КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено
Завідувач кафедри ІСТ
Олександр КУЧАНСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)
"___"_____ 2022р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальності 126 «Інформаційні системи та технології»
освітньої програми «Програмні технології інтернет речей»

на тему: «Розробка системи логістики доставки замовлень хлібозаводу "Кулиничі"»

Виконала: студентка 4 курсу, групи ІР-41
(шифр групи)

Соломія ТИМОЩУК _____ _____
(Ім'я, ПРІЗВИЩЕ) (підпис)

Керівник к.т.н., доцент Мирослава ГЛАДКА _____ _____
(посада, науковий ступінь, вчене звання, Ім'я, ПРІЗВИЩЕ) (підпис)

Консультант нормо контроль к. т. н., доц.Ростислав ЛІСНЕВСЬКИЙ _____
(назва розділу) (посада, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент директор підприємства «Рокитне Хліб» Лілія ТИМОЩУК._____
(посада, науковий ступінь, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у пояснювальна записка не має запозичень з праць інших авторів без відповідних посилань.
Здобувач освіти _____
(підпис)

Київ – 2022 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри,
д.т.н., доцент
Олександр КУЧАНСЬКИЙ

_____
«__» _____ 2022 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Здобувач освіти: Соломія ТИМОЩУК
Група: ІР-41

1. **Тема кваліфікаційна робота бакалавра**: «Розробка системи логістики доставки замовлень хлібзаводу “Кулиничі”».
Затверджена протоколом засідання кафедри ІСТ №18/20 від 03.12.2021 року
2. **Строк подання студентом готової роботи** - «22» червня 2022 р.
3. **Вихідні дані до роботи**: Дослідження можливості імплементації системи логістики на підприємстві з виготовлення хлібобулочних виробів. Розроблення алгоритму менеджменту та керування розумною системою логістики хлібопекарні. Обробка та аналіз даних, отриманих з датчиків транспортної системи підтриємства та безпосередньо транспортних засобів.
4. Зміст роботи: РОЗДІЛ 1. АНАЛІЗ ВИМОГ ДО СИСТЕМИ, ЩО ДОСЛІДЖУЄТЬСЯ ТА ОПИС ІМПЛЕМЕНТАЦІЇ РОЗУМНИХ СИСТЕМ ЛОГІСТИКИ(Застосування рішень IoT у логістиці підприємств, найефективніші варіанти впровадження системи логістики доставки, потенційні можливості застосування системи логістики, можливість комплексної взаємодії пристроїв, огляд аналогів та актуальність обраної тематики, постановка задачі). РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ СИСТЕМИ(основні принципи проектування системи, необхідні компоненти та пристрої для реалізації проектування системи, вибір глобальної системи позиціонування(GPS), автоматична система ідентифікації транспортних засобів та телематичні дані про транспортні засоби, радіочастотна ідентифікація, системи на основі камер, вибір інструментів реалізації системи) . РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ(проектування архітектури бази даних, розробка інтерфейсу системи, тестування системи)

5. **Перелік графічного матеріалу**: схеми розміщення датчиків на транспортних засобах підприємства, зображення систем-аналогів, схеми таблиць бази даних, алгоритми керування IOT пристроями, схеми інформаційних потоків у IOT системі, зображення інтерфейсу користувача, зображення інфографіки досліджуваних та отримуваних даних.

6. **Календарний план виконання роботи:**

Етапи виконання кваліфікаційної роботи бакалавра	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи бакалавра	01.10.2022	виконано
2. Наказ про затвердження тем кваліфікаційної роботи бакалавра та призначення керівників	03.12.2021	виконано
3. Розробка плану кваліфікаційної роботи бакалавра і його погодження з керівником	25.12.2021	виконано
4. Написання I розділу кваліфікаційної роботи	20.03.2022	виконано
5. Написання II розділу кваліфікаційної роботи	25.04.2022	виконано
6. Написання III розділу кваліфікаційної роботи	25.04.2022	виконано
7. Підготовка висновків і пропозицій	05.05.2022	виконано
8. Попередній захист кваліфікаційної роботи	07.06.2022	виконано
9. Перевірка на плагіат	15.06.2021	виконано
10. Нормоконтроль	17.06.2021	виконано
11. Рецензування кваліфікаційної роботи бакалавра і представлення роботи на кафедру	15.06.2022	виконано
12. Захист кваліфікаційної роботи бакалавра	23.06.2022	

Дата видачі завдання «03» 12 2021 р.

Керівник роботи: к. т. н., доцент Мирослава ГЛАДКА_____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «бакалавр» 4-го курсу групи ІР-41

Соломія ТИМОЩУК_____

(Власне Ім'я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота бакалавра Соломії ТИМОЩУК

Тема роботи: «Розробка системи логістики доставки замовлень хлібзаводу “Кулиничі"».

Мета кваліфікаційної роботи бакалавра – розробка розумної системи логістики доставки замовлень для хлібопекарні з використанням IoT технологій та розробки схеми, алгоритму коректного моніторингу, обробки та аналізу даних отримуваних з транспортної системи підприємства.

Об’єкт дослідження – IoT система логістики.

Предмет дослідження – імплементація розумної системи логістики.

Кваліфікаційна робота бакалавра складається зі змісту, вступу, основної частини, яка включає 4 розділи, 4 висновки та списку використаних джерел. Всього ** сторінок.

КЛЮЧОВІ СЛОВА: IOT, СИСТЕМА ЛОГІСТИКИ, РОЗУМНА СИСТЕМА, ДАТЧИКИ, IOT ПРИСТРОЇ

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of bachelor Solomiia TYMOSHCHUK.

Work topic: “Implementation of Logistics System for Orders Delivery of Kulinichi Bakery"

The purpose of the bachelor’s qualification work is the development of a smart logistics system for delivery of orders for the bakery using IoT technologies and scheme development, an algorithm for correct monitoring, processing, and analysis of data obtained from the transport system of the enterprise.

The object of research is the IoT logistics system,

The subject of research is the implementation of the intelligent logistics system.

The bachelor's qualification work consists of the content, introduction, and main part, which includes four sections, conclusions, and a list of sources used. Total ** pages.

KEYWORDS: IOT, LOGISTICS SYSTEM, SMART SYSTEM, SENSORS, IOT DEVICES

ЗМІСТ

ВСТУП

За останні десятиліття ланцюги поставок стають все більш глобальними, а зростаючий попит на електронну комерцію призвів до збільшення поставок і очікувань клієнтів. Паралельно зі все більш вимогливими глобальними вимогами та вимогами клієнтів, логістична галузь має справлятися з величезною нестачею персоналу у транспортному секторі, фрагментованими ринками та низькою прибутковістю. Тому важливо знайти рішення для оптимізації логістичного процесу з точки зору ефективності, вантажопідйомності та планування. Сучасні технології, такі як IoT, Blockchain та хмара, відіграють головну роль у цій темі,

Проблематика логістичної системи підприємства є важливою складовою дослідження, адже успішне функціонування будь–якої організації, її конкурентоспроможність безпосередньо залежать від грамотної побудови логістичної системи підприємства. Логістична діяльність підприємства – це частина управління ланцюгом поставок, яка планує, впроваджує та контролює ефективний прямий та зворотний потік зберігання товарів, послуг та пов'язаної з ними інформації між точкою походження та точкою споживання з метою задоволення потреб клієнтів. Управління логістикою є важливою складовою ділових операцій, оскільки її діяльність впливає не тільки на внутрішні процеси, але і на відносини з клієнтами.

Для підприємства з виготовлення хліба вкрай важлива вчасна доставка великої кількості замовлень в різні місця з врахуванням трафіку, витрат пального, витрат на робочу силу та з урахуванням економії часу. Тому важливим є застосування IOT девайсів для покращення та удосконалення системи логістики замовлень хлібопекарні.

Отже основними завданнями кваліфікаційної роботи бакалавра є необхідність:

- дослідити можливість впровадження IoT системи в логістичній галузі підприємства з виготовлення хліба;
- розробити схему встановлення відповідних пристроїв та датчиків на транспортних засобах підприємства;
- досягти ефективності в обробці отримуваної інформації;

- досягти оптимізації у процесах зчитування та зберігання інформації з датчиків;
- досягти ефективності у організації доставки хлібобулочних виробів.

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ДО СИСТЕМИ, ЩО ДОСЛІДЖУЄТЬСЯ ТА ОПИС ІМПЛЕМЕНТАЦІЇ РОЗУМНИХ СИСТЕМ ЛОГІСТИКИ

1.1 Застосування рішень IoT у логістиці підприємств

Сьогодні існує велика кількість можливостей застосування IoT у транспортному секторі підприємств різних галузей, тому перехід до включення технологій Інтернету речей у систему логістики спроможний здійснити прорив у розвитку підприємства в цілому. Крім того, розгортання таких систем поширюється в різних випадках використання, дні, коли операції без підтримки IoT забезпечували конкурентну перевагу в транспорті, ймовірно, підходять до кінця. [1]

Зокрема рішення IoT складається з сенсорних мереж, які генерують різні типи даних, які компанія може аналізувати, щоб створити інформацію для прийняття бізнес-рішень. Екосистема IoT поєднує ці рішення з іншими цифровими інструментами та технологіями, щоб створити ще більшу цінність для компанії та її клієнтів.

Сьогодні в логістиці та дистрибуції найбільш актуальними технологіями для екосистеми Інтернету речей, як штучний інтелект (AI), прогнозна аналітика, блокчейн і хмарні обчислення. Але концепція екосистеми IoT не залежить від технологій: у майбутньому та в інших галузях можна створити переконливу цінність шляхом інтеграції периферійних і квантових обчислень, доповненої реальності та інших технологій.

Найбільш можливим застосуванням IoT у системі логістики видається використання спеціальних датчиків саме на транспорті, який здійснює перевезення. З детальною передачею інформації про використання пального, вимірювання рівню трафіку на дорогах, тиску в шинах, а також стан двигуна і аналіз коректності руху/маршруту за допомогою вбудованих передньої та задньої камери.

Хлібопекарня “Кулиничі” є підприємством з виготовлення хліба, що займається доставкою хлібобулочних виробів по 959 містам України та більше ніж 10 областям. Але управління логістикою підприємства і досі здійснюється без налагодженої системи. Завдяки логістичній системі, розробленій спеціально для хлібозаводу “Кулиничі”, можна покращити їх оперативне управління та

конкурувати з іншими хлібопекарнями, у яких вже імплементовані розумні системи логістики. Ця система використовуватиме веб-додаток і IoT пристрої для відслідковування відповідності доставки замовлень, витрат ресурсів таких як час, пальне, робоча сила і тд.

Перш ніж розробляти логістичну систему, потрібно знати, що таке логістика. Спираючись на інформацію, знайдену в джерелах, саме [9] можна стверджувати, що логістика - це управління потоком продукції між точкою виготовлення і точкою клієнта, який споживає продукт. Загалом можна стверджувати, що логістика є частиною постачання, ланцюгом, який забезпечує ефективне транспортування та зберігання продуктів та потоку інформації.
Розвиток IoT вплинув на зростання багатьох секторів, особливо на сектор логістики. Нові технології виробництва дозволяють інтегрувати у виробничі підприємства гнучкі процеси, що дозволяють впровадити інтелектуальну систему, яка може обмінюватися даними між собою та контролювати процес виробництва та логістики, одночасно.

Логістична система є одним із транспортних термінів для координації функцій управління промисловістю, сучасних системних логістичних додатків, які призначені для обслуговування та підтримки різноманітних бізнес та промислових функцій. Логістична система, яка спроможна використовувати IoT, спроможна здійснювати планування, управління та керування доставкою замовлень та інформаційним потоком. Виходячи з даних визначень, логістичну систему можна визначити як систему, яка використовує IoT для управління потоками різних даних, пов’язаних безпосередньо з транспортною системою підприємства і промислових процесів.

Що стосується компонентів системи логістики, логістична система має багато системних компонентів, які складаються з основних послуг управління ланцюгом поставок, таких як: обробка замовлень, зберігання, транспортування та фінансове управління. Як висновок розумна логістична система може мати чотири підсистеми. такі як: (1) можливості обробки замовлень, запитів та зберігання інформації, (2) функції планування та впровадження розумної доставки замовлень, (3) функція точного та своєчасного зворотного зв’язку

транспортної системи та транспортних засобів, (4) своєчасний статистичний аналіз даних і функції прийняття рішень.

1.2 Варіанти впровадження системи логістики доставки

Щоб побачити потужність справді підключеної екосистеми IoT, розглянемо приклад підключених транспортних засобів. Як показано на рис 1.1, сучасні підключені вантажівки не тільки переміщують вантаж, але зазвичай генерують величезні обсяги даних, таких як місцезнаходження, стан двигуна (швидкість, час простою, рівень палива), умови навколишнього середовища (температура, вологість, освітленість), дані про транспортний засіб (поштовхи, рух), поведінка водія (втома, непостійні моделі водіння) та безпека (крадіжка, втручання, активація сигналізації).

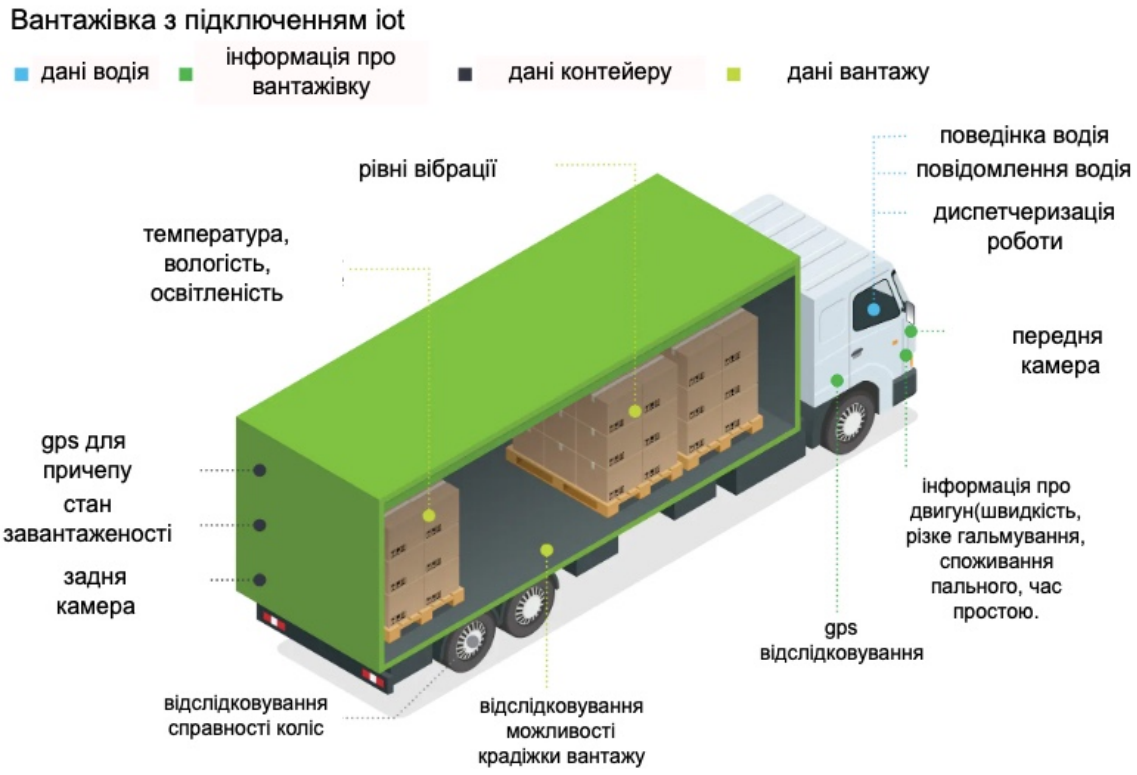


Рисунок 1.1 - Схема вантажівки з підключеним IoT

Завантаження цих даних з інтелектуального автопарку в хмарну систему даних і передача їх в інші технології та процеси транспортування можуть підтримувати маршрутизацію, відстеження відправлень, відповідність якості, керування автопарком, керування роботою водіїв та безпеку. У наскрізній транспортній екосистемі з підтримкою IoT інформація буде безперебійно проходити по мережі, створюючи цикл інформації на рис. 1.2.

Рисунок 1.2 - Схема потоку інформації у транспортній екосистемі

Тут датчики різних транспортних засобів можуть автоматично завантажувати телематичні дані, такі як витрата палива, місцезнаходження та температура вантажу, у підключені хмарні сховища та базу даних спеціального призначення, які можуть об’єднувати інформацію в структуровані набори даних і передавати її в алгоритми аналітики, які використовують ШІ. Отримані у результаті дані про стан транспортного засобу, продуктивність водія, якість вантажу тощо можуть дозволити центру управління підприємства контролювати показники роботи автопарку та показники безпеки, а також прогнозувати проблеми з обслуговуванням, простої, проблеми з вантажем і навіть дорожньо-транспортні пригоди.

Підприємство може використовувати ці дані в режимі реального часу для відповідних дій, включаючи оптимізацію маршрутів, динамічне планування, профілактичне обслуговування та швидке реагування у разі поломки або аварії.

1.3 Потенційні можливості застосування системи логістики

Потенційні застосування виходять за межі підключеного автомобіля:

Термінальні операції**.** За допомогою [3] Інтернету речей і технологій відстеження місцезнаходження, таких як GPS, термінали (або автотранспортні станції) можуть отримувати оновлену інформацію про вхідні відправлення,

наприклад очікуваний час стикування, кількість відправлення та вимоги до зберігання на рис 1.3. Поєднання Інтернету речей з штучним інтелектом і прогнозною аналітикою дозволяє терміналам розумно використовувати ці дані, щоб краще планувати вихідні відправлення та керувати потужністю. Використовуючи інформаційні панелі з підтримкою IoT, термінал може підтримувати оновлені показники використання потужності та термінів відвантаження.

Рисунок 1.3 - Схема моніторингу багатосмугового трафіку

Безпека транспортування. Аварії, травми, непрацездатні водії, безпека дорожнього руху і пов’язані з цим збитки — є одними з найпомітніших проблем для підприємства Релевантні технології на ринку зосереджені на підвищенні безпеки транспортування: в автомобільних телематичних рішеннях використовуються акселерометри, трекери та монітори двигуна для збору даних про місцезнаходження, споживання палива, швидкості та гальмування. [1]

Підключені фізичні пристрої (наприклад, мобільні телефони) забезпечують зворотний зв’язок водіям та іншим особам за допомогою сповіщень про порушення порогів безпеки, завершуючи цикл від фізичного до цифрового до фізичного в повній екосистемі.

Можливість прогнозування необхідності обслуговування. Розумні технології — поєднання IoT із планами безпеки та технічного обслуговування в рамках систем управління активами можуть дозволити постачальникам транспортних послуг комплексно керувати обслуговуванням активів.

Одне з рішень, що використовує датчики збору даних, які фіксують погодні умови та навантаження на обладнання в своїх поїздах для живлення віддаленої діагностичної платформи в режимі реального часу. У поєднанні з штучним інтелектом і прогнозною аналітикою це може покращити можливості прогнозного технічного обслуговування, щоб передбачити майбутні збої.

Це дозволяє включати виправлення під час використання активу, покращуючи продуктивність, ефективність і довговічність транспорту і очікувано, зменшить витрати на технічне обслуговування парку.

