        accountId
      } = args;

      return await UserAPIAdapter.createUser({
        login,
        password,
        firstName,
        lastName,
        avatarUrl,
        accountId
      });
    },

    async UpdateUser(_, args) {
      // …
    },

    async DeactivateUser(_, args) {
      // …
    },

    async DeleteUser(_, args) {
      const { id } = args;
```

```
      return await UserAPIAdapter.deleteUser(id);
    },

    async SignIn(_, args) {
      const { login, password, userAgent } = args;

      return await UserAPIAdapter.signIn(login, password, userAgent);
    },

    async VerifyToken(_, args) {
      const { token } = args;

      return await UserAPIAdapter.verifyToken(token);
    },

    async RefreshToken(_, args) {
      const { refreshToken } = args;

      return await UserAPIAdapter.refreshToken(refreshToken);
    },

    async SignOut(_, __, { req }) {
      return await UserAPIAdapter.signOut({ authorization:
req.headers.authorization });
    },
  }
```

В цьому прикладі кожен метод який був описаний в схемі створений на мові JavaScript та запущений в середовищі Node.js, який виконується на



сервері.

Рис. 3.4. Візуалізація запиту

Плюсом такого підходу є те, що ми можемо швидко та точечно внести зміни до реалізації методів запитів, наприклад замінивши одну таблицю на іншу, можливо навіть яка знаходиться на іншому сервері та в іншій базі даних, оскільки таблиці ніяким чином не зв’язані між собою на рівні бази даних, всі зв’язки реалізуються логікою в коді. Приклад коду який робить запити до бази даних:

```
const axios = require('axios');
const _ = require('lodash');

const API_URL = 'https://em.staging.api.onereach.ai/http/668917f5-beec-
4534-a7b4-b59ea7cca124/users/auth';
const USERS_API = 'https://em.staging.api.onereach.ai/http/668917f5-beec-
4534-a7b4-b59ea7cca124/users';
const USERS_SESSIONS_API =
'https://em.staging.api.onereach.ai/http/668917f5-beec-4534-a7b4-
b59ea7cca124/users/sessions';

module.exports = {
  async getUserById(userId, { authorization }) {
```

```
  const response =  await axios.get(
    USERS_API,
    {
      headers: { authorization },
      params: { userId }
    }
  );

  return response.data;
},

async createUser(payload) {
  const response =  await axios.get(API_URL, payload);

  return response.data;
},

async deleteUser(id) {
  const response =  await axios.delete(API_URL, { params: { id } });

  return response.data;
},

async signIn(login, password, userAgent) {
  const response =  await axios.post(API_URL, {
    type: 'authenticate_credentials',
    login,
    password,
    userAgent
  });

  return response.data;
},

async signOut({ authorization }) {
  const token = authorization.split(' ')[1];

  const response =  await axios.post(API_URL, {
    type: 'invalidate_session',
    token,
  });

  return response.data;
```

```
  },

  async verifyToken(token) {
    const response =  await axios.post(API_URL, {
      type: 'validate_token',
      token,
    });

    return _.get(response, 'data.success') || false;
  },

  async refreshToken(refreshToken) {
    const response =  await axios.post(API_URL, {
      type: 'refresh_token',
      refreshToken
    });

    return response.data;
  },

  async getUserSessions({ authorization }) {
    const response =  await axios.get(
      USERS_SESSIONS_API,
      { headers: { authorization } }
    );

    return response.data;
  }
}
```

Важливу роль в системі також грає безпека даних користувачів та контроль доступів під час запитів на окремі ресурси. Для цього використовується технологія JWT, для формування ключів доступів які будуть видаватися кожному користувачу та ідентифікувати його в системі.

JSON Web Token (JWT) - це відкритий стандарт (RFC 7519) для створення токенів доступу, що базується на форматі JSON. Як правило, використовується для передачі даних для автентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і

передаються клієнту, який надалі використовує цей токен для підтвердження своєї особи.

Токен JWT складається з трьох частин: заголовка (header), даних (payload) та підпису або даних шифрування. Перші два елементи – це JSON об'єкти певної структури. Третій елемент обчислюється на основі перших і залежить від обраного алгоритму (у разі використання непідписаного JWT може бути опущений). Токени можуть бути перекодовані в компактне уявлення (JWS/JWE Compact Serialization): до заголовка та корисного навантаження застосовується алгоритм кодування Base64-URL, після чого додається підпис і всі три елементи розділяються точками («.»).

Для цього створимо деяку точку куди ми можемо дати цей токен та у відповідь отримати відповідь чи дозволено цьому користувачи робити якісь дії чи просто запит в системі чи ні. Приклад коду для перевірки валідності токена який надав користувач при запиті:

