Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

Розроблення системи автоматизованого моніторингу технічного стану мобільних роботів на підприємстві
(тема)

Виконав:
студент 2 курсу, групи КТРСм-22-2

Кравченко С.В.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник доцент Бронніков А.І.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024

Я, як студент ХНУТЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Кравченко С. В.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
«____»_____20_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кравченку Сергію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи автоматизованого моніторингу технічного стану мобільних роботів на підприємстві

Затверджена наказом по університету від 03.11.2023 № 1288Ст
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи моніторинг роботів, трансляція даних, режим реального часу

4. Перелік питань, що потрібно опрацювати в роботі
Вступ; Аналіз проблеми та постановка задачі