1.4 Можливість комплексної взаємодії пристроїв

Екосистема IoT дозволяє всім точкам мережі спілкуватися один з одним у режимі реального часу за допомогою технології інтегрованих комунікаційних систем. Використовуючи інформацію з низькими затримками, власник підприємства може створити динамічну мережу, яка враховує попит і доступність автопарку, а також приймати рішення в режимі реального часу щодо маршрутизації та керування автопарком.

Описана вище система складається з підключених пристроїв у автомобільному парку, які збирають інформацію про маршрут і надсилають постійні оновлення в хмару. Завдяки ефективному використанню аналітики власники підприємства можуть приймати динамічні рішення щодо маршрутизації та зміни маршруту при необхідності в режимі реального часу за допомогою технології комунікаційних систем.

Крім того, адміністратори підприємства можуть використовувати технології допомоги водієві для визначення умов, придатних для здійснення маршруту.

Покращення управління станом продукції. Так як переміщення хлібобулочних виробів через ланцюжок поставок супроводжується високим ризиком втрат, то системи Інтернету речей можуть допомогти підвищити ефективність транспортування забезпечуючи безпеку та якість продукції, а також

можуть визначити проблеми, які спричиняють втрату вартості продукту, використовуючи аналітику даних датчиків. [4]

1.5 Огляд аналогів та актуальність обраної тематики

Сьогодні транспорт зазнає кардинальних змін через технологічний стрибок в інформаційних технологіях. І найбільший ефект у такій трансформації має Інтернет речей. Тесла з автопарковкою, вантажівки без водія, вантажні дрони та роботи Amazon для доставки — це лише деякі з численних руйнівних інновацій, які стали можливими в логістиці завдяки підключенню через Інтернет. GPS, стільниковий Інтернет і блокчейн — це технології, здатні вирішити багато «традиційних» проблем у транспортному секторі.

Оскільки автоматизація як така спрямована на пом’якшення негативного людського фактору, дрони та автономні транспортні засоби здаються можливим рішенням для випадків використання логістики, де люди поступаються машинам.

І молоді стартапи, і зрілі промислові гіганти, які працюють у секторі IoT, чітко усвідомлюють, що логістика є дуже перспективною сферою для різноманітних реалізацій IoT. Наявні рішення IoT, які вже довели свою доцільність у транспортуванні, швидше підвищать інтерес клієнтів до інших новорозроблених систем, ніж перегріють конкуренцію. Звичайно, конкуренція буде зростати, але через відносну зародковість сектору IoT буде достатньо місця для кожного гідного гравця на ринку.

Поділяючи таке бачення, нижче представлено наступні приклади життєздатних рішень IoT у розумному транспорті.

SkyCell. Коли десь у вашому ланцюжку поставок з вашими контейнерами відбувається щось не так, ви повинні бути проінформовані про це, щоб вчасно втрутитися. Система SkyCell [10] , заснована як на технологіях IoT, так і на блокчейн, гарантує вам повну інформацію про стан всіх ваших відправок у режимі реального часу. Розумні контейнери, оснащені датчиками IoT, підключеними до хмарної бази даних, забезпечують повне відстеження всіх поставок. Віддалений контроль відправлень підтримується незмінними записами в розподіленій книзі. Саме тут блокчейн зустрічається з Інтернетом речей, саме інтерфейс системи наведений на рис. 1.4.

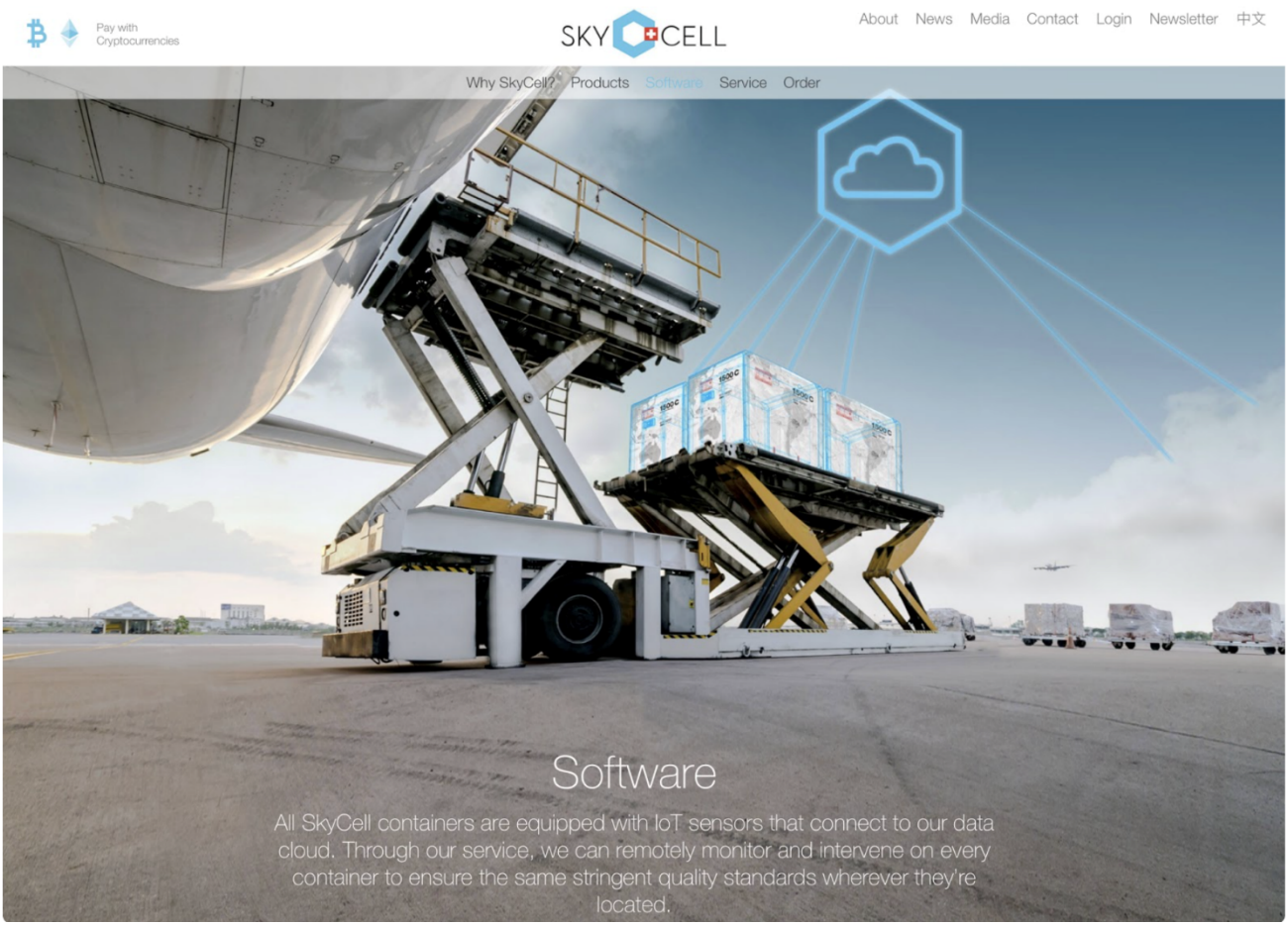


Рисунок 1.4 - Система логістики Skycell

Безпека логістики, заснована на надійній перевірці даних, — це те, що блокчейн може забезпечити з надзвичайною ефективністю та надійністю. Крадіжка кобальту в Роттердамі, описана на початку цієї статті, навряд чи могла статися відвантаженням, підключеним до системи IoT відстеження та відстеження. Як тільки IoT робить неживі об'єкти (речі) буквально комунікативними, надана здатність спілкуватися з контейнерами може усунути численні ризики, притаманні рутині логістики. SkyCell правильно зрозумів: більше не потрібно покладатися на випадковість, якщо IoT пропонує вам точні знання про все, що відбувається у вашому ланцюжку поставок. Розумне перевезення передбачає передусім розумні контейнери. А можливість оплатити послугу відстеження криптовалютою (BTC та ETH) не завадить у 21 столітті.

Руптела - інтерфейс наведений на рис. 1.5. Постачальник управління автопарком Ruptela [11] з Литви не новачок на ринку логістики. Клієнти з 127 країн доступні з 2007 року. Успіх заснований на широкому спектрі послуг від Ruptela. На відміну від SkyCell, компанія зосереджується на безпеці

транспортних засобів, крім контейнерів. Крім того, Ruptela пропонує апаратні датчики та аксесуари разом із програмним забезпеченням. Такий цілісний підхід до логістичної безпеки дає свої плоди.

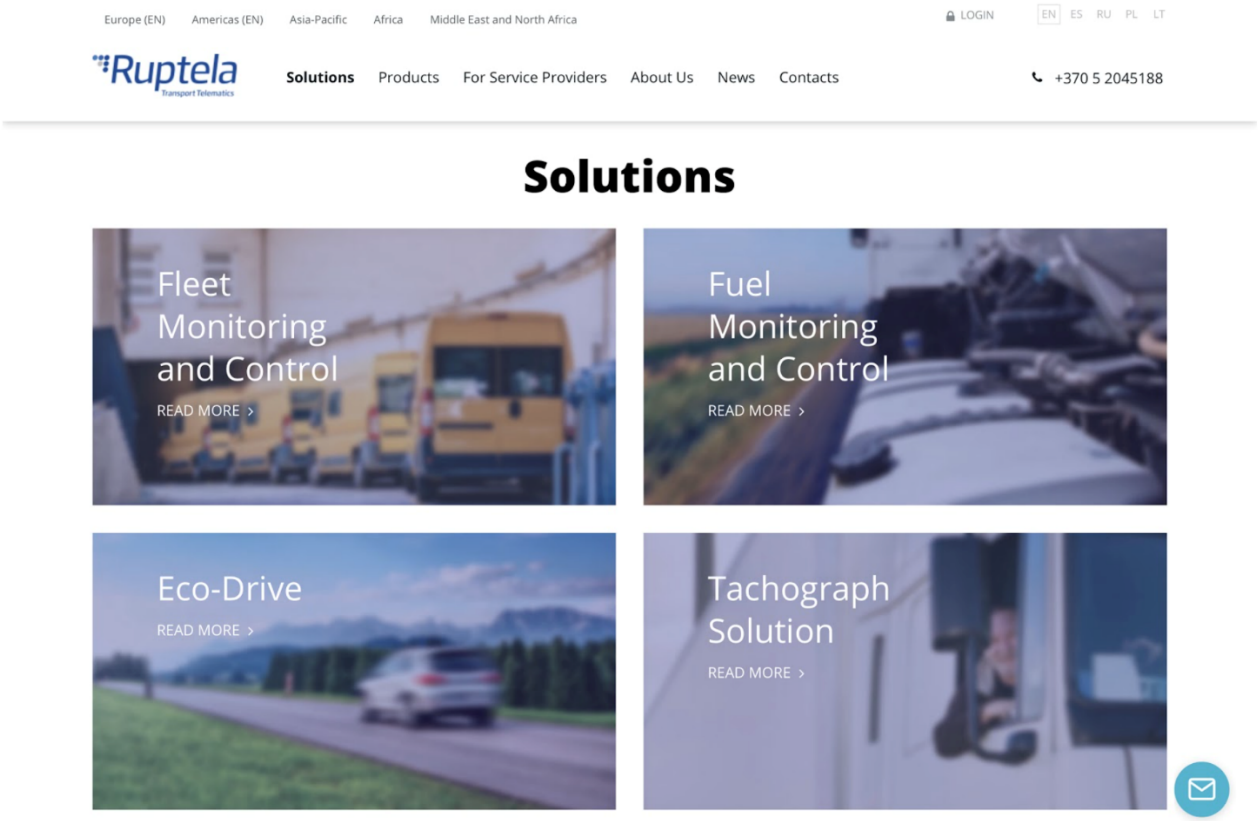


Рисунок 1.5 - Система логістики Ruptela

Система моніторингу та керування автопарком на основі GPS від Ruptela охоплює стандартну кількість завдань для кожного менеджера транспорту: дані про транспортні засоби, водіїв, пасажирів та вантажі збираються в режимі реального часу за допомогою різних пристроїв IoT, що стосуються домену. Оскільки паливо – це кров для транспортних засобів, Ruptela піклується про здоров'я автопарків за допомогою систем контролю палива та захисту від крадіжки палива. Датчики дверей з повідомленнями про відкриття запобігають несанкціонований доступ до вантажу. Різні засоби ідентифікації водія разом із такими простими функціями, як дистанційне блокування запалювання, допомагають убезпечити транспортні засоби. Постійна температура моніторинг в холодильному обладнанні захищає швидкопсувні продукти від пошкодження. Таким чином, Ruptela надає клієнтам один з найбільш повних наборів рішень IoT на ринку логістичної безпеки.

Sawtooth

Морепродукти швидко псуються, і це проблема, якщо ланцюги поставок морепродуктів не забезпечені надійною системою IoT відстеження та відстеження. Стартап Sawtooth на базі Hyperledger показує, як може працювати така система. Як і у випадку зі SkyCell, ми бачимо, який плідний симбіоз виникає, коли Інтернет речей доповнюється розподіленими реєстрами.

Sawtooth [12] робить ставку на повну прозорість усього шляху морепродуктів від рибальських суден до наших столів. Ніхто, крім кінцевих споживачів, не повинен розбиратися, чи є той чи інший постачальник морепродуктів достатньо надійним. Розташування, температура, вологість, рух, удари та багато інших параметрів телеметрії, які контролюються пристроями IoT, вступають у дію, коли відбувається безпечна логістика морепродуктів. Але як ми можемо переконатися, що зібрані дані є оригінальними? Проблема довіри – це те, що можуть вирішити технології розподіленої книги (DLT). Ось чому Sawtooth застосував Hyperledger до своєї системи відстеження, завдяки якій «постачальники та споживачі знають, що вони отримують і за що платять». Однак рішення Sawtooth не обмежуються лише ланцюгами постачання морепродуктів і ланцюг роботи можна побачити на рис. 1.6.

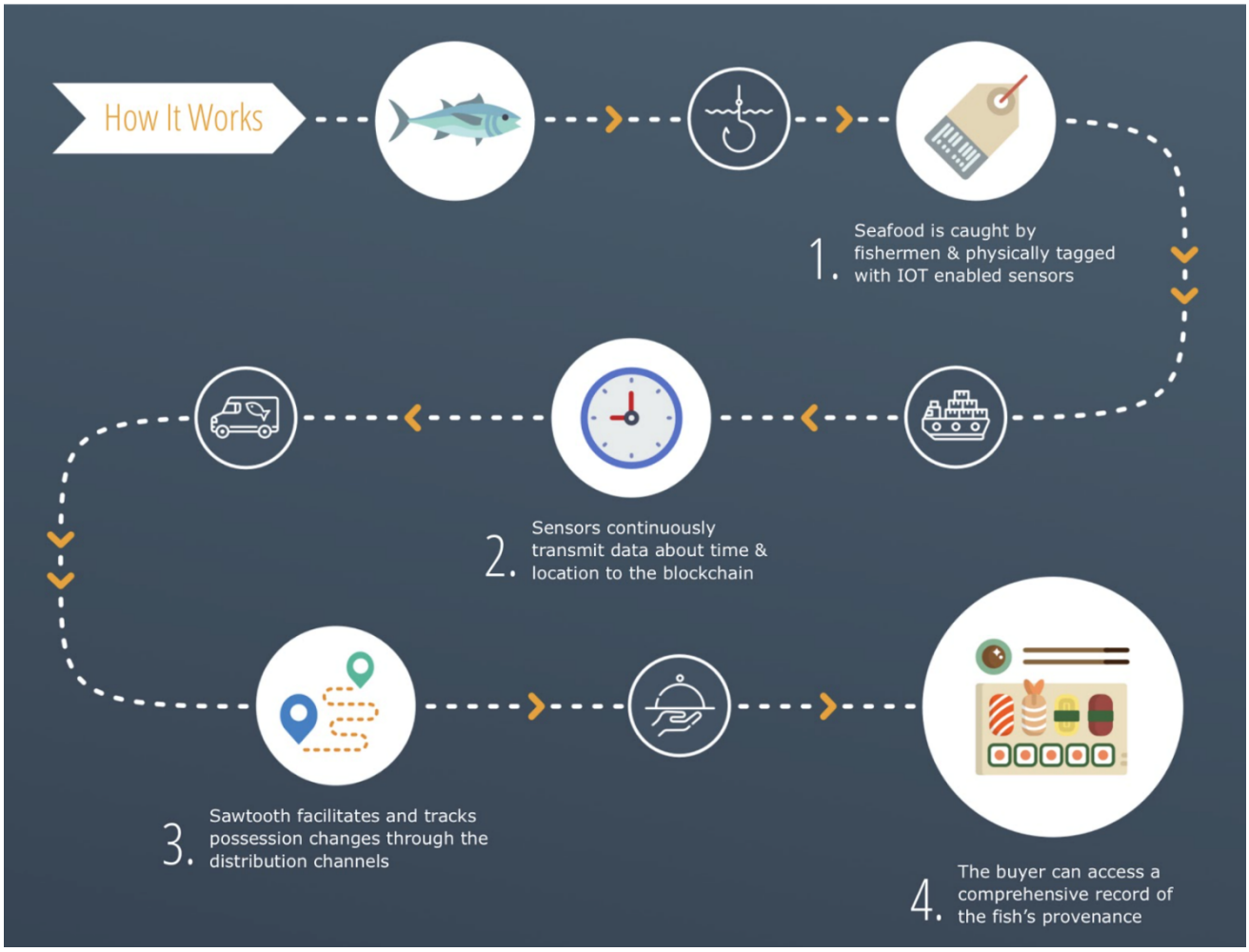


Рисунок 1.6 - Система логістики Sawtooth

1.6. Постановка задачі

Під час опису предметної області вдалось встановити вимоги до розроблюваної системи. Тобто, залежно від процесу обробки замовлень система повинна включати в себе ряд функцій, що дадуть змогу якісно та швидко додавати та аналізувати дані доставки замовлень та логістики належним чином.

Функції передбачені в системі, що реалізується:

- Система повинна дозволяти користувачеві вносити дані про водіїв, закріплених за конкретним транспортним засобом, про транспортні засоби, датчики, які містяться у кожному транспортному засобі, дані про середню температуру в транспортних засобах, тиск в шинах, середню швидкість, активний та пасивний час руху автомобіля, витрати та споживання палива, час та середні енергетичні затрати на роботу двигуна.
- Передбачена можливість внесення, редагування, видалення даних.

- Фільтрація пошуку по поточним транспортним засобам та датчикам, включно з датою їх створення чи оновлення.

При роботі з програмою адміністратор повинен мати можливість вирішувати такі завдання:

- додавати, редагувати базу;
- переглядати дані про логістику та середні показники з транспортних засобів.

Кожен доданий запис має бути унікальним, дані мають бути доступні до перегляду у будь-який час.

Система має бути реалізована обраною мовою програмування з використанням реляційної бази даних.

Вхідною інформацією для розроблюваної системи є:

- Дані про водіїв.
- Дані про транспортні засоби.
- Дані про датчики.
- Дані про показники температури, швидкості тз, роботи двигуна, часу роботи тз.

1.7 Висновок до розділу

Отже, у даному розділі було досліджено приклади застосування IOT у логістичній сфері, також було оцінено можливість взаємодії компонентів системи. Крім цього, було наведено аналоги до розроблюваної системи логістики і проаналізовано їх актуальність.