```
const jwt = require('jsonwebtoken');

const TOKEN_SECRET = await this.mergeFields['config'].get({path:
'TOKEN_SECRET'});

const token = await this.mergeFields['waitForRequest'].get({path:
'request.body.token'})

const decoded = jwt.verify(token, TOKEN_SECRET);

const userId = _.get(decoded, 'userId');

if(!userId)
  throw new Error('Invalid token');

return decoded;
```

При виконанні даного коду буде повернуте булеве значення того чи дозволено користувачу робити запит. Дану логіку ми можемо винести в окремий метод на сервері та запускати перед кожним запитом. Перевагою цього методу також є те, що перевірка на валідність не потребує запиту до бази даних, оскільки ключи за якими йде перевірка знаходяться прямо в коді.

На основі цього також реалізуємо інші сутності, запити до них та реалізуємо логіку в коді. Після цього отримуємо працюючу систему систему.

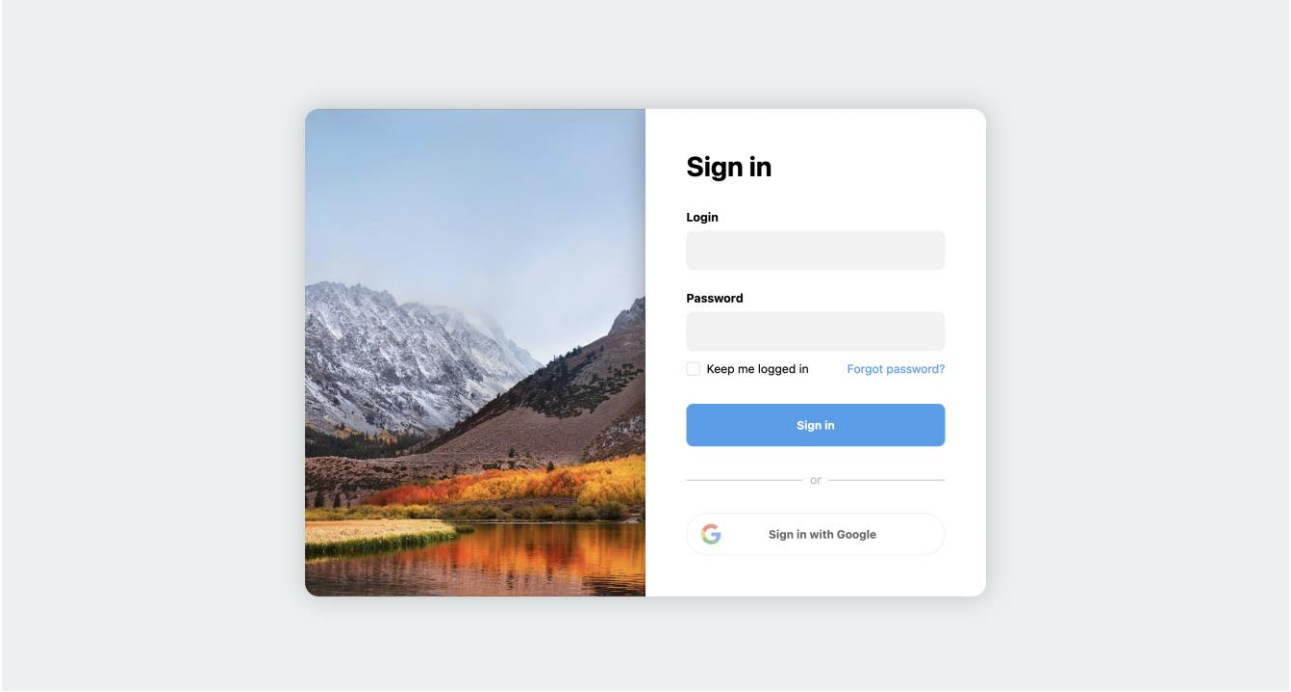
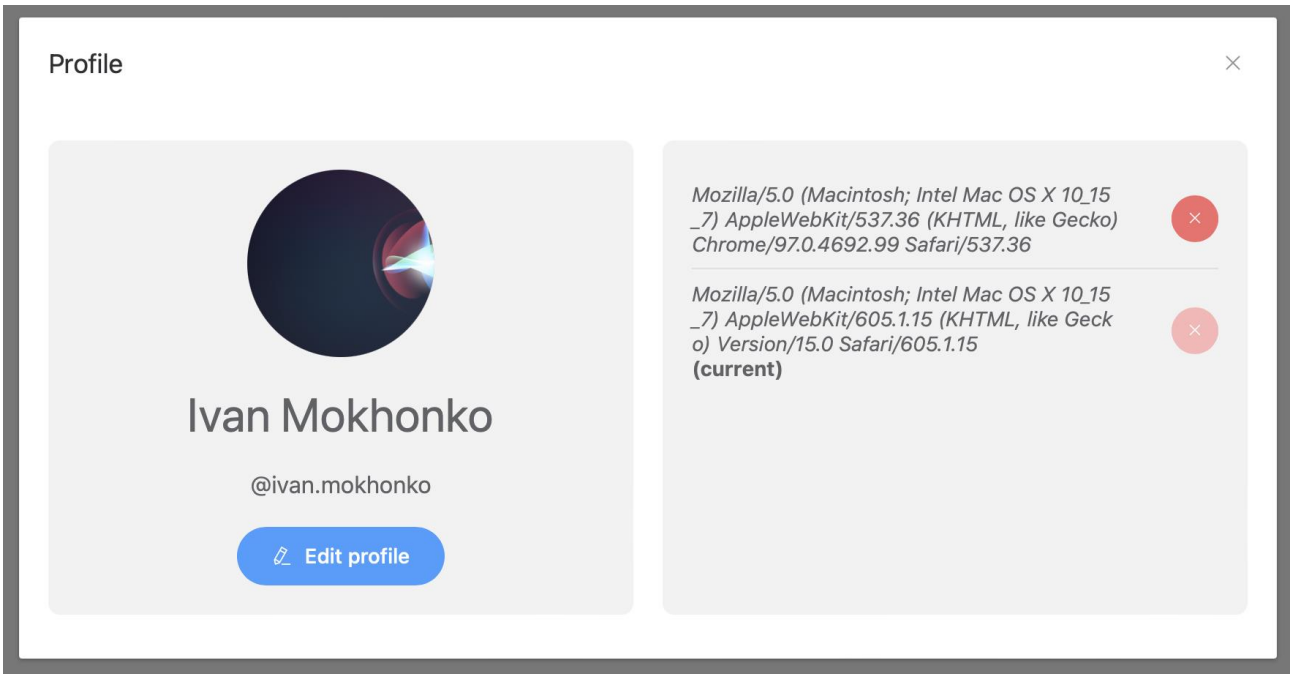


Рис. 3.5. Приклад інтерфейсу для входу в систему.

Для входу користувачу потрібно ввести його логін та пароль, який був сформований для нього. Під час першого входу в його акаунт, користувачу буде запропоновано змінити пароль. Також система підтримує авторизацію через електронну пошту від компанії Google.

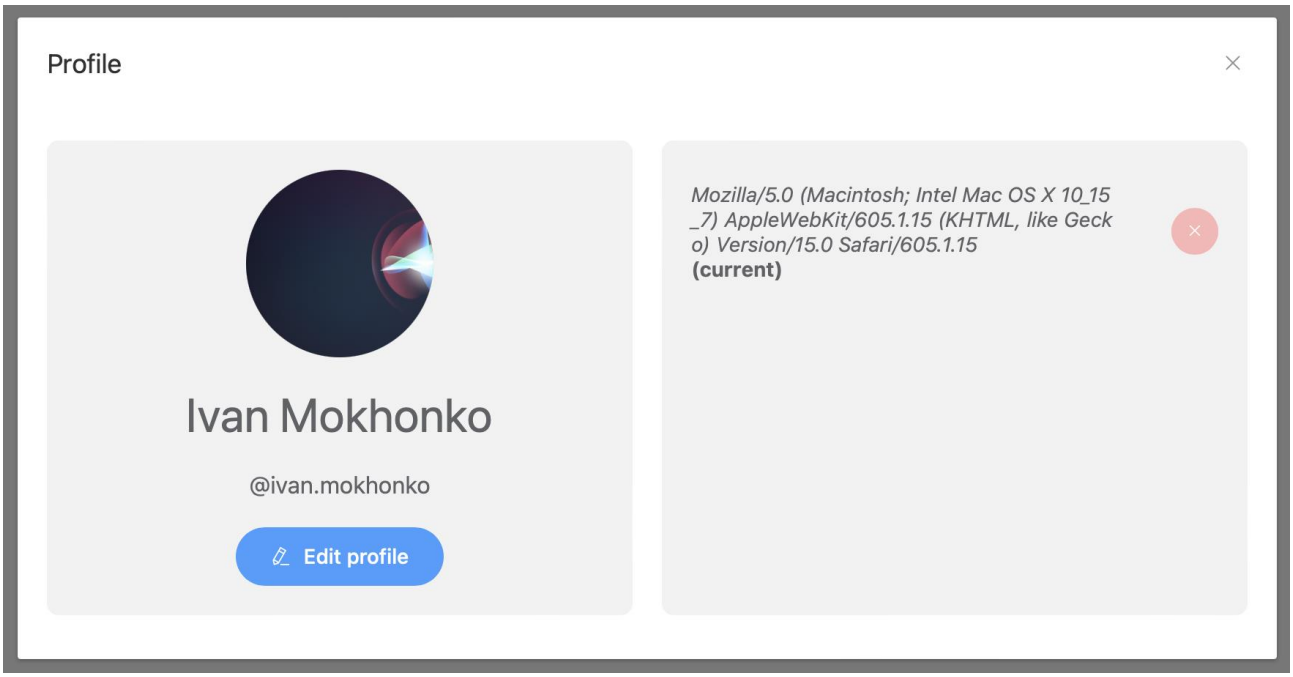
Проведемо вхід користувача з двух браузерів, щоб отримати дві сесії та



відприємо профіль щоб побачити їх.