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ СИСТЕМИ

2.1. Основні принципи проектування системи

Одною із ефективних способів для організацій побудувати свої екосистеми IoT — це рішення, які є специфічними для галузі виготовлення хлібобулочних виробів. Рішення, розроблені спеціально для цієї галузі зазвичай можуть генерувати результати швидше, ніж ті, які потрібно переробляти, щоб відповідати потрібному середовищу.

Наближаючись до впровадження, можна передбачити такий початковий план дій:

1. Визначення вдосконалення, необхідних для модернізації платформ і створення екосистеми, що постійно розвивається, щоб використовувати потужність даних для прийняття більш розумних і швидших рішень.
2. Звернення уваги на потенційні перешкоди під час налаштування інформаційного потоку (наприклад, проблеми з підключенням на комунікаційному рівні, обсяг даних і частота передачі через інтерфейси).
3. Використовування масштабованих рішень за допомогою прискорювачів та агрегаторів, що може призвести до припинення більшості робіт з розробки, прискорення рентабельності інвестицій та збільшення прибутковості.
4. Інтегрування поточних фреймворків щоб дані безперешкодно протікали через хмарні рішення.
5. Створіть інструменти видимості для організації та клієнтів на вершині екосистеми, щоб збільшити цінність.

Першим кроком в реалізації системи логістики є створення схеми із детальним описом рівнів використання, застосунків, платформи, інформації, доступу. Передбачено, що для впровадження даної системи пекарня буде використовувати транспортний центр зі створеною системою менеджменту транспортної системи, системою обробки даних, системою безпеки; комп’ютерний та мобільний термінал, які дозволяють відстежувати дані у реальному часі, а також центри моніторингу та диспетчеризації, що відповідають за коректне оброблення замовлень та обрахунок витратних матеріалів по маршруту. На рівні платформи відбувається відстеження логістичних процесів у реальному часі, автоматичне сортування продукції відповідно до маршруту,

оптимізація процесів виготовлення та доставки товарів, а також процес обробки замовлень. Рівень інформації відповідає за коректне збереження усіх отриманих даних у базу даних для збереження інформації про логістику з датчиків та розумних пристроїв, інформація про виготовлення та товарообіг - у базу даних продукції, замовлення - у базу даних замовлень. На рис. 2.1 зображена схема логістики підприємства базована на Інтернеті речей, яка відображає процеси описані вище.

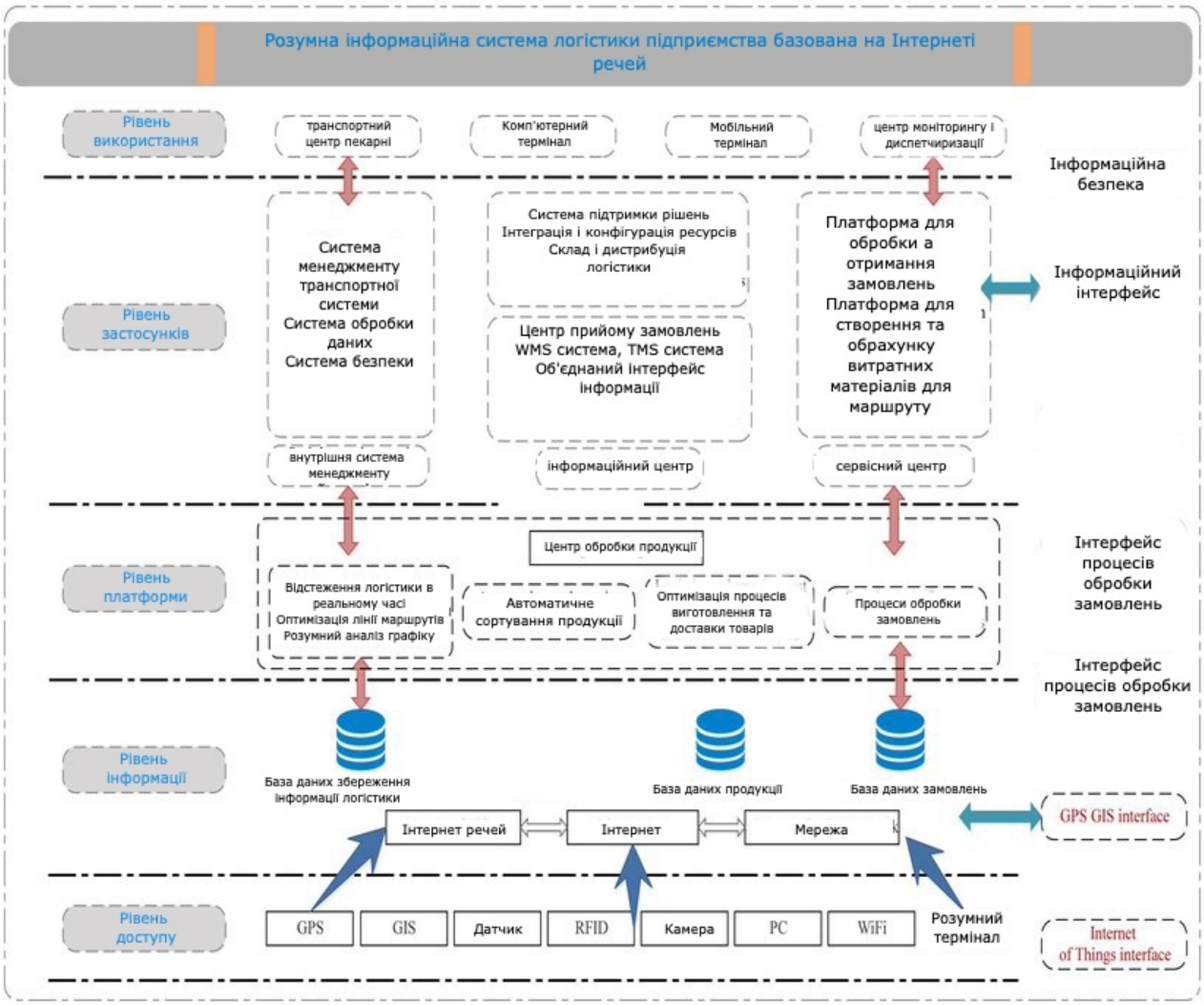


Рисунок 2.1 - Схема логістики підприємства, базована на Інтернеті речей

Для реалізації проекту необхідно передбачити відповідне тестове середовище: цп з сервером бази даних(наприклад Inter L5520), опз(Kingston), операційна система(Centos), база даних(MySQL).

Таблиця 2.1 - Обладнання, передбачене системою логістики

Тестове середовище	Сервер бази даних	Сервер додатків	Клієнт
1	2	3	4
Цп	Inter L5520	Inter L5520	IOS A13
опз	Kingston 2g	Kingston 2g	4g
Операційна система	Centos 5.5	Centos 6.5	IOS
проміжний шар		Apache Tomcat 7.0	
база даних	MySQL 5.5.28		

2.2. Необхідні компоненти та пристрої для реалізації проектування системи

Розміщення датчиків та пристроїв на транспортному засобі обумовлено необхідністю зчитування даних з певною частотою. Отже, для реалізації даної системи необхідним є пристрій для вимірювання тиску в шинах, термометр, задня камера, пристрій для вимірювання витрат пального, тахограф, датчик для вимірювання стану двигуна, передня камера.

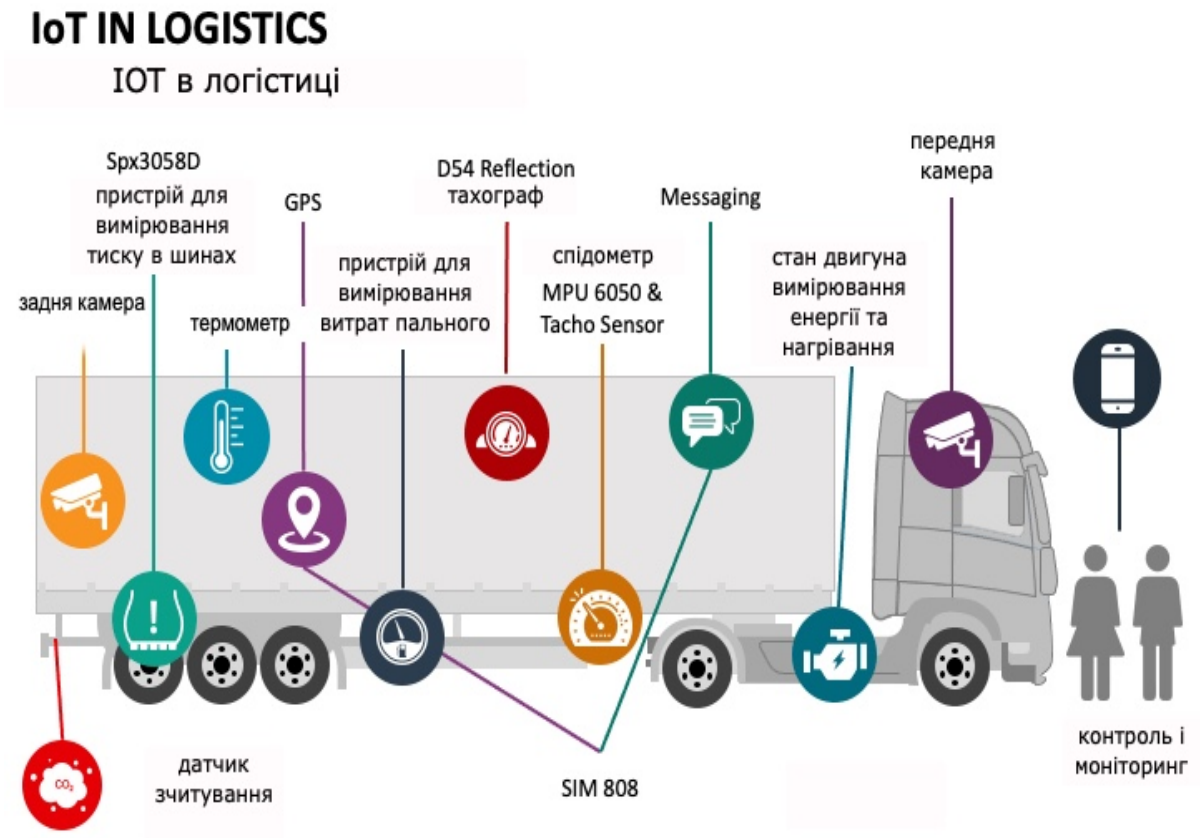


Рисунок 2.2 - Схема розміщення датчиків та пристроїв на транспорті підприємства

На рис. 2.2 відображена схема розміщення датчиків на транспортному засобі, що здійснює доставку хлібобулочних виробів, також у таблиці 2.2 є перелік самих датчиків.

Таблиця 2.2 - Датчики та пристрої, необхідні для реалізації схеми

Порядковий номер	**Зображення**	**Назва**
1	2	3
1.		Датчик зчитування тиску в шинах *Tire Pressure Monitoring System*
2.		Датчик для вимірювання рівня споживання пального
3.		GPS - датчик
4.		Передня та задня камери

Продовження таблиці 2.2

Порядковий номер	**Зображення**	**Назва**
1	2	3
5.		Датчики вимірювання потужності двигуна
6.		Розумний тахограф для вимірювання швидкості

1. TPMS (*Tire Pressure Monitoring System*) — **це система активного контролю тиску повітря в шинах автомобіля** . У режимі реального часу він інформує водія про зміну тиску в шинах (падіння по відношенню до рекомендованого значення), показуючи піктограму з бічним перерізом шини та знаком оклику.

У прямому типі датчики можуть бути встановлені всередині або зовні. Для внутрішнього монтажу датчики розміщуються або всередині шини, або на місці її батьківського клапана. Однак для зовнішнього монтажу датчик розміщують на клапані. В обох випадках тиск повітря в шинах транспортного засобу контролюється на постійній основі, і водій інформується про його зміни.

2. Датчик палива – це пристрій, призначений для точного вимірювання рівня палива в баках автомобіля. Схему датчика наведено на рис 2.3. Згідно з цими вимірюваннями, платформа відстеження та телематики GPS має такі дані:

- рівень палива в баку автомобіля;
- витрата палива за період часу;
- середня витрата палива. наприклад милі на галон (mpg);
- заправки або зливу палива.

Застосування.

Датчики палива використовуються як на стаціонарних агрегатах, наприклад, на паливних баках на АЗС, так і на транспортних засобах: автомобілях, локомотивах, кораблях тощо. [18]

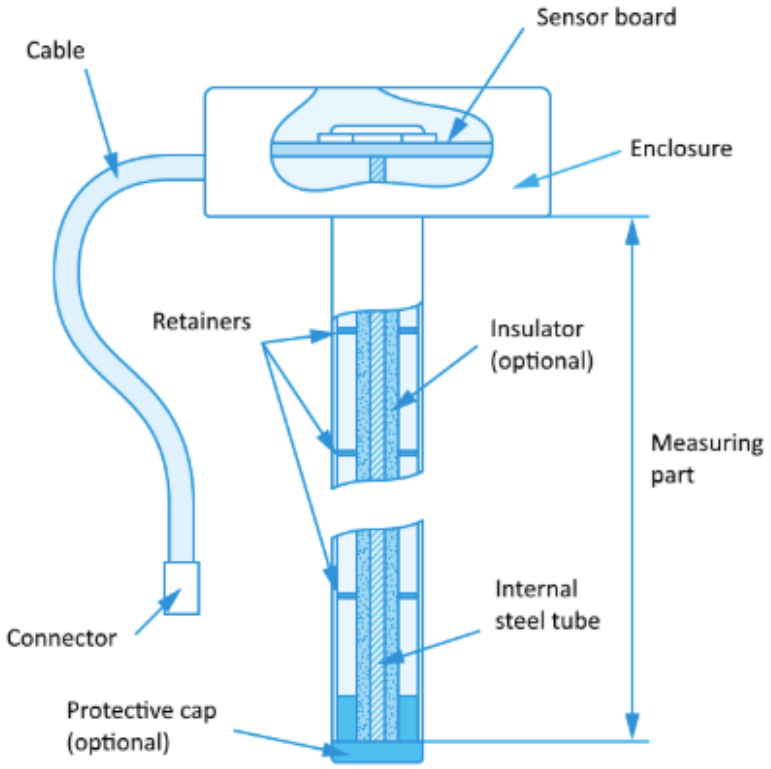


Рисунок 2.3. - Схема ємнісного датчика палива

3. Датчики GPS — це приймачі з антенами, які використовують супутникову навігаційну систему з мережею з 24 супутників на орбіті навколо Землі для надання інформації про місцезнаходження, швидкість та час.

5. Датчики вимірювання потужності двигуна. Система управління двигуном сучасного автомобіля складається з різноманітних електронних та електричних компонентів. Крім того, до них входять датчики двигуна, реле та виконавчі механізми. Вони забезпечують блок керування двигуном автомобіля життєво важливі параметри, необхідні для ефективного керування різними функціями двигуна.

Загалом, датчики двигуна - це електромеханічні пристрої, які контролюють різні параметри двигуна. Двигун використовує різні типи датчиків, таких як термопари, датчики температури опору (RTD) і датчики з ефектом Холла.

6. Розумні тахографи — це нове покоління обов'язкових бортових цифрових реєстраторів для забезпечення часу водіння та відпочинку професійних водіїв (соціальне регулювання). Нові функції повною мірою використовують передові цифрові технології, такі як супутникове позиціонування, зв'язок на короткій відстані для органів внутрішніх справ та взаємозв'язок з іншими телематичними додатками (такими як інтелектуальна парковка або додатки для оплати під час водіння), через інтерфейс інтелектуальної транспортної системи. Це дозволить автоматично записувати початок і кінцеве місце поїздок, а також дасть можливість віддаленого доступу до деяких даних тахографа через бездротову передачу даних до органів управління.

Завдяки використанню описаних вище пристроїв та датчиків стала можливою реалізація даної схеми.

2.2.1 Вибір глобальної системи позиціонування(GPS)

Глобальна система позиціонування (GPS) є однією з технологій, які в даний час використовуються організаціями для відстеження транспортних засобів, вантажівок, поїздів і кораблів у режимі реального часу. nПеревагами використання технології GPS для відстеження є: більш прозорий транспортний ланцюг, підвищення ефективності доставки за рахунок швидшого виявлення проблем, виявлення вузьких місць, більш надійні дані для планування, швидший час реакції у разі затримок та підвищення рівня задоволеності клієнтівї. Однак, коли приймачі GPS знаходяться в приміщенні, вони більше не можуть добре спілкуватися із супутниками для надання точної інформації про місцезнаходження (He et al., 2009). Підключивши системи GPS-картографування до систем планування ресурсів підприємства, це може полегшити планування вхідної та вихідної логістики. Використання цього призведе до безперебійної роботи та підвищеної надійності прийняття рішень щодо завантаження, збільшення транспортної пропускної здатності, оптимізації доставки замовлень у міських районах, автоматичних звітів про надзвичайні ситуації, динамічну маршрутизацію вантажів та підтримку прийняття рішень

щодо відправлення в режимі онлайн. Крім систем GPS-відстеження, існують також інші рішення, які використовуються організаціями для відстеження в реальному часі в мережі ланцюга поставок, що дає подібні переваги, наприклад, телематика і автоматична система ідентифікації.

2.2.2 Автоматична система ідентифікації транспортних засобів та телематичні дані про транспортні засоби

Автоматична система ідентифікації (AIS) дозволяє збирати інформацію про транспортні засоби. AIS автоматично надає інформацію про ідентифікацію автомобіля, тип, положення, курс, швидкість та навігаційний статус. Цю інформацію потім можна використовувати для покращення відстеження в ланцюгу поставок продукції. Інформація, зібрана з AIS, також все частіше використовується для відстеження впливу на навколишнє середовище.

Телематична система — це ПЗ, яке синхронізує дані з автомобільних датчиків, трекерів і інших мобільних пристроїв, і показує їх на екрані комп'ютера або ж смартфону, використовуючи спеціальні додатки.

Для телематичних систем транспортних засобів головною функцією є позиціонування та передачі даних, що дозволяє відстежувати та відстежувати маршрути, які проїхали водії, а також передавати інформацію, таку як інформація про замовлення, цифрові накладні на доставку або інформація про статус доставки. Другою основною частиною телематичних систем автомобіля є бортові системи моніторингу, які збирають дані транспортного засобу. Огляд вимірюваних параметрів наведено в таблиці 2.3.

Таблиця 2.3 - Параметри, що можливо виміряти бортовими системами моніторингу

Загальні дані	**Поведінка водія**	**Труднощі поїздки**	**Інспекція автомобіля**	**Дод. інформація**
1	2	3	4	5
Дата, час	Порушення поведінки	Брутто транспортного засобу	Рівень палива	Температура
Розміщення	Перемикання передач	Оцінка ваги	Рівень масла	Місце для завантаження

Продовження таблиці 2.3

Загальні дані	**Поведінка водія**	**Труднощі поїздки**	**Інспекція автомобіля**	**Дод. інформація**
1	2	3	4	5
Стан транспортного засобу; стан водія	Постійність середньої швидкості	Середній градієнт	Порушення звітності	
Час роботи/ відпочинку			Технічне обслуговування	
Зміна тахографа			Тиск у шинах	
Пройдена відстань				
Витрати палива				
Швидкість				

Розширене використання телематики транспортних засобів постачальниками логістики забезпечує більш ефективне управління автопарком, враховуючи технічне обслуговування транспортних засобів, діагностику транспортних засобів, керування водієм, керування паливом, управління здоров'ям та безпекою та динамічне планування транспортних засобів за допомогою спільної інформації в реальному часі .