Рис. 3.6. Приклад профілю з поточними сесіями

Якщо нам потрібно видалити сесію, натискаємо на кнопку видалити

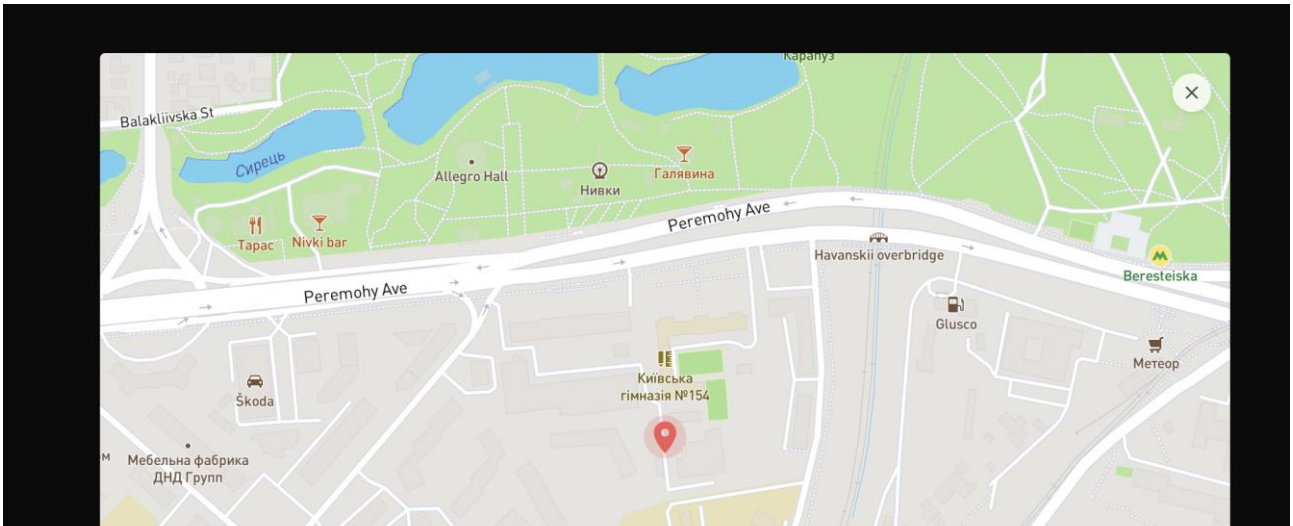


справа від сесії та бачимо результат:

Рис. 3.7. Приклад видаленої сесії

Поточну сесію користувач може видалити тільки натиснувши кнопку виходу з акаунту. Якщо користувач забув дані для входу, і потрібно вийти з загубленого девайсу, дану процедуру може виконати адміністратор системи, або користувач які має права на видалення сесій інших користувачів.

Для реалізації можемо скористуватися додатком системи, або підключити свою реалізацію, для цього використаємо якийся мессенджер в якому можливо отримати геопозицію.



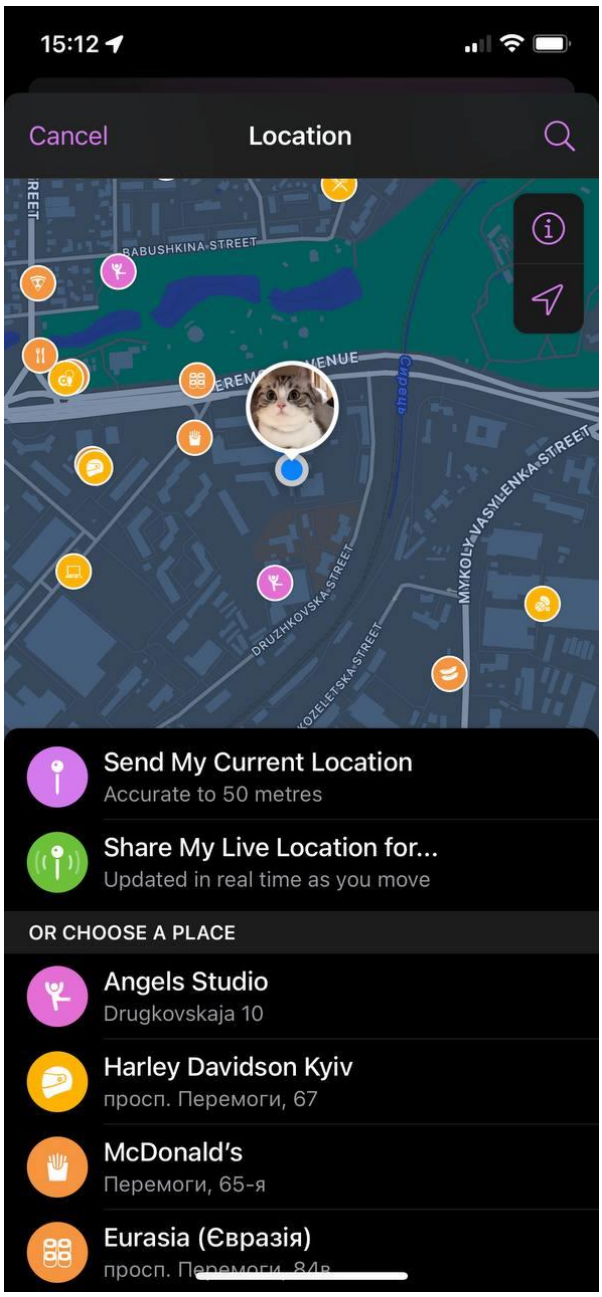


Рис. 3.8. Приклад отримання геопозиції в мессенджері Rich Web Chat

Також, створимо бота в мессенджері Telegram, плюсом є те, що є можливість стеження за геопозицією протягом деякого часу.

Рис. 3.9. Приклад інтерфейсу для дозволу на перегляд відстежуваної геолокації

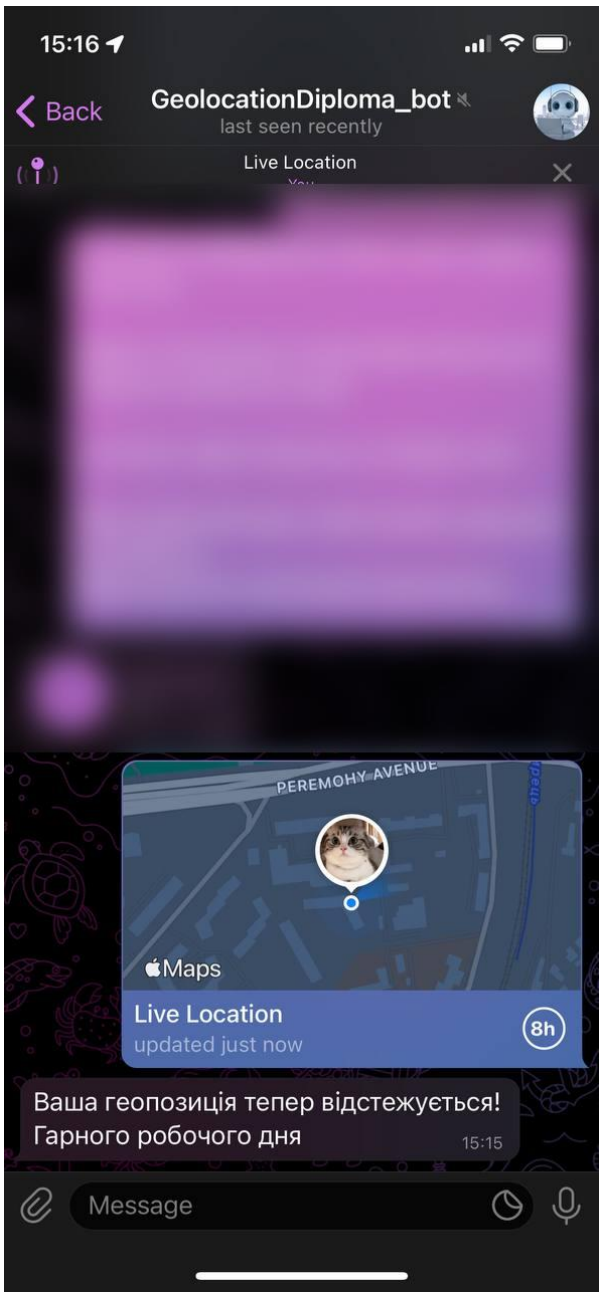


Рис. 3.10. Приклад геолокації та відповіді бота про те що він почав відклідковувати її

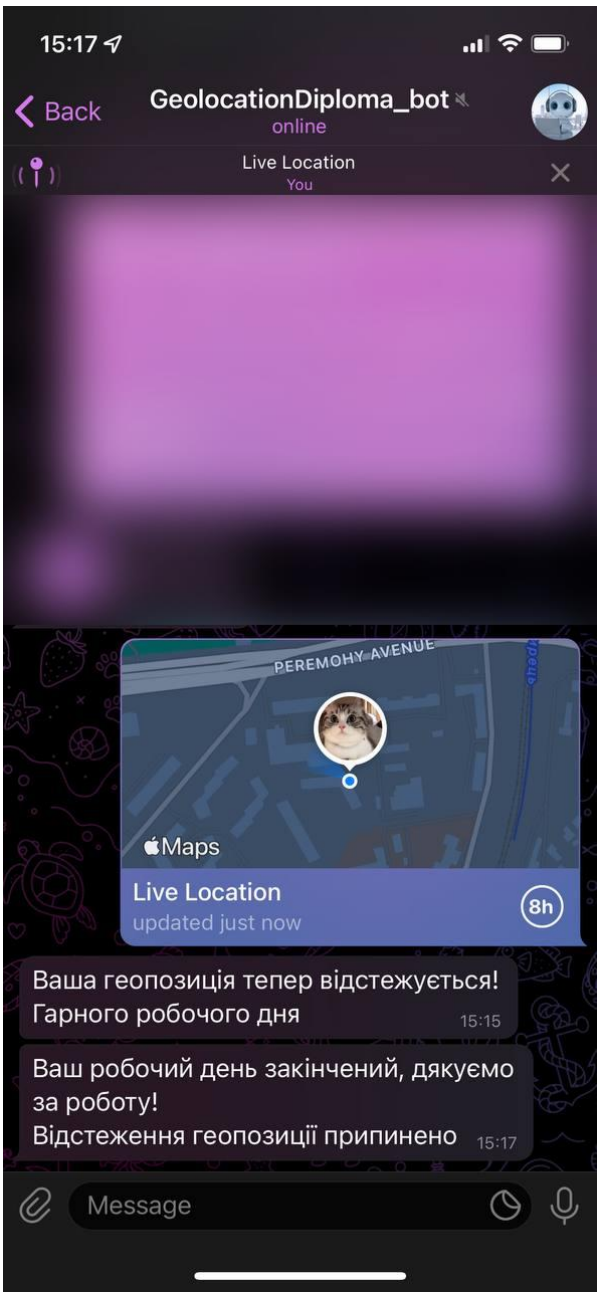


Рис. 3.11. Приклад відповіді бота після того як робочий час закінчився.

Як бачимо ми можемо інтегрувати цю логіку систему в будь-яке місце де це можливо, що надає великі можливості по розширенню та покращенню функціоналу

ВИСНОВКИ