2.2.3 Радіочастотна ідентифікація

Радіочастотна ідентифікація (RFID) – це технологія бездротового зв'язку, яка може зчитувати та записувати інформацію за допомогою радіосигналів без необхідності механічного чи оптичного контакту. RFID складається з трьох елементів: мітки, утвореної мікросхемою, яка підключена через антену; зчитувач, який надсилає радіосигнали та отримує дані та інформацію у відповідь від тегів, і проміжне програмне забезпечення, яке з'єднує апаратне забезпечення RFID із корпоративними програмам. Технологія RFID може забезпечити економічно ефективну систему для ідентифікації товарів та підключення об'єктів та обладнання до баз даних та мереж (Yan et al., 2018). Інформація, яка стає доступною завдяки застосуванню технологій RFID, має вирішальне значення для покращення роботи ланцюга поставок завдяки підвищенню видимості ланцюга

поставок та інтеграції між учасниками ланцюга поставок. RFID може забезпечити кілька переваг в управлінні ланцюгом поставок, таких як: покращення відстежуваності та видимості, підвищення ефективності, покращена точність інформації, зменшення втрат запасів та інформації в реальному часі.

2.2.4 Системи на основі камер

Збір даних за допомогою систем на основі камер надає корисну додаткову інформацію до існуючих методів збору даних. За допомогою алгоритмів обробки зображень можна реалізувати кілька функцій, починаючи від простого виявлення товарів і закінчуючи складним керуванням на основі імпульсів камери. Це виявлення та відстеження об'єктів здійснюється за допомогою чотирьох характеристик; Колір, краї, рух і текстура.

Системи на основі камер мають кілька сфер застосування, наприклад моніторинг зон зберігання, підрахунок об’єктів, що потрапляють у поле зору камери, визначення місця розташування об’єктів, відстеження шляху посилок і моніторинг стану завантаженості вантажопідйомних пристроїв. Спільною метою для всіх сфер застосування візуальних компонентів є підвищення прозорості, що часто супроводжується підвищенням ефективності. За допомогою системи на основі камери можна безперервно витягувати інформацію

2.2.5 Смартфон

Зростаюча популярність смартфонів із вбудованими функціями зв’язку та обробки демонструє все більший інтерес до досліджень. Дослідники виявили зростаючий інтерес до створення інтелектуальних рішень IoT за допомогою смартфонів через вбудовані датчики, такі як цифрові компаси, акселерометри, гіроскопи та GPS-системи. Ці датчики можуть забезпечувати такі функції, як зчитування коду швидкого реагування (QR), що залежить від смартфона, відстеження GPS в реальному часі, автоматичне сповіщення про реагування та багаторівневий контроль доступу до даних.

Датчики на базі смартфонів дають змогу створювати нові програми в різних сферах. Останнім часом з’явилися можливості для підвищення видимості всередині логістичних компаній, постачальників логістичних послуг, вантажовідправників і перевізників. Системи управління автопарком вже кілька

років використовують надійні портативні термінали, які є основною комунікаційною платформою в багатьох операціях великих операторів. Смартфони тепер відображають функціональність цих пристроїв і стають життєздатною альтернативою як пристрої збору даних у сфері логістики.

2.3. Вибір інструментів реалізації системи

Система логістики буде розроблена за допомогою веб-додатку. Веб-додаток — це система, яка дозволяє користувачам надсилати та отримувати дані до/з бази даних через Інтернет за допомогою свого веб-браузера, Крім того ще опираючись на [5], можна стверджувати, що веб-додаток, - це прикладна програма, яка працює у зв'язку з веб-сервером для виконання різноманітних завдань від простої до складної транзакції. З наведених вище визначень, веб-додаток можна визначити як систему, яка дозволяє користувачам виконувати певні функції за допомогою веб-браузера.

Щодо компонентів, то веб-додатки мають веб-сервер, сервер додатків, сервер баз даних, які інтегруються з одним одним та підключаються до Інтернету. На додаток веб-додаток має механізм MVC, який є моделлю (рівень даних), представлення (рівень презентації) і контролером (рівень керування). Крім цього означення, є твердження, що веб-додаток має трирівневу архітектуру, яка складається з логіки програми, презентації та керування даними, які розробляються та підтримуються на іншому рівні.

На прикладному рівні мова програмування — це та, яка виконує обчислення або інші логіки у веб-додатку. Дана реалізація система реалізується на мові програмування Ruby з використанням фреймворку Ruby on Rails та бази даних MySql.

2.3.1 Мова програмування Ruby та фреймворк Ruby on Rails

У рамках імплементації програмного забезпечення для розроблюваної системи, мною було обрано мову програмування Ruby та фреймворк Ruby on Rails. Ruby on Rails — це фреймворк, побудований на основі Ruby — мови програмування, створеної в 90-х роках. Ідея RoR проста — надати розробникам інтуїтивно зрозумілу структуру для швидкої розробки надійних, високопродуктивних веб-сторінок. Ruby on Rails відомий як повностекова

платформа MVC (модель-перегляд-контролер). Код розділений на три взаємопов'язані шари:

- Модель містить логіку програми, всі необхідні дані та класи високого рівня.
- Перегляд — це представлення в інтерфейсі користувача даних, присутніх у Modal. Це те, з чим користувачі взаємодіють і бачать на своїх екранах.
- Контролер з’єднує модель і вид, отримує введення від користувача та вирішує, як обробляти вхідні дані.

Саме з використанням даних технологій створення даної системи буде досить швидким, адже створення подібних систем саме з використанням цієї мови програмування забезпечує мінімальні витрати часу та швидкість і легкість розгортання обладнання.

2.3.2 Вибір бази даних

MySQL і PostgreSQL - дві провідні реляційні бази даних з відкритим кодом, які слугують задньою частиною незліченних комерційних, відкритих і внутрішніх програм.

Postgres (як це часто називають) і MySQL існують протягом тривалого часу. Обидва вони є безпечними СУБД з підтримкою кластеризації та відмовостійкістю мережі. Але незважаючи на все, що їх об’єднує, PostgreSQL і MySQL мають ряд характеристик, які відрізняють їх один від одного.

PostgreSQL — це об'єктно-реляційна база даних, а MySQL — суто реляційна. Це означає, що PostgreSQL пропонує більш складні типи даних і дозволяє об’єктам успадковувати властивості, але це також робить роботу з PostgreSQL більш складною.

PostgreSQL має єдиний ACID-сумісний механізм зберігання даних. MySQL підтримує 16 різних механізмів зберігання даних, які підходять для різних випадків використання. Механізм зберігання за замовчуванням, InnoDB, надає індексовані таблиці. PostgreSQL породжує новий системний процес із власним виділенням пам'яті для кожного клієнтського з'єднання, яке він встановлює, тому він вимагає багато пам'яті в системах з великою кількістю клієнтських з'єднань. MySQL використовує один процес і підтримує один потік (або шлях виконання) для кожного з'єднання, що добре працює для більшості додатків, обсяг яких не перевищує корпоративний.

Три поширені функції бази даних: представлення даних, тригери та збережені процедури. PostgreSQL має більш надійні уявлення та підтримує матеріалізовані уявлення, що може підвищити продуктивність для складних запитів.

У таблиці 2.4 наведено основні характеристики для порівняння баз даних POSTGRESQL та MYSQL.

Таблиця 2.4 Порівняння баз даних PostgreSQL та MySQL

Ознаки	PostgreSQL	MySQL
1	2	3
Архітектура	Об’єктно-реляційна, багато процесорна	Реляційна; єдиний процесор
Підтримувані типи даних	Числові, дата/час, символ, логічне значення, мережева адреса, JSON, XML, масиви, діапазони	Числові, дата/час, символи
Підтримувані індекси	B-дерево, хеш, GiST, SP-GiST, GIN, BRIN	B-дерево, R-дерево, хеш та інвестовані індекси для певних типів даних
Продуктивність	Підходить для програм з великим об‘ємом як читання так і запису	Підходить для додатків з великим обсягом зчитування
Безпека	Контроль доступу, кілька варіантів зашифрованого підключення	Контроль доступу, зашифровані з‘єднання
Підтримка	Підтримка спільноти; компанії, що мають власний випуск MySQL, можуть запропонувати підтримку.	Підтримка спільноти, а також контракти на підтримку, надані постачальниками.

Обидві бази даних підтримують тригери AFTER і BEFORE для операторів SQL INSERT, UPDATE і DELETE; PostgreSQL також пропонує тригер INSTEAD OF і може виконувати складні інструкції SQL у тригері за допомогою функцій. Обидві бази даних підтримують стандартні збережені процедури SQL, але PostgreSQL додатково пропонує можливість викликати процедури, написані іншими мовами, ніж SQL.

Отже, з проведеного порівняльного можна зробити висновок, що для реалізації даної системи з точки зору простоти імплементації та можливості

швидкого впровадження та взаємодії з мовою програмування Ruby було обрано базу даних MySql.

2.6 Висновок до розділу

У даному розділі було описано основні принципи проектування системи, також підібрано відповідні компоненти для реалізації технічної частини системи. Крім того, було описано роботу окремих пристроїв та датчиків, які потенційно можуть використовуватись у системі. Також було підібрано базу даних, яка задовільняє потреби розроблюваної системи.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Проектування архітектури бази даних

При розробці програмного забезпечення для розумної системи логістики важливу і одну з основних ролей відіграє коректна побудова архітектури бази даних для зберігання інформації про: наявні датчики, транспортні засоби, водіїв, зчитування та відповідно зберігання таких показників як температура всередині транспортного засобу, швидкість, робота двигуну та ефективність його роботи, тиск у шинах, активний та пасивний час роботи транспортного засобу.

На наявній діаграмі на рисунку 3.1 зображений перелік таблиць бази даних, що відповідатимуть за збереження даних. Отже, система буде містити такі таблиці як:

Таблиця Sensors (Датчики) з полями:

- sensor_id - ідентифікатор кожного датчику;
- sensor_type - тип кожного датчику;
- sensor_data - інформація, зчитувана з кожного датчику.

У таблиці sensors зберігатиметься інформація про кожен датчик та його тип, також зчитувана інформація з датчиків, потраплятиме у дану таблицю.

Таблиця Reads(Зчитування)

- read_id - id зчитування;
- sensor_id - id датчика.

У таблиці reads міститиметься інформація про кожне зчитування і id датчика , з якого відбулось зчитування інформації. Дана таблиця є вкрай важливою у побудові логіки роботи самої системи, так як у подальшому саме поле read_id у таблицях кожного з показників буде використовуватись як foreign_key.

Таблиця Temperatures(Температура)

- temperature_id - ідентифікатор кожного з даних температури;
- average_temp - середня температура за день;
- read_id - ідентифікатор зчитування.

У таблиці temperatures міститимуться дані про середні показники температури всередині автомобіля.

Таблиця Velocities(Швидкості)

- velocity_id - ідентифікатор кожного внесеного показника середньої швидкості;

- average_speed - середня швидкість за певний проміжок часу;
- read_id - id зчитування.

У таблиці velocities міститимуться дані про середні показники швидкості протягом дня конкретного транспортного засобу.

Таблиця Oil_Consumptions(Витрати палива)

- oil_consumption_id - ідентифікатор кожного внесеного показника середніх; витрат палива;
- average_consumption - показник середнього споживання палива;
- read_id - ідентифікатор зчитування.

У таблиці oil_consumptions міститимуться дані про середні показники споживання пального кожного дня для конкретного транспортного засобу.

Таблиця Engine Works (Робота двигуна)

- engine_work_id - ідентифікатор кожного внесеного показника роботи двигуна;
- average_power - середня потужність;
- average_level_heating - середня потужність нагрівання;
- read_id - ідентифікатор зчитування.

У таблиці engine_works міститимуться дані про середню потужність двигуну протягом дня, а також середні пооказники його нагрівання протягом дня.

Таблиця Tyre_Pressures(Тиск в Шинах)

- tyre_pressure_id - ідентифікатор кожного внесеного показника відсотку тиску в шинах;
- tyre_percentage - середній відсоток тиску в шинах;
- read_id - ідентифікатор зчитування.

У таблиці tyre_pressures міститимуться дані про відсоток тиску у шинах транспортного засобу за конкретний період.

Таблиця Time(Час)

- time_id - ідентифікатор кожного внесеного показника часу;
- working_hours - часи роботи тз;
- passive_hours - часи простою тз;
- read_id - ідентифікатор зчитування.

У таблиці times містимуться дані про активний та пасивний час транспортного засобу на дорозі.

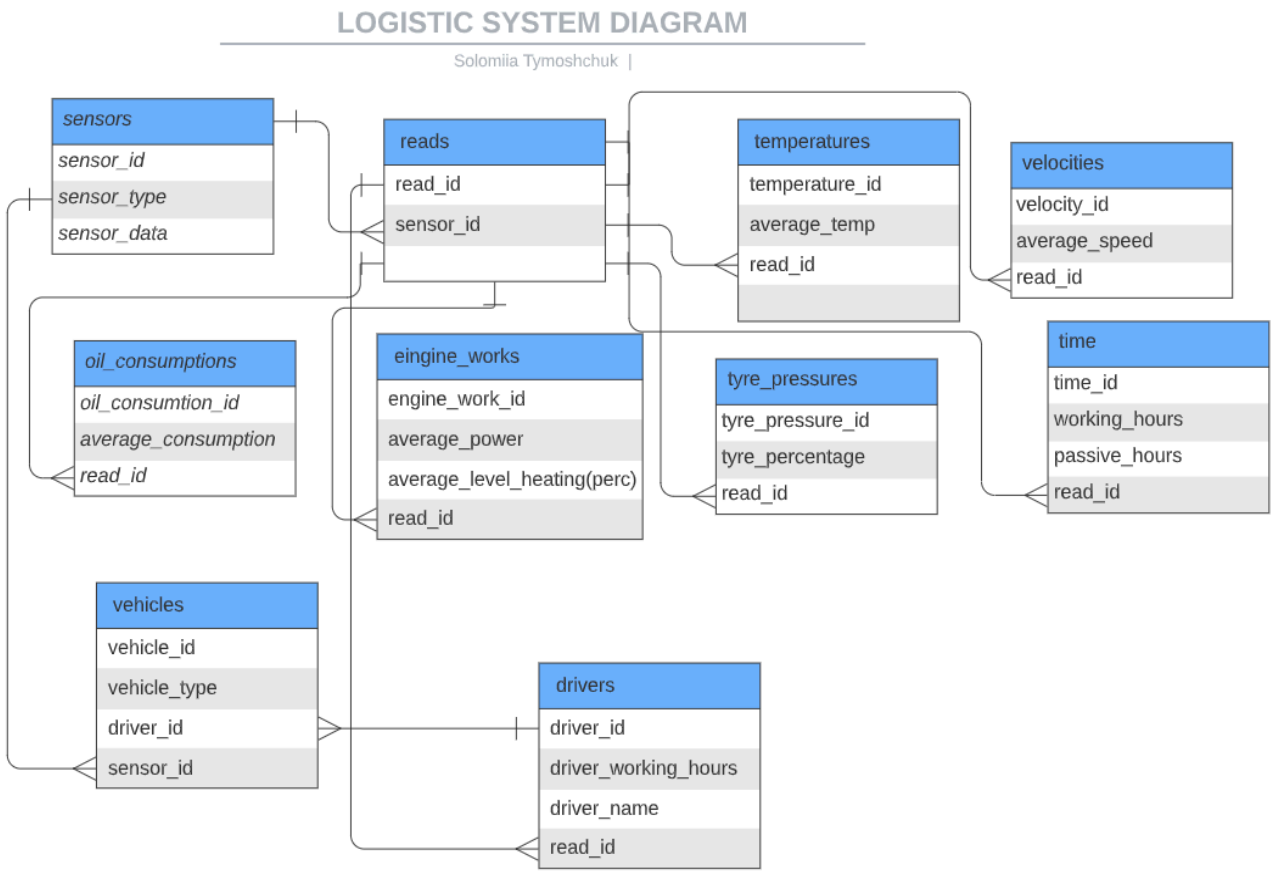


Рисунок 3.1 - Діаграма бази даних розроблюваної системи логістики доставки замовлень хлібопекарні

Таблиця Vehicles(Траспортні засоби)

- vehicle_id - ідентифікатор транспортного засобу;
- vehicle_type - тип тз;
- driver_id - id водія закріпленого за тз.

У таблиці vehicles міститеться інформація про транспортні засоби, а саме їх тип, id та id водія, закріпленого за конкретним транспортним засобом.

Таблиця Drivers(Водії)

- driver_id - ідентифікатор кожного водія;
- driver_working_hours - часи роботи водія;
- driver_name - ім’я водія;
- read_id - ідентифікатор зчитування(для передачі даних про часи роботи водія).

У таблиці drivers міститеться інформація про водіїв та час їх роботи.

База даних системи була розроблена з використанням вбудованого механізму у фреймворк Ruby on Rails.

На рис. 3.2 власне зображений сам файл міграції для додавання таблиці Sensors(Датчики) до схеми бази даних з відповідними полями.

Міграції — це зручний спосіб послідовної зміни схеми бази даних з часом. Вони використовують Ruby DSL, тому не потрібно писати SQL вручну, дозволяючи схемі та змінам бути незалежними від бази даних.

Можна розглядати кожну міграцію як нову «версію» бази даних. Схема починається з нічого, і кожна міграція змінює її, щоб додати або видалити таблиці, стовпці або записи. Active Record знає, як оновити схему на цій шкалі часу, перенісши її з будь-якої точки історії до останньої версії. Active Record також оновить db/schema.rb файл, щоб він відповідав оновленій структурі вашої бази даних.

Ця міграція на рис. 3.2 додає таблицю, temperatures з рядками average_temp, read_id. Викликаний стовпець первинного ключа id також буде додано неявно, оскільки це первинний ключ за замовчуванням для всіх моделей Active Record. Timestamps додає два стовпці created_atта updated_at. Ці спеціальні стовпці автоматично керуються Active Record, якщо вони існують.

```ruby
class CreateTemperatures < ActiveRecord::Migration[5.2]
  def change
    create_table :temperatures do |t|
      t.integer :average_temp
      t.integer :read_id
    end
    add_index :temperatures, :read_id
  end
end
```

Рисунок 3.2 - Зображення класу створення нової таблиці у бд, за допомогою міграції

Ми визначаємо зміни, які мають відбутися, рухаючись вперед у часі. Перед запуском цієї міграції таблиці не буде. Після цього таблиця буде існувати. Active Record також знає, як скасувати цю міграцію: якщо ми відкотимо цю міграцію, вона видалить таблицю.(команда rollback).

У базах даних, які підтримують транзакції з операторами, які змінюють схему, міграції загорнуті в транзакцію. Якщо база даних не підтримує це, тоді, коли міграція не вдається, її частини, які вдалися, не відкатуються. Вам доведеться відкочувати зміни, внесені вручну.

І на рис. 3.3 зображено схему після додавання нової таблиці за допомогою міграцій. Цей файл у фремворку Ruby on Rails має назву schema.rb. Він документує остаточний поточний стан схеми бази даних. Часто, особливо якщо у вас є кілька міграцій, складно вивести схему лише з міграцій. З schema.rb. Сам ActiveRecord справді не використовуватиме його. Він перевіряє базу даних під час виконання, оскільки це набагато безпечніше, ніж очікувати, що користувачі будуть оновлювати її у schema.rb. Однак, щоб уникнути плутанини розробників, завжди повинен підтримуватись файл, який оновлюється під час ваших міграцій.

```
create_table "temperatures", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "average_temp"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index column_name ["read_id"], name: "index_temperatures_on_read_id"
end
```

Рисунок 3.3 - Файл schema.rb

3.1.2 Зв’язки між моделями бази даних

Загалом у описаній вище інформації зустрічається поняття “Active Record”. Active Record — це M у MVC — модель — це рівень системи, відповідальний за представлення бізнес-даних та логіки. Active Record полегшує створення та використання бізнес-об'єктів, дані яких вимагають постійного зберігання в базі даних. [2]

Active Record дає кілька механізмів, найважливішим з яких є можливість:

- Представляти моделі та їх дані.
- Представляють асоціації між цими моделями.
- Представляти ієрархії успадкування через пов’язані моделі.
- Перевірте моделі, перш ніж вони будуть збережені в базі даних.
- Виконуйте операції з базою даних об’єктно-орієнтованим способом.

Отже, перелік моделей у базі даних такий: модель: OilConsumption, таблиця oil_consumptions, дана модель має зв’язок тільки з таблицею зчитування

під назвою reads і зв’язок носить назву has_one, що означає що один показник пального може мати лише одне зчитування. Temperature(модель наведена на рис. 3.5), таблиця temperatures, дана модель має зв’язок тільки з таблицею зчитування під назвою reads і зв’язок носить назву has_one, що означає що один показник пального може мати лише одне зчитування.

```
class Read < ApplicationRecord
  has_many :drivers
  belongs_to :sensor
  has_many :temperatures
  has_many :velocities
  has_many :oil_consumptions
  has_many :engine_works
  has_many :tyre_pressures
  has_many :times
  has_many :drivers
end
```

Рисунок 3.4 - Модель Read

EngineWork, таблиця engone_works, дана модель має зв’язок тільки з таблицею зчитування під назвою reads і зв’язок носить назву has_one, що означає що один показник пального може мати лише одне зчитування. Velocities, таблиця engone_works, дана модель має зв’язок тільки з таблицею зчитування під назвою reads і зв’язок носить назву has_one, що означає що один показник пального може мати лише одне зчитування. Таблиця Sensors натомість може мати багато зчитувань, що обумовлює зв’язок has_many. А модель Read має відношення belongs_to з таблицею sensors, і зворотній зв’язок з таблицями temperatures, velocities, oil_consumptions, engine_works, tyre_pressures, times, drivers.

На рис. 3.4 зображена модель Read , яка наслідує ApplicationRecord. Активний запис — це M у MVC — модель — це рівень системи, відповідальний за представлення бізнес-даних та логіки. Active Record полегшує створення та використання бізнес-об'єктів, дані яких вимагають постійного зберігання в базі даних. Це реалізація шаблону Active Record, який сам по собі є описом системи реляційного відображення об'єктів.

Active Record був описаний Мартіном Фаулером у його книзі *Шаблони архітектури корпоративних додатків* . У Active Record об’єкти несуть як постійні дані, так і поведінку, яка оперує цими даними. Active Record вважає, що забезпечення логіки доступу до даних як частини об’єкта навчить користувачів цього об’єкта тому, як записувати та читати з бази даних.

Реляційне відображення об’єктів , яке зазвичай називають його абревіатурою ORM, — це техніка, яка з’єднує багаті об’єкти програми з таблицями в системі керування реляційною базою даних. Використовуючи ORM, властивості та зв’язки об’єктів у програмі можна легко зберігати та витягувати з бази даних без безпосереднього написання операторів SQL і з меншим загальним кодом доступу до бази даних.

```ruby
class Temperature < ApplicationRecord
  has_one name :read

  scope :high_temp, -> { where("average_temp >= 25") }
end
```

Рисунок 3.5 - Модель Temperature

3.2 Розробка інтерфейсу системи

Інтерфейс до даної системи як і було передбачено у плануванні системи, реалізовується на мові програмування Ruby та з використанням фреймворку Ruby on Rails. Фронтенд частина доповнюється мовою розмітки гіпертексту HTML та каскадними таблицями стилів CSS.

3.2.1 Реалізація контролерів

Для кожної моделі, наявної у розроблюваній аплікації створюється контролер, який відповідатиме за бізнес-логіку програми.

Крім того, контролер Rails є логічним центром програми. Він координує взаємодію між користувачем, самим інтерфейсом користувача(тобто фронтенд частиною або тз view) та моделлю. Контролер містить у собі ряд важливих допоміжних служб.

- Він відповідає за маршрутизацію зовнішніх запитів до внутрішніх дій. Він надзвичайно добре обробляє зручні для людей URL-адреси.
- Він керує кешуванням, що може підвищити продуктивність програми.

- Він керує допоміжними модулями, які розширюють можливості шаблонів перегляду, не збільшуючи їх код.
- Він керує сеансами, створюючи у користувачів враження постійної взаємодії з нашими програмами.

Процес створення контролера дуже простий і схожий на процес, який вже використовувався для створення моделі. Наприклад, для створення контролера моделі read використовується така команда з терміналу проекту:

`rails generate controller Read`. На рис. 3.6 можна спостерігати створений клас контролера. У ньому містяться такі методи як index, show, new, edit, create, update, destroy, delete. Кожен з цих методів дає змогу виконувати певні дії над даними бази даних через взаємодію з view та власне самими таблицями бази даних.

```ruby
class ReadsController < ApplicationController
  before_action :set_read, only: %i[ show edit update destroy ]

  # GET /reads or /reads.json
  def index ... end

  # GET /reads/1 or /reads/1.json
  def show ... end

  # GET /reads/new
  def new ... end

  # GET /reads/1/edit
  def edit ... end

  # POST /reads or /reads.json
  def create
    @read = Read.new(read_params)

    respond_to do |format|
      if @read.save
        format.html { redirect_to @read, notice: "read was successfully created." }
        format.json { render :show, status: :created, location: @read }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @read.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /reads/1 or /reads/1.json
  def update
    respond_to do |format|
      if @read.update(read_params)
        format.html { redirect_to @read, notice: "read was successfully updated." }
        format.json { render :show, status: :ok, location: @read }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @read.errors, status: :unprocessable_entity }
      end
    end
```

Рисунок 3.6 - Контролер до моделі Read

3.2.2 Розробка користувацького додатку системи

Спочатку створюється домашня сторінка системи під назвою index.html.erb, у яку і поміщаємо базову інформацію, наведену на рис. 3.7.

Рисунок 3.7 - Головна сторінка системи

Також на цій сторінці міститься панель навігації, що дає можливість переходу між кожною з сторінок. Крім того, на домашній сторінці було наведено схему розміщення датчиків на транспортному засобі підприємства для кращого розуміння, з яких пристроїв зчитуватимуться та передаватимуться показники необхідні для логістичної системи підприємства.

На прикладі сторінки датчиків(Sensors) можна побачити, що система дає змогу додавати, редагувати та видаляти дані. На рис. 3.8 зображено контейнер, у якому видно додані у систему девайси, власне ці самі датчики містяться у таблицях бази даних, описаних вище.

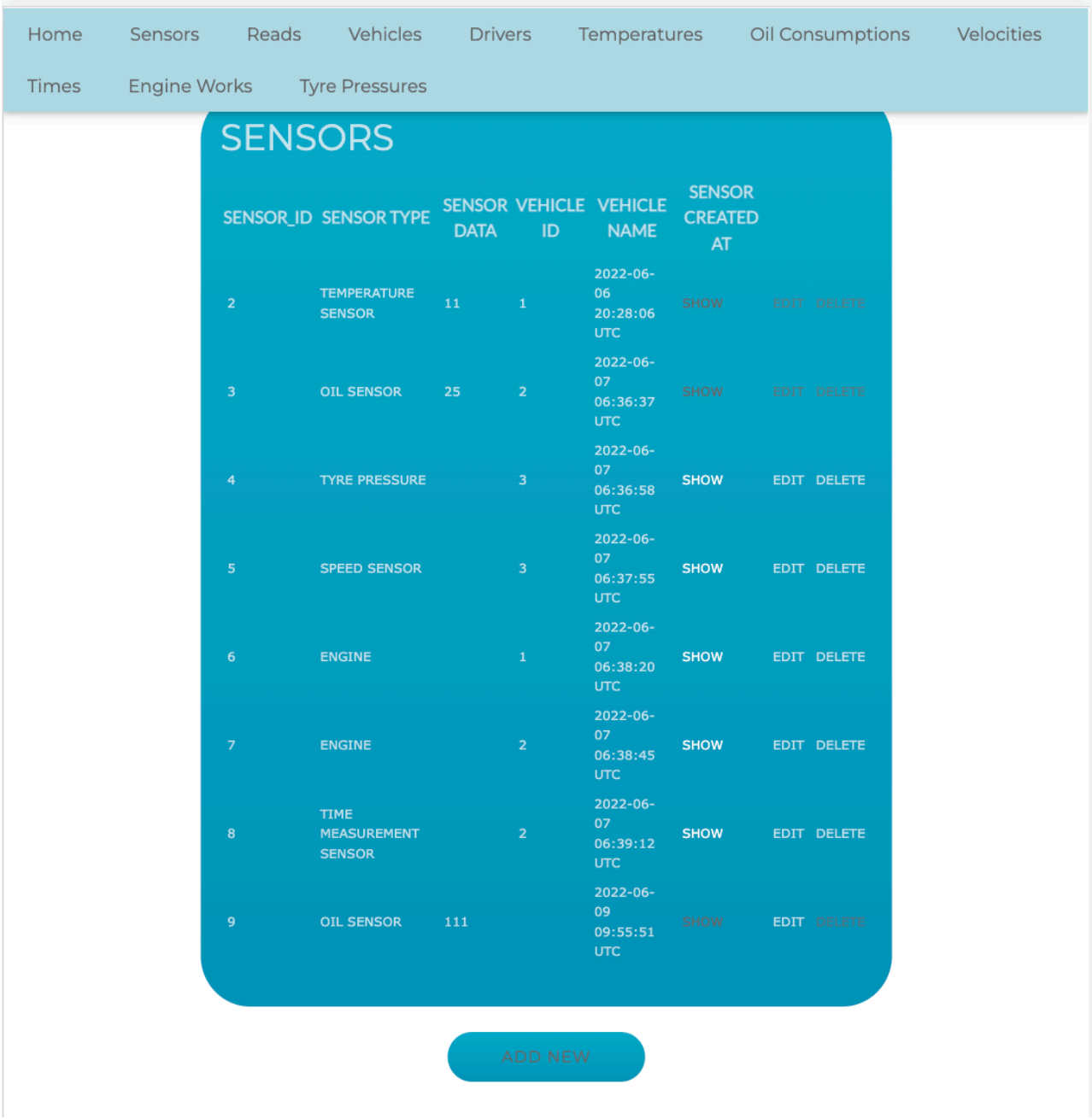


Рисунок 3.8 - Таблиця датчиків, які внесені у систему

Крім цього, на даній сторінці(рис. 3.9) наведена інфографіка стосовно того, які конкретні датчики були внесені у систему. Дана інфографіка є динамічною, тобто при кожному внесенні нових даних до таблиці, відсоток певних датчиків буде змінюватись, що дасть змогу аналізувати пристрої. Такий тип візуалізації можна застосовувати і до інших типів даних при необхідності, при цьому здійснюючи сортування бажаним чином. На рис. 3.9 інфографіка кількості датчиків у системі по їх типу.

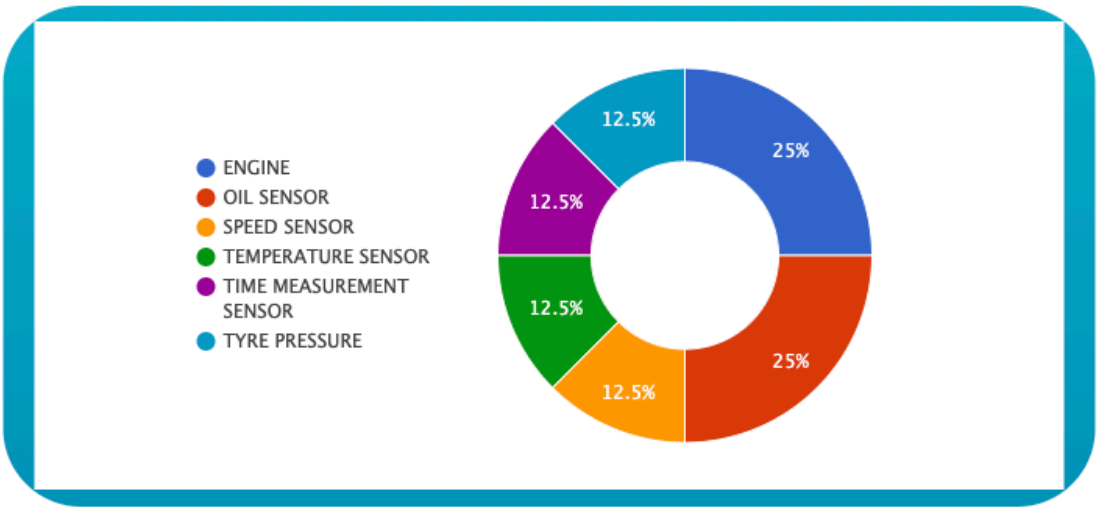


Рисунок 3.9 - Інфографіка кількості датчиків по їх типу у відсотковому співвідношенні

На рис. 3.10 зображено зміну відсотків датчиків при додаванні датчику типу oilsensor.

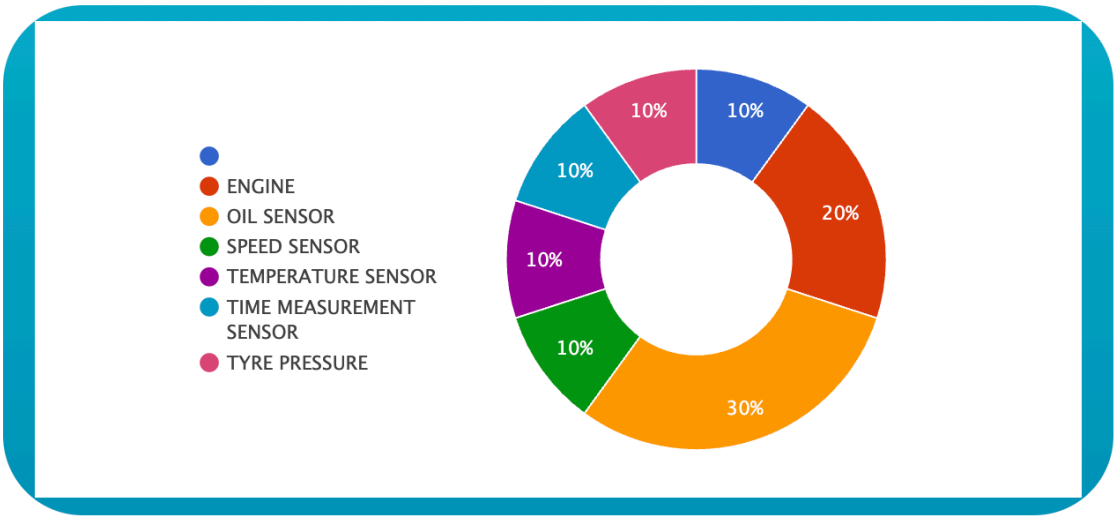


Рисунок 3.10 - Оновлена інфографіка кількості датчиків по їх типу у відсотковому співвідношенні

Для додавання нового датчика у систему потрібно перейти на сторінку створення нових датчиків під назвою маршруту sensors/new. На рис. 3.11 видно поля, які необхідно заповнити для додавання датчику у систему, ці поля відповідають полям, наявним у базі даних таблиці sensors: sensor_type, sensor_name, sensor_date та veehicle, що відповідає полю vehicle_id.

Також на цій сторінці є наявний список усіх автомобілів підприємства з ім’ям водія та назвою конкретного автомобіля.

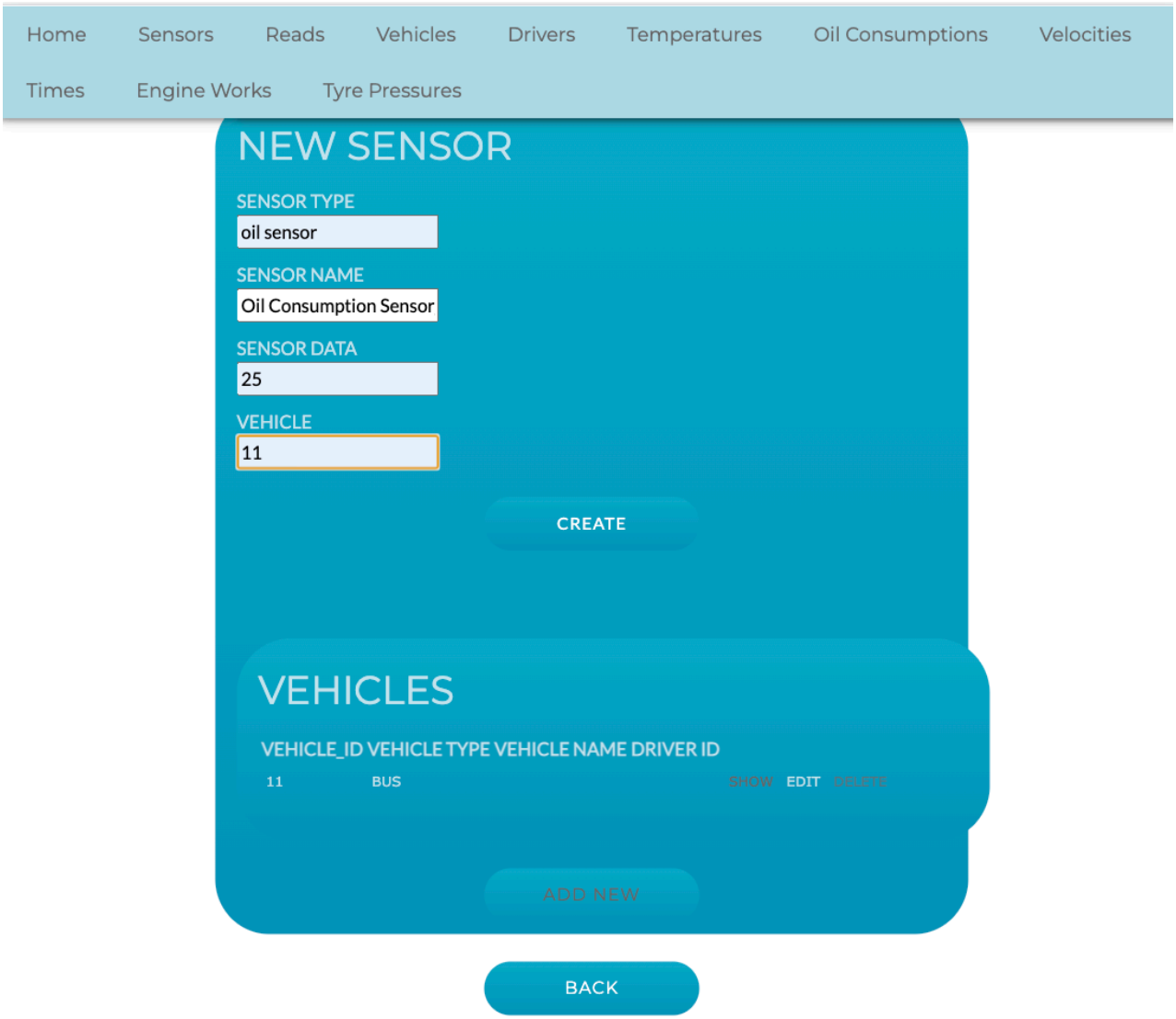


Рисунок 3.11 - Сторінка додавання нового датчику

Крім того, для кожного добавленого рядку у базу даних системи доступний перегляд, редагування та видалення рядку. На рис. 3.12 можна побачити id сенсору, його тип та чисельну інформацію про нього.