В даній магістерській роботі проаналізовано основні проблеми та недоліки існуючих систем для контролю доступу та моніторингу відвідування підприємства працівниками. Було проведено порівняльні аналізи рішень які існують на ринку та виявлено багато їх переваг та недоліків. В результаті аналізу було змодельовано інформаційну систему для моніторингу відвідуваності працівників підприємства яка надає додаткові покращення до існуючих рішень, або повністю дублює їх якщо вони не були впроваджені та дозволяє отримувати докладну статистику по переміщенню працівників.

Основною ідеєю є концепція створення системи яка дозволяє вирішувати поставлені проблемі в цій області за допомогою менших витрат на обладнання та підтримку, які наразі існують на ринку. Описана ідея може стати основою розвитку для майбутніх рішень моніторингу та контролю доступу та підвищенням безпеки праці загалом.

Система яка була створена може бути використана майже на будь-якому підприємстві яке в тій чи іншій мірі постає перед проблемою моніторингу працівників на підприємстві та може бути легко впроваджена як для малого бізнесу так і для великого. Концепція легкого розширення має великий потенціал та дозволить налаштовувати будь які рішення під конкретну специфіку. Розширення функціоналу дозволяє не тільки покращувати проблеми моніторингу, а й такі як поширення хвороб, завдяки даним які вона надає.