Рисунок 3.12 - Сторінка перегляду датчику

Наступною сторінкою системи є сторінка зчитувань, у якій міститься інформація про кожне зчитування, а також інформація про id конкретного датчику та його тип для кращого розуміння інформації користувачем системи.

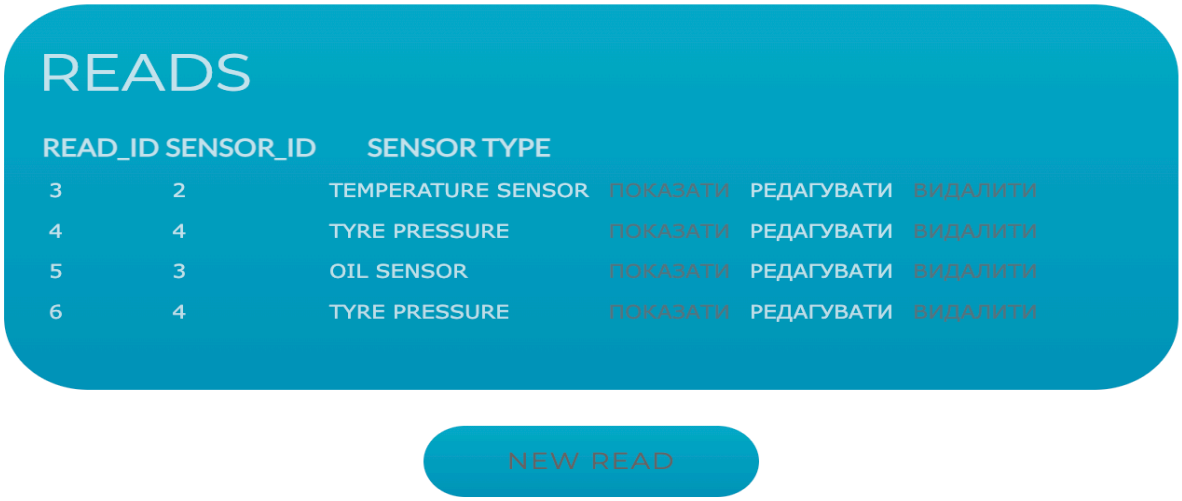


Рисунок 3.13 - Сторінка перегляду усіх зчитувань

З вкладки системи зчитування є можливість вносити нові зчитування та обирати зі списку наявні датчики у системі по їх назві, що значно спрощує роботу з самою системою. Це можна спостерігати на рис. 3.14.

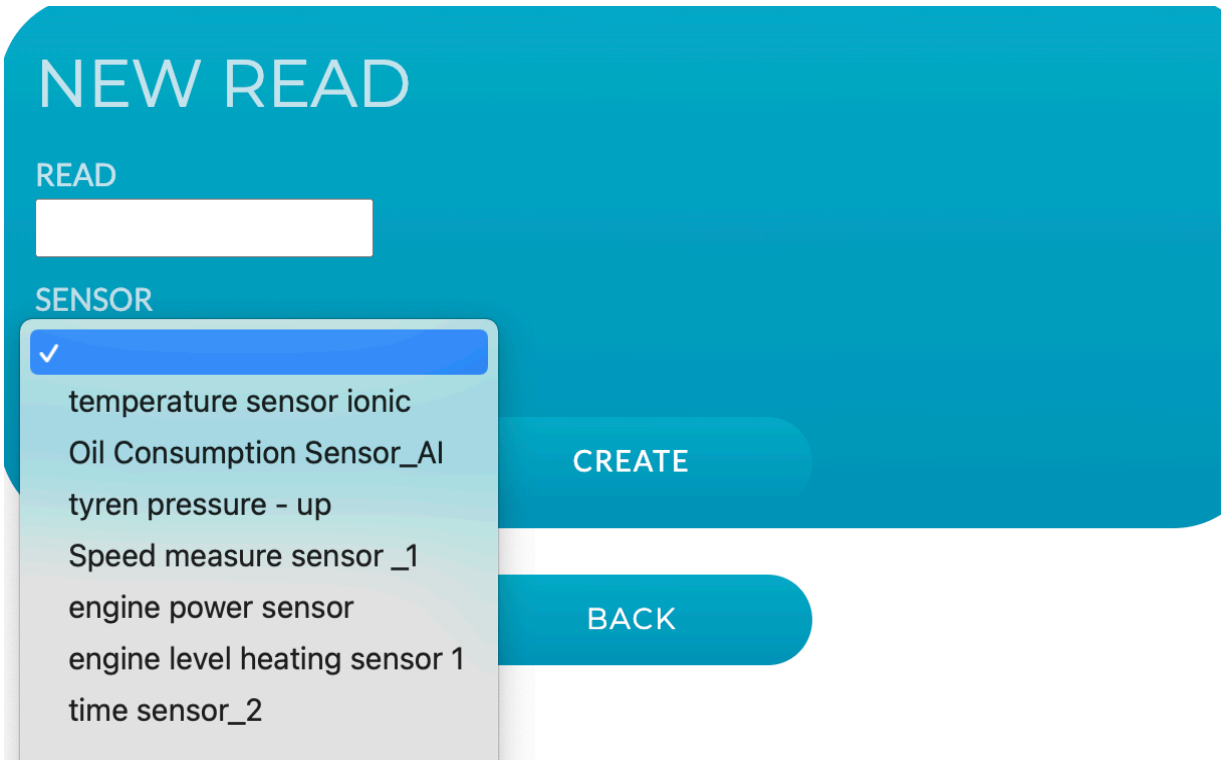


Рисунок 3.14 - Сторінка додавання нових зчитувань

Після додавання як і для таблиці датчики, для зчитувань можна побачити дані по доданому зчитуванню з конкретним id зчитування, id датчику та його назвою.

Рисунок 3.15 - Сторінка перегляду конкретного зчитування

Система також дає можливості редагування та видалення даних і перед видаленням користувач отримає попередження чи дійсно хоче видалити певний запис.

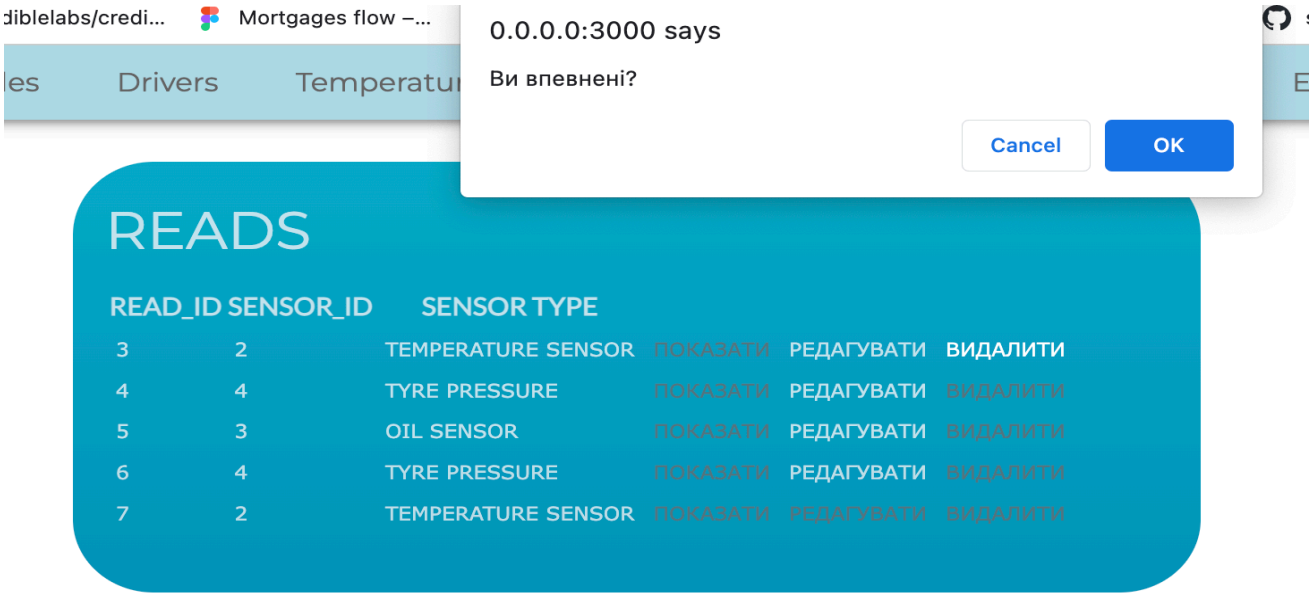


Рисунок 3.16 - Сторінка видалення конкретного зчитування

У даній системі передбачені сторінки показників кожного з зчитувань. На рис. 3.17 зображені показники зчитувань температур за місяць червень. По полю у таблиці created_at можна спостерігати, коли дана температура була зчитана.

Для даної таблиці вкрай важливою є можливість візуалізації зчитуваних даних, тобто створення конкретної інфографіки для даних. У загальному різні типи інфографіки полегшують роботу з даними та дають можливість у подальшому розуміти як покращувати систему логістики, роботу працівників, виявляти збої та корегувати недоліки у самих пристроях та у системі. [5] Тому завдяки бібліотеці Chartkick, реалізованій на мові програмування Ruby стало можливим відтворювати інформацію про зчитування, конкретні показники середньої температури на графіку.

Отже, на рис. 3.18 можна спостерігати лінійний графік середніх показників температури за червень.

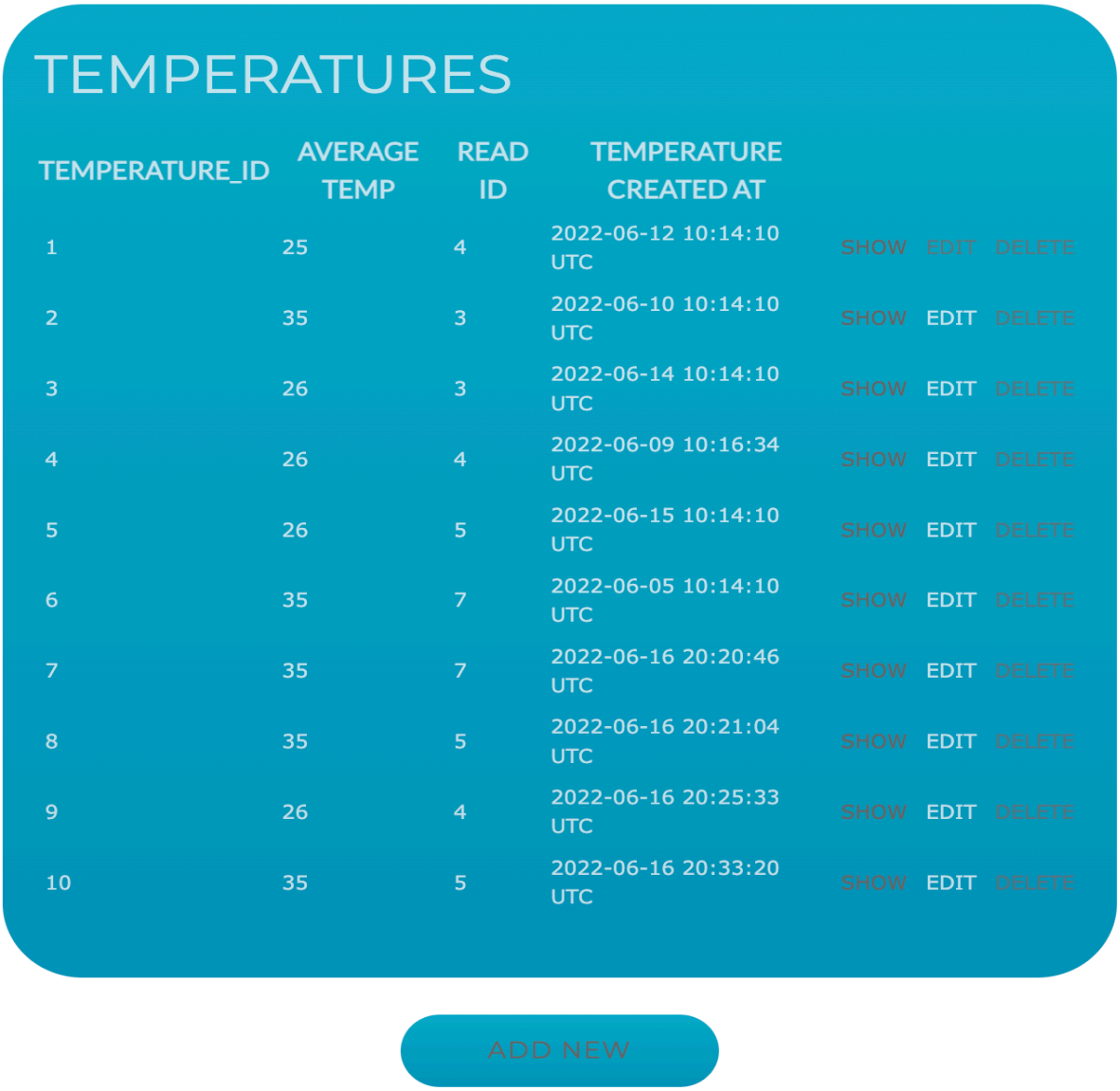


Рисунок 3.17 - Сторінка перегляду усіх показників температур

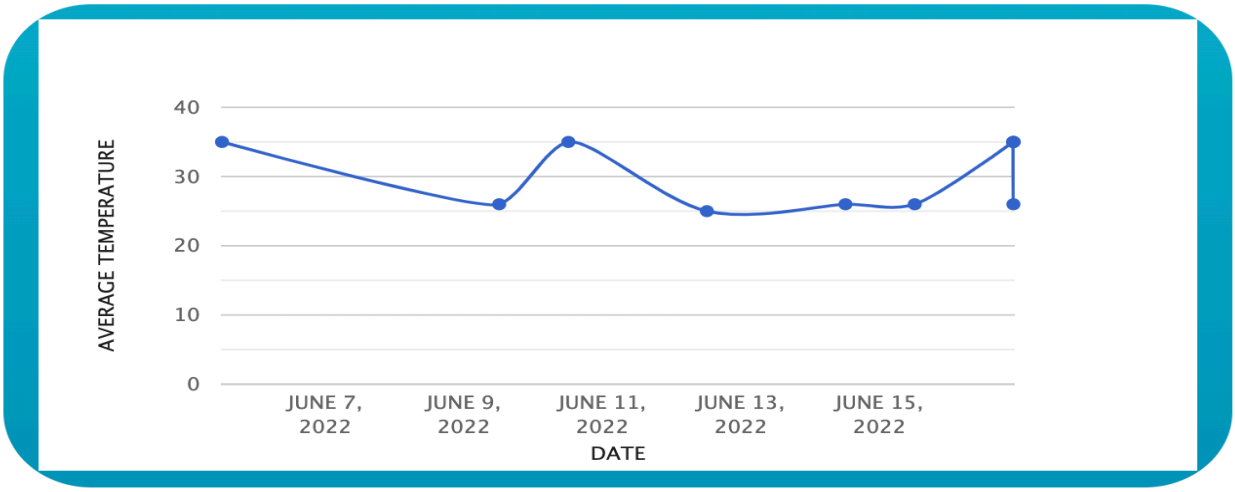


Рисунок 3.18 - Інфографіка усіх показників температур

Також одними з важливих показників є середні показники витрат пального. Дані показники можна спостерігати на рис. 3.19.

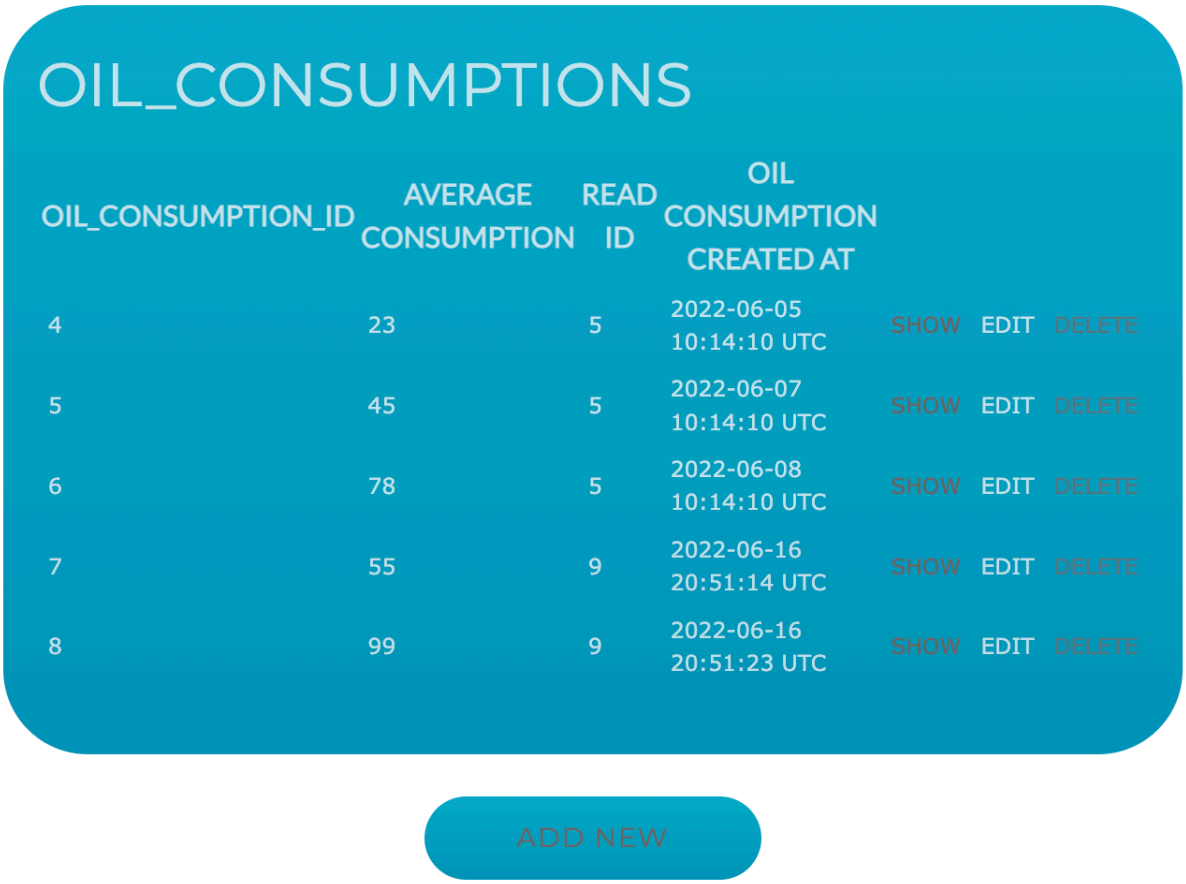


Рисунок 3.19 - Сторінка перегляду усіх показників середніх витрат пального

Крім того, на даній сторінці можна побачити інфографіку щодо витрат пального за конкретні дні у місяці червень. Зокрема з даної діаграми на рис. 3.20 можна зробити висновок, що 16.06 була найбільша витрата палива, що сягала близько 100 літрів. На даному етапі інформація додана у система є тестовою і може дещо відрізнятись від реальних показників. Наступною сторінкою з важливими показниками є сторінка показників роботи двигуна, на якій можна побачити дані про середній рівень нагрівання двигуну та його середню потужність. Ці показники відіграють вагому роль у слідкуванні за коректною роботою транспортних засобів так як двигун є одною із основних складових конкретного транспортного засобу.

Майже вся енергія, яка використовується для транспортування та електроенергії, надходить від теплових двигунів. Гарячі предмети, навіть гази , мають теплову енергію, яку можна перетворити на щось корисне. Теплові двигуни переміщують енергію з гарячого місця в холодне і відволікають частину

цієї енергії в механічну. Для функціонування теплових двигунів потрібна різниця температур .

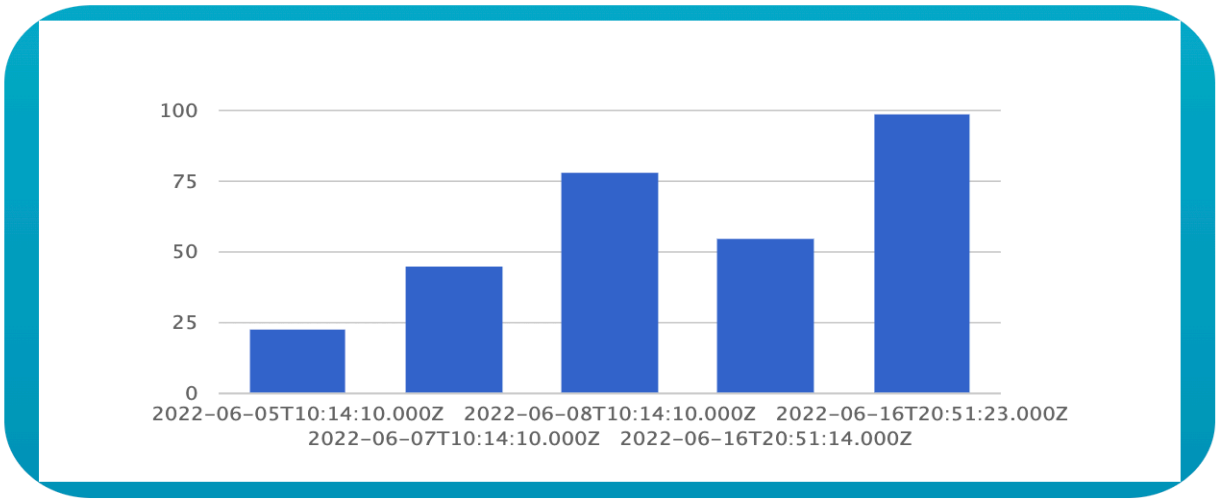


Рисунок 3.20 - Діаграма витрат пального за конкретні періоди часу

Тепловий двигун має два побічні продукти: роботу і тепло. Призначення більшості двигунів - виробляти роботу, а тепло обробляється просто як відходи . Когенерація використовує відпрацьоване тепло для корисних речей. Обігрівач в автомобілі працює за допомогою когенерації - відбирає відпрацьоване тепло від двигуна для нагрівання повітря, яке прогріває салон. Ось чому запуск автомобільного обігрівача взимку мало впливає на витрату газу, але запуск кондиціонера влітку може коштувати приблизно 10-20% від витрати бензину автомобіля.

На рис. 3.21 можна спостерігати список вимірювань показників з роботи двигуна, а на рис. 3.22 наведена відсоткова різниця між рівнем нагрівання двигуна та його середньою потужністю протягом дня.

Зробивши оглядовий аналіз даної кругової діаграми можна зробити висновок, що найбільшою різниця у відсотках між рівнем нагрівання двигуну та його потужністю була 16.06, коли рівень середньої потужності дорівнював 34.

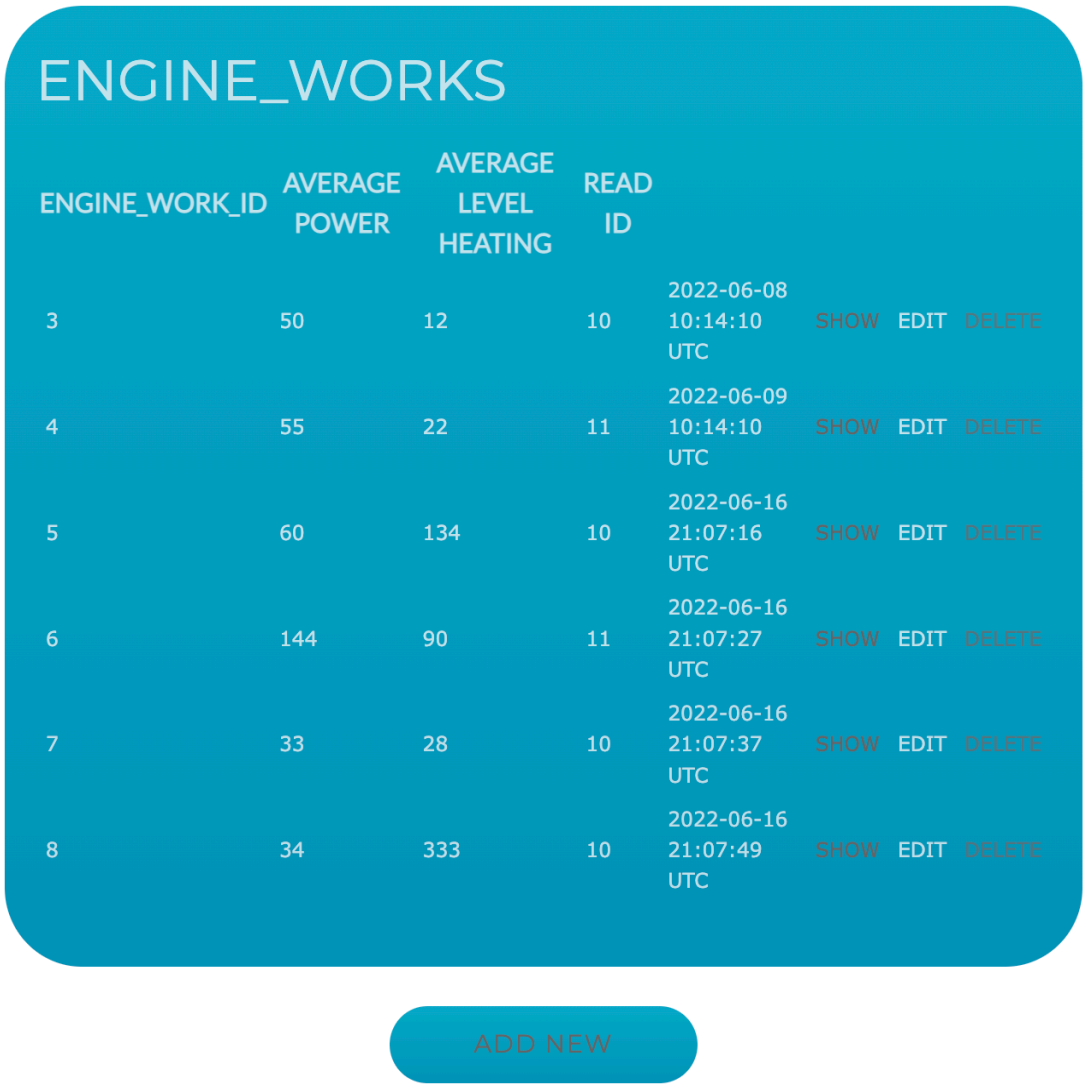


Рисунок 3.21 - Сторінка перегляду даних про роботу двигуна

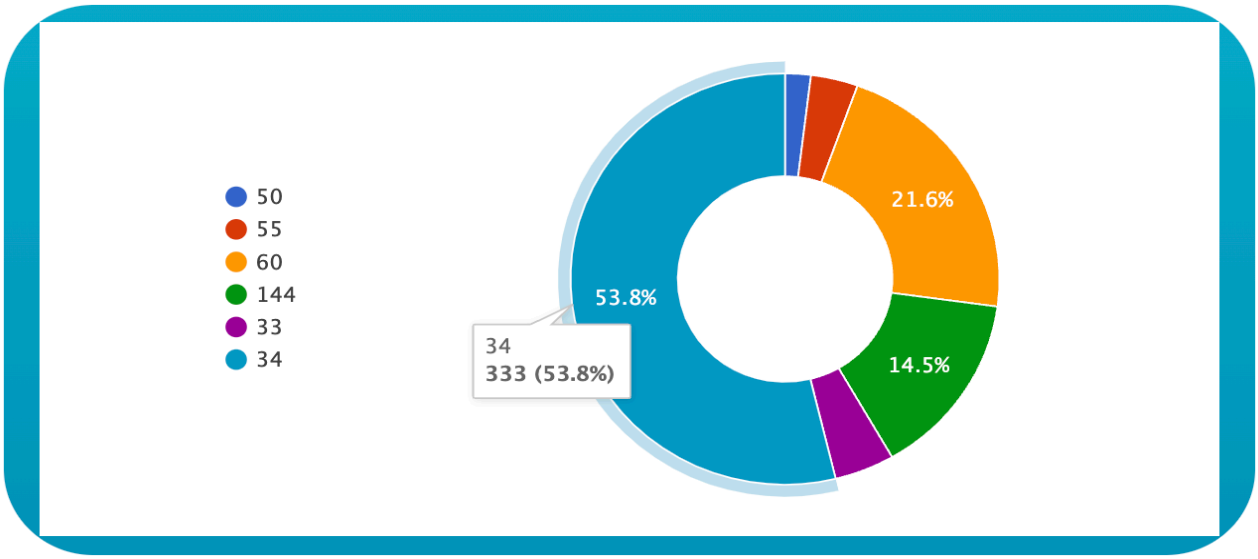


Рисунок 3.22 - Кругова діаграма відсоткового співвідношення середньої потужності двигуну до рівня його нагрівання

3.3. Тестування системи

3.3.1 Тестування інтерфейсу системи

Для даної системи важливим етапом є її тестування, а саме тестування окремих компонентів та їх взаємодії між собою.

Веб-тестування або тестування веб-додатків — це практика програмного забезпечення, яка забезпечує якість, перевіряючи, чи функціональність даного веб-додатка працює належним чином або відповідно до вимог. Веб-тестування дозволяє знаходити помилки в будь-який момент, до випуску або щодня.

Тестування є дуже важливою частиною розробки програмного забезпечення. Всякий раз, коли в коді вносяться зміни, незалежно від того, наскільки вони незначні, помилки можуть з'явитися в іншому місці системи. Вартість виправлення цих помилок зростає, чим пізніше вони будуть виявлені в процесі розробки. Наявність ефективного веб-тестування може запобігти цим додатковим витратам.

Для тестування даної системи будуть застосовані такі види тестування як:

Функціональне тестування використовується для того, щоб переконатися, що функціональність програмного забезпечення працює так, як призначено для кінцевого користувача.

Регресійне тестування можна описати як «повторне функціональне тестування». Він використовується, щоб переконатися, що функціональність програмного забезпечення продовжує працювати після того, як його частини були змінені новим кодом або конфігураціями. Наприклад, коли створюються нові функції, регресійне тестування гарантує, що старі функції програмного забезпечення продовжують працювати належним чином.

Тестування продуктивності, таке як стресове та навантажувальне тестування, гарантує, що веб-додаток витримує тривалі періоди активності або пікове навантаження користувачів. Досягнення необхідних умов стресу або рівня навантаження неможливе, якщо це робити вручну, тому автоматизація є ключем до підтвердження того, що ваша програма може працювати в будь-якій ситуації.

3.3.2 Тестування IOT пристроїв

Тестування IOT – це тип тестування для перевірки пристроїв IOT . Сьогодні зростає потреба в наданні кращих і швидших послуг. Існує величезний попит на доступ, створення, використання та обмін даними з будь-якого пристрою. Мета полягає в тому, щоб забезпечити кращу розуміння та контроль над різними взаємопов'язаними пристроями IOT. Отже, структура тестування IOT є важливою та етапи тестування наведені на рис. 3.23.

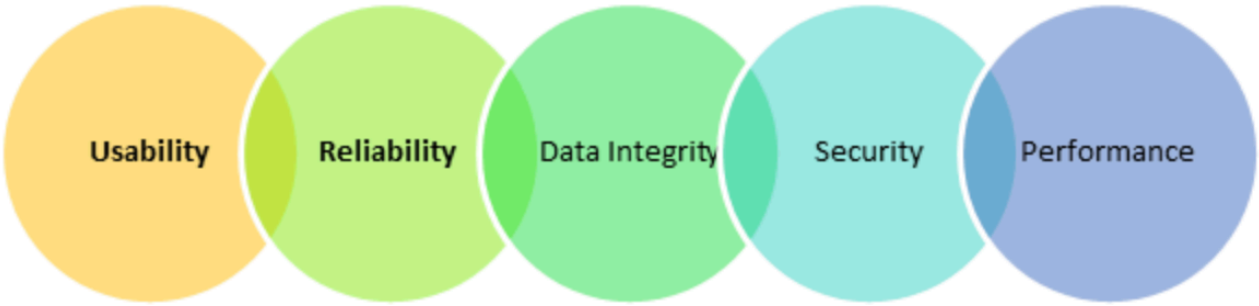


Рисунок 3.23 - Види тестування IOT пристроїв

Тестування пристроїв Інтернету речей пов'язане з безпекою, аналітикою, пристроєм, мережами, процесорами, операційними системами, платформами та стандартами.

Тестування юзабіліті:

Користувачі використовують дуже багато пристроїв різної форми та форм-факторів. Крім того, сприйняття також різниться від одного користувача до іншого. Тому перевірка зручності використання системи дуже важлива в тестуванні IoT.

Тестування на сумісність:

Існує багато пристроїв, які можна підключити через систему IOT. Ці пристрої мають різну конфігурацію програмного та апаратного забезпечення. Тому можливі комбінації величезні. Тому перевірка сумісності в системі IOT є важливою. [21]

Тестування надійності та масштабованості:

Надійність і масштабованість важливі для створення тестового середовища IOT, яке включає моделювання датчиків за допомогою інструментів і технологій віртуалізації.

Перевірка цілісності даних:

Важливо перевірити цілісність даних під час тестування IOT, оскільки воно включає велику кількість даних та їх застосування.

Перевірка безпеки:

У середовищі IOT багато користувачів мають доступ до величезної кількості даних. Таким чином, важливо підтвердити користувача за допомогою аутентифікації, мати контроль конфіденційності даних як частину тестування безпеки.

Тестування продуктивності:

Тестування продуктивності важливо для створення стратегічного підходу до розробки та впровадження плану тестування IOT.

У таблиці 3.1 наведено приклади тестування для IoT пристроїв.

Таблиця 3.1 - Тестування IOT пристроїв

Тестові категорії	Зразок умов випробування
1	2
Перевірка компонентів	- апаратне забезпечення пристрою; - вбудоване програмне забезпечення; - хмарна інфраструктура; - підключення до мережі; - стороннє програмне забезпечення; - тестування датчиків; - тестуванн команд; - тестування формату даних; - випробовування на міцність; - тестування безпеки.
Перевірка функції:	- основне тестування пристрою; - тестування між пристроями IOT; - обробка помилок; - дійсний розрахунок.
Перевірка умов:	- ручне кондиціонування; - автоматичне кондиціонування; - профілі кондиціонування.

Продовження таблиці 3.1

Тестові категорії	Зразок умов випробування
1	2
Перевірка продуктивності:	- частота передачі даних; - передача кількох запитів; - синхронізація; - тестування переривання; - продуктивність пристрою; - перевірка узгодження.
Безпека та перевірка даних:	- перевірка пакетів даних; - перевірка втрати даних чи пошкодження пакетів; - шифрування/дешифрування даних; - значення даних; - ролі та відповідальність користувачів та їх модель використання.
Перевірка шлюзу:	- тестування хмарного інтерфейсу; - тестування протоколу від пристрою до хмари; - тестування затримки.
Перевірка аналітики:	- перевірка аналізу даних датчиків ; - операційна аналітика системи IOT; - аналітика системного фільтра; - перевірка правил.
Перевірка комунікації:	- сумісність(- M2M; - тестування трансляції; - тестування переривань; - протокол.

Інструменти тестування IOT:

Два найбільш ефективні інструменти тестування IOT:

1.Шодан.

Shodan – це інструмент тестування IOT, який можна використовувати, щоб дізнатися, які з ваших пристроїв підключені до Інтернету. Це дозволяє відстежувати всі комп’ютери, які є безпосередньо доступними з Інтернету.

2. Речовий.

Thingful — це пошукова система для Інтернету речей. Це забезпечує безпечну взаємодію між мільйонами об’єктів через Інтернет. Цей інструмент

тестування IOT також контролює використання даних і дає змогу приймати більш рішучі та цінні рішення.

3.4 Висновок до розділу

У даному розділі було детально описано ключові інструменти, використовувані у розробці інтерфейсу системи, також було продемонстровано саму систему з розробленими та створеними таблицями бази даних. На завершення, будо описано можливі варіанти тестування системи та IOT пристроїв.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було реалізовано систему логістики замовлень хлібозаводу “Кулиничі” з використанням IOT технологій. Також було зроблено дослідження ефективності впровадження IOT у системі логістики хлібзаводу, зроблена оцінка та аналіз різних підходів до реалізації даної системи.

Як результат мною було змодельовано схему розміщення датчиків зчитування відповідних даних та окремих пристроїв на транспорті підприємства. Крім того, було розроблено схему таблиць бази даних, у яких передбачено записування інформації, зчитуваної з датчиків. За допомогою використання мови програмування Ruby та фреймворку Ruby on Rails, бази даних MySQL вдалось створити інтерфейс системи з можливістю додавання, редагування та видалення даних.

Було з’ясовано потенційні можливості застосування системи, не тільки на базі хлібозаводу і було виявлено, що розроблювальна система є універсальною, адже може використовуватись на різних типах підприємств, де є потреби у ефективній логістиці та доставці.

Для досягнення ефективності у розробці мною було розглянуто декілька аналогів та проаналізовано переваги подібних систем саме у застосуванні IOT рішень у логістичних цілях. Було визначено, що такі системи здебільшого є стартапами з великими шансами росту та розвитку, тому дана сфера є досить перспективною.

Було проведено аналіз технологій, можливих для використання у реалізації програмного забезпечення системи. До цього ж було зроблено порівняльний аналіз наявних баз даних для підключення до інтерфейсу системи для зберігання даних, зчитуваних з датчиків.

Було підібрано відповідні пристрої, а саме перелік датчиків, які є найбільш доцільними та вдалими до використання на транспортному засобі підприємства. У окремій таблиці було зроблено описовий аналіз кожного з них. До відібраних датчиків входять: датчики глобальної системи позиціонування(GPS), датчики зчитування тиску в шинах, датчик для вимірювання рівня споживання пального,

датчики для вимірювання потужності двигуна, розумний тахограф для вимірювання швидкості та передні та задні камери.

Програмно був реалізований інтерфейс до системи, включаючи сторінки додавання, редагування та видалення даних, що дозволяє користувачам ефективно взаємодіяти із системою. До цього ж було зроблено візуальний аналіз даних за допомогою графіків та діаграм, що у майбутньому допоможе ефективніше використовувати ресурси підприємства в результаті якісного аналізу різних типів даних, наявних у системі.