Під час створення системи були використані сучасні технології та методики створення системних архітектур. Була продемонстрована можливість гнучного налаштування системи як за допомогою візуального інтерфейсу так і для програмістів які мають можливість розширення функціоналу.

Під час виконання роботи було використано багато літератури та технічної документації по розробці функціоналу та компонентів системи. Для створення такої комплесної архітектури потрібно зважено вибирати технології

та методи імплементації алгоритмів, оскільки це напряму може мати вплив як на кількість серверних потужностей які потрібні для фукнціонування так і для розробників які будуть підтримувати цю екосистему.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

2. https://valtek.com.ua/ua/system-integration/security-control-system/access-control/access-control-review — 5 грудня 2022
3. Дэвид А.Марка, Клемент Л.МакГоуэн. Методология структурного анализа и проектирования. Пер. с англ. – М.: ТОО ФРЭД,1993. – 240 с.
4. https://wiegand.com.ua/uslugi/ — 5грудня 2022
5. Сытник В.С., Сорока Х. и др. Компьютеризация информационных процессов на промышленных предприятиях – К.: Техника, Катовище, 1991-215с.
6. Theo Despoudis. TypeScript Quickly 1st Edition — Amazon, 2020
7. https://uk.wikipedia.org/wiki/Турнікет — 5 грудня 2022
8. https://securitylab.com.ua/ua/sistemy-kontrolya-dostupa/turnikety/ — 5 грудня 2022
9. https://vuejs.org/v2/guide/ — 5 грудня 2022
10. Клир Дж. Системология. Автоматизация решения системных задач. – М.: Радио и связь. 1990. – 538 с.
11. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html — 5 грудня 2022
12. Dan Vanderkam. Effective TypeScript: 62 Specific Ways to Improve Your TypeScript 1st Edition — Amazon, 2019
13. https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html — 5 грудня 2022
14. https://docs.aws.amazon.com/AmazonS3/latest/userguide/tutorial-s3-object-lambda-uppercase.html — 5 грудня 2022
15. https://developer.mozilla.org/ru/docs/Web/API/Geolocation_API — 5 грудня 2022
16. https://developers.google.com/maps/documentation/javascript/examples/map-geolocation — 5 грудня 2022

17. https://professorweb.ru/my/html/html5/level8/8_1.php — 5 грудня 2022
18. Дэвид Флэнаган: JavaScript. Полное руководство (7-е издание), 2020
19. Node.js Серверное программирование на JavaScript, 2020
20. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series) 1st Edition, 2017
21. https://sass-scss.ru/documentation/ — 5 грудня 2022
22. Роман Хади, Александр Аграновский. Практическая криптография — Amazon, 2020
23. https://webpack.js.org/concepts/ — 5 грудня 2022
24. Роберт Мартін. Чистый код. Создание, анализ и рефакторинг, 2008
25. Agile Software Development: Principles, Patterns and Practices — Pearson Education, 2002
26. Agile Principles, Patterns, And Practice — Prentice Hall, 2007
27. The Clean Coder: A Code of Conduct for Professional Programmers — Prentice Hall, 2011