Ключовими функціями системи є можливість додавання, редагування, видалення даних; створення інфографіки по отримуваним даним; крім того, передбачено, що датчики використовувані у системі відповідатимуть за підрахунок середніх витрат палива, часу роботи та простою; середню температуру кузова; тиск в шинах; рівень нагрівання та середні показники енергії двигуна; також коректний розрахунок маршрутів доставки замовлень підприємства.

На завершальному етапі роботи над кваліфікаційною роботою бакалавра було досліджено можливі методи тестування системи та IOT пристроїв.

Отже, процес реалізації та створення даної системи містив у собі довготривалу роботу над системою, її вдосконалення, редагування під час виявлення помилок, а також креативну частину - створення графічного інтерфейсу користувача, враховуючи необхідність створити дійсно зручну, універсальну та функціональну систему для кожного її користувача. Провівши тестування системи з більшою кількістю таблиць, можна зробити висновок, що реалізація є вдалою.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Актуальність діяльності хлібопекарень URL: https://bankchart.com.ua/biznes/biznes_start/statti/hlibopekarnya_yak_biznes_nyuansi_i_vitrati (Дата звернення 12.05.2022)
2. Розробка веб-систем на Ruby URL: https://web-creator.ru/articles/about_ruby_on_rails (Дата звернення 14.05.2022)
3. Klas Eskilson Creating User Interfaces Using Web-based Technologies to Support Rapid Prototyping in a Desktop Astrovisualization Software URL: http://www.diva-portal.org/smash/get/diva2:1169029/FULLTEXT01.pdf (Дата звернення 14.05.2022)
4. Actuality of developing the IoT systems URL: https://www.atlantis-press.com/proceedings/ameii-15/22019 (Дата звернення 17.05.2022)
5. How to use MySQL with your Ruby on Rails application URL: https://www.ionos.com/digitalguide/server/know-how/use-mysql-with-ruby-on-rails(Дата звернення 19.05.2022)
6. MySQL Reference Manual URL: https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html (Дата звернення 20.05.2022)
7. RubyMine Overview Page URL: https://www.jetbrains.com/ruby/(Дата звернення 20.05.2022)
8. Ruby On Rails Documentation URL: https://api.rubyonrails.org/(Дата звернення 20.05.2022)
9. Ruby Documentation URL: https://www.ruby-lang.org/en/documentation/(Дата звернення 23.05.2022)
10. Система логістики SkyCell URL: https://www.skycell.ch/software/ (Дата звернення 23.05.2022)
11. Система логістики Ruptela URL: https://www.ruptela.com/(Дата звернення 23.05.2022)
12. Система логістики Sawtooth URL: https://sawtooth.hyperledger.org/docs/1.2/ (Дата звернення 23.05.2022)
13. Документація по IOT URL: https://docs.microsoft.com/ru-ru/azure/iot-fundamentals/ (Дата звернення 29.05.2022)

14. Застосування IOT у транспортному секторі URL: https://pl.farnell.com/the-transportation-process-in-the-iot-world (Дата звернення 29.05.2022)
15. Види обладнання та датчики в IOT URL: https://habr.com/ru/company/otus/blog/580288/ (Дата звернення 29.05.2022)
16. Створення бази даних для збереження інформації IOT пристроїв URL: https://moodle.taltech.ee/mod/book/view.php?id=40726&chapterid=8873(Дата звернення 30.05.2022)
17. Економічна сутність та значення логістики для діяльності підприємства URL: http://www.economy.in.ua/pdf/5_2015/30.pdf (Дата звернення 14.05.2022)
18. Вибір пристроїв для реалізації IOT системи URL: https://deps.ua/ua/katalog/iot.html (Дата звернення 02.06.2022)
19. IOT in Logistics URL: https://www.digiteum.com/internet-of-things-logistics/ (Дата звернення 02.06.2022)
20. PostgreSQL and MySQL Difference URL: https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-vs-mysql/ (Дата звернення 02.06.2022)
21. Web Application Testing URL: https://www.softwaretestinghelp.com/web-application-testing/ (Дата звернення 02.06.2022)

ДОДАТКИ

Додаток А - Код програми

```
  has_one :read
  has_many :vehicles
end
class EngineWork < ApplicationRecord
  has_one :read
end
class Read < ApplicationRecord
  has_many :drivers
  belongs_to :sensor
  has_many :temperatures
  has_many :velocities
  has_many :oil_consumptions
  has_many :engine_works
  has_many :tyre_pressures
  has_many :times
  has_many :drivers
end
class Sensor < ApplicationRecord
  has_many :reads
  has_one :vehicle
end
class Temperature < ApplicationRecord
  has_one :read
  scope :high_temp, -> { where("average_temp >= 25") }
end
class TyrePressure < ApplicationRecord
  has_one :read
end

class Vehicle < ApplicationRecord
  has_one :sensor
  has_many :drivers
end
class Velocity < ApplicationRecord
  has_one :read
end
<%= render :partial => 'welcome/navbar'
<%= form_with(model: driver, local: true) do |form| %>
  <% if driver.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(driver.errors.count, "error") %> prohibited this driver from being saved:</h2>
      <ul>
      <% driver.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
      </ul>
    </div>
  <% end %>
<div class="modal-box">
  <h1>New driver</h1>
  <div class="field">
    <%= form.label :driver_name %>
    <%= form.text_field :driver_name %>
  </div>
  <%=form.submit 'Create', class: 'btn'%>
<!--    <button class="btn"><%#= form.submit %></button>-->
</div>
  <button class="btn"><%= link_to '', drivers_path %>Back</button>
<% end %>
<h1>drivers#create</h1>
<p>Find me in app/views/drivers/create.html.erb</p>
<h1>New driver</h1>
```

```
<p id="notice"><%= notice %></p>
<div class = "modal-box">
<h1 > drivers</h1>
<table style="align: center">
  <thead>
    <tr onclick="">
      <th>Driver_ID</th>
      <th>Driver Name</th>
      <th>Read ID</th>
      <th colspan="3"></th>
    </tr>
  </thead>
  <tbody >
    <% Driver.all.each do |driver| %>
      <tr>
        <td><%= driver.id %></td>
        <td><%= driver.driver_name%></td>
        <td><%= driver.read_id%></td>
        <td><%= driver.created_at%></td>
        <td><%= link_to 'SHOW', driver,  class: "btn-1" %></td>
        <td><%= link_to 'EDIT', edit_driver_path(driver) %></td>
        <td><%= link_to 'DELETE', driver, method: :delete, data: { confirm: 'Ви впевнені?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>
</div>
<button class="btn"><a href="/drivers/new">ADD NEW</a></button>
</body>
<div class = "modal-box"><%= pie_chart Driver.group(:driver_name).count , donut: true, legend: "left" %>
</div>
</html>
<%= render :partial => 'welcome/navbar' %>
<p id="notice"><%= notice %></p>
<div class="modal-box">
<p>
  <strong>Driver:</strong>
  <%= @driver.id %>
</p>
<p>
  <strong>Driver Type:</strong>
  <%= @driver.driver_name %>
</p>
</div>
<%= link_to 'Edit', edit_driver_path(@driver) %> |
<%= link_to 'Back', drivers_path %>
<!DOCTYPE html>
<html>
  <head>
    <title>PekarnyaProject</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>
    <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag "//www.google.com/jsapi", "chartkick" %>
  </head>
  <body>
    <%= yield %>
    <%= render :partial => 'welcome/navbar' %>
  </body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
  <title>W3.CSS Template</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <style>
     body,h1,h2,h3,h4,h5,h6,p {font-family: "Lato", sans-serif;
     font-size: 32px}
     .w3-bar,h1,button {font-family: "Montserrat", sans-serif}
     .fa-anchor,.fa-coffee {font-size:200px}
  </style>
</head>
<body>
<!-- Navbar -->
<div class="w3-top">
  <div class="w3-bar w3-card w3-left-align w3-large" style="background: lightblue;">
    <a class="w3-bar-item w3-button w3-hide-medium w3-hide-large w3-right w3-padding-large w3-hover-white w3-large
w3-blue" href="javascript:void(0);" onclick="myFunction()" title="Toggle Navigation Menu"><i class="fa fa-bars"></
i></a>
    <a href="#" class="w3-bar-item w3-button w3-padding-large w3-white">Home</a>
    <a href="/sensors" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Sensors</a>
    <a href="/reads" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Reads</a>
    <a href="/vehicles" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Vehicles</a>
    <a href="/drivers" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Drivers</a>
            <a href="/temperatures" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-
white">Temperatures</a>
      <a href="/oil_consumptions" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Oil
Consumptions</a>
    <a href="/velocities" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Velocities</a>
    <a href="/times" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Times</a>
      <a href="/engine_works" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Engine
Works</a>
       <a href="/tyre_pressures" class="w3-bar-item w3-button w3-hide-small w3-padding-large w3-hover-white">Tyre
Pressures</a>
  </div>
  <!-- Navbar on small screens -->
  <div id="navDemo" class="w3-bar-block w3-white w3-hide w3-hide-large w3-hide-medium w3-large">
    <a href="#" class="w3-bar-item w3-button w3-padding-large">Link 1</a>
    <a href="#" class="w3-bar-item w3-button w3-padding-large">Link 2</a>
    <a href="#" class="w3-bar-item w3-button w3-padding-large">Link 3</a>
    <a href="#" class="w3-bar-item w3-button w3-padding-large">Link 4</a>
  </div>
</div>
<!--Header -->
<header class="w3-container w3-center w3-black "  style="padding:128px 16px; background: linear-gradient(#00ACC8,
#0094B9);">
  <h1 class="w3-margin w3-jumbo">Logistics System for Orders Delivery of Kulinichi Bakery</h1>
</header>
<!-- First Grid -->
<div class="w3-row-padding  w3-container">
  <div style="text-align: left">
    <div class="w3-twothird w3-center">
    </div>
    <div class="w3-center" >
      <%= image_tag "/assets/picture.png", style: "width: 60%;" %>
    </div>
  </div>
</div>
<!-- Second Grid -->
<!-- Footer -->
<footer class="w3-container w3-padding-64 w3-center w3-opacity">
  <div class="w3-xlarge w3-padding-32">
    <i class="fa fa-facebook-official w3-hover-opacity"></i>
    <i class="fa fa-instagram w3-hover-opacity"></i>
    <i class="fa fa-snapchat w3-hover-opacity"></i>
    <i class="fa fa-pinterest-p w3-hover-opacity"></i>
    <i class="fa fa-twitter w3-hover-opacity"></i>
    <i class="fa fa-linkedin w3-hover-opacity"></i>
  </div>
  <p>Powered by>Solomiia Tymoshchuk</p>
```

```
</footer>

<script>
   // Used to toggle the menu on small screens when clicking on the menu button
   function myFunction() {
      var x = document.getElementById("navDemo");
      if (x.className.indexOf("w3-show") == -1) {
         x.className += " w3-show";
      } else {
         x.className = x.className.replace(" w3-show", "");
      }
   }
</script>
</body>
</html>
<%= render :partial => 'welcome/navbar' %>
<%= form_with(model: vehicle, local: true) do |form| %>
  <% if vehicle.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(vehicle.errors.count, "error") %> prohibited this vehicle from being saved:</h2>
      <ul>
      <% vehicle.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
      </ul>
    </div>
  <% end %>
<div class="modal-box">
  <h1>New Vehicle</h1>
  <div class="field">
    <%= form.label :vehicle_type, 'VEHICLE TYPE' %>
    <%= form.text_field :vehicle_type %>
  </div>
  <div class="field">
    <%= form.label :vehicle_name, 'VEHICLE NAME' %>
    <%= form.text_field :vehicle_name %>
  </div>
  <div class="field">
    <%= form.label :driver_id, 'DRIVER ID' %>
    <%= form.text_field :driver_id %> %>
  </div>
  <%=form.submit 'Create', class: 'btn'%>
<!--    <button class="btn"><%#= form.submit %></button>-->
  <div style="margin-top: auto"> <%= render :file => 'drivers/index.html.erb' %> </div>
</div>
  <button class="btn"><%= link_to '', vehicles_path %>Back</button>
<% end %>
ActiveRecord::Schema.define(version: 2022_06_09_110015) do
create_table "drivers", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "driver_working_hours"
  t.string "driver_name"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_drivers_on_read_id"
end

create_table "engine_works", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "average_power"
  t.integer "average_level_heating"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_engine_works_on_read_id"
end
create_table "oil_consumptions", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "average_consumption"
  t.integer "read_id"
  t.datetime "created_at", null: false
```

```
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_oil_consumptions_on_read_id"
end
create_table "order_strings", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.decimal "quantity", precision: 10
  t.decimal "price", precision: 10
  t.bigint "order_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.bigint "product_id"
  t.index ["order_id"], name: "index_order_strings_on_order_id"
  t.index ["product_id"], name: "product_id_idx"
end

create_table "reads", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "sensor_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
end
create_table "sensors", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.string "sensor_type"
  t.string "sensor_data"
  t.string "sensor_name"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.integer "vehicle_id"
  t.index ["vehicle_id"], name: "index_sensors_on_vehicle_id"
end
create_table "temperatures", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "average_temp"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_temperatures_on_read_id"
end
create_table "times", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "working_hours"
  t.integer "passive_hours"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_times_on_read_id"
end
create_table "tyre_pressures", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "tyre_percentage"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_tyre_pressures_on_read_id"
end
create_table "vehicles", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.string "vehicle_type"
  t.integer "driver_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.string "vehicle_name"
  t.index ["driver_id"], name: "index_vehicles_on_driver_id"
end
create_table "velocities", options: "ENGINE=InnoDB DEFAULT CHARSET=utf8", force: :cascade do |t|
  t.integer "average_speed"
  t.integer "read_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["read_id"], name: "index_velocities_on_read_id"
end
```

КИЇВСЬКИЙ НАЦІАНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

СИСТЕМА ЛОГІСТИКИ ДОСТАВКИ ЗАМОВЛЕНЬ ХЛІБОЗАВОДУ "КУЛИНИЧІ"

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВИКОНАЛА
ТИМОЩУК СОЛОМІЯ ВОЛОДИМИРІВНА

КЕРІВНИК
АСИСТЕНТ ГЛАДКА МИРОСЛАВА ВІКТОРІВНА

ВСТУП

Мета: розробка розумної системи логістики доставки замовлень для хлібопекарні з використанням IoT технологій та розробки схеми, алгоритсу коректного моніторингу, обробки та аналізу даних отримуваних з транспортної системи підприємства.
Об'єкт дослідження – IoT система логістики.
Предмет дослідження – імплементація розумної системи логістики.

ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ЗАВДАННЯ

- дослідити можливість впровадження IoT системи в логістичній галузі підприємства з виготовлення хліба;
- розробити схему встановлення відповідних пристроїв та датчиків на транспортних засобах підприємства;
- досягти ефективності в обробці отримуваної інформації;
- досягти оптимізації у процесах зчитування та зберігання інформації з датчиків;
- досягти ефективності у організації доставки хлібобулочних виробів;
- розробка системи на обраній мові програмування;
- розробка бази даних.

ПЛАН ВПРОВАДЖЕННЯ

1. Визначення вдосконалення, необхідних для модернізації платформ і створення екосистеми, що постійно розвивається, щоб використовувати потужність даних для прийняття більш розумних і швидших рішень.

2.Звернення уваги на потенційні перешкоди під час налаштування інформаційного потоку (наприклад, проблеми з підключенням на комунікаційному рівні, обсяг даних і частота передачі через інтерфейси).

3.Використовування масштабованих рішень за допомогою прискорювачів та агрегаторів, що може призвести до припинення більшості робіт з розробки, прискорення рентабельності інвестицій та збільшення прибутковості.

4.Інтегрування поточних фреймворків щоб дані безперешкодно протікали через хмарні рішення.

5.Створіть інструменти видимості для організації та клієнтів на вершині екосистеми, щоб збільшити цінність.

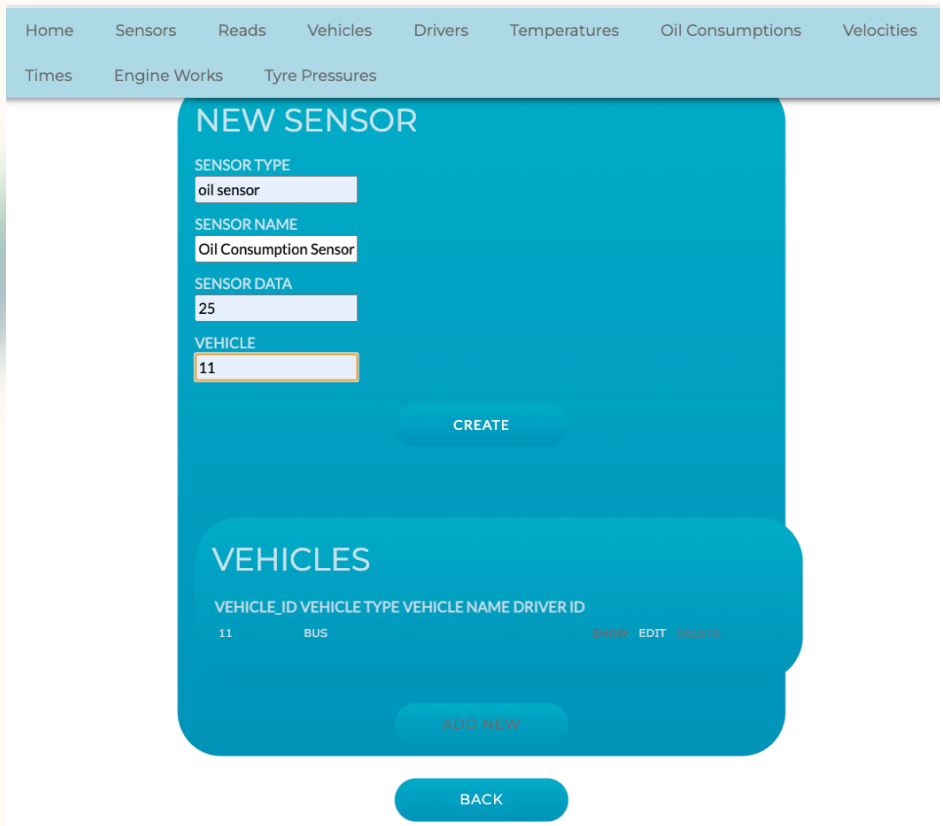
ПРИНЦИП РОБОТИ СИСТЕМИ

ДАНІ, ЗЧИТУВАНІ З ДАТЧИКІВ, РОЗМІЩЕНИХ НА ТРАНСПОРТНОМУ ЗАСОБІ ПІДПРИЄМСТВІ НАДХОДЯТЬ У СИСТЕМУ ТА СЕРЕДНІ ПОКАЗНИКИ ЗБЕРІГАЮТЬСЯ В РОЗРОБЛЕНІЙ БАЗІ ДАНИХ ПІДПРИЄМСТВА.

Дана система має такі компоненти як: пристрій для вимірювання тиску в шинах, термометр, задня камера, пристрій для вимірювання витрат пального, тахограф, датчик для вимірювання стану двигуна, передня камера.

ІНТЕРФЕЙС СИСТЕМИ

СТОРІНКА ДОДАВАННЯ ДАТЧИКУ



ІНФОГРАФІКА К-СТІ ДАТЧИКІВ ПО ЇХ ТИПУ