28. https://medium.com/@cramirez92/s-o-l-i-d-the-first-5-priciples-of-object-oriented-design-with-javascript-790f6ac9b9fa — 5 грудня 2022
29. https://blog.logrocket.com/solid-principles-single-responsibility-in-javascript-frameworks/ — 5 грудня 2022
30. Николас Закас. ECMAScript 6: The Definitive Guide for JavaScript Developers — Prentice Hall, 2016
31. Шелли Пауэрс. Learning Node: Moving to the Server-Side 2nd Edition — Prentice Hall, 2012
32. https://graphql.org/learn/ — 5 грудня 2022
33. Згуровський М.З. Интегрированные системы оптимального управления и проектирования. – К.: Вища шк., 1990. – 240 с.
34. https://expressjs.com/en/guide/routing.html — 5 грудня 2022
35. https://expressjs.com/en/guide/writing-middleware.html — 5 грудня 2022
36. https://expressjs.com/en/guide/using-middleware.html — 5 грудня 2022
37. https://expressjs.com/en/guide/database-integration.html — 5 грудня

38. Theo Despoudis. TypeScript 4 Design Patterns and Best Practices: Discover effective techniques and design patterns for every programming task — Amazon, 2021
39. Shannon Bradshaw. MongoDB: The Definitive Guide by Shannon Bradshaw, Eoin Brazil — Amazon, 2010i
40. https://git-scm.com/doc — 5 грудня 2022
41. Gary Dessler. Human Resource Management, Global Editsion — Amazon, 2019
42. Laszlo Bock. Work Rules!: Insights from Inside Google That Will Transform How You Live and Lead — Amazon, 2015
43. Мушик Э., Мюллер П. Методы принятия технических решений: Пер. с нем. – М.: Мир, 1990. – 208 с.
44. https://axios-http.com/docs — 5 грудня 2022
45. Amy C. Edmondson. The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth — Amazon, 2018
46. Donald Knuth. The Art of Computer Programming — Amazon, 2011
47. Erich Gamma. Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition — Amazon, 1994
48. https://jwt.io/introduction — 5 грудня 2022
49. Системы поддержки экономических решений. / Багриновский АН СССР. Центральный экономико-математический институт. – М.: 1991. – 234 с.
50. https://docs.github.com/en — 5 грудня 2022
51. Eric Freeman. Head First Design Patterns (A Brain Friendly Guide) — Amazon, 2004
52. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К.: Держстандарт України. 1995. – 38 с.
53. Герасимов Б.М., Тарасов В.А., Токарев И.В. Человеко-машинные системы принятия решений с элементами искусственного интеллекта. – К.: Наукова думка, 1993. – 183 с.
54. Эддоус М., Стенсфилд Р. Методы принятия решения / Пер. с англ. под ред. И.И. Елисеевой. – М.: ИОНИТИ, 1997. – 590 с

55. Тихомиров Ю.В. Microsoft SQL Server 7.0. – СПб.: БХВ –Санкт-Петербург, 1999.-720с.
56. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. в англ. – М.: ДМК Пресс, 2001. – 176 с.: ил. (Серия «Объектно-ориентированные технологии в программировании»).
57. Колпаков В.М. Теорія і практика прийняття управлінських рішень.:Навч. Посібник (Рос. Мовою): К.:МАУП, 2000.–255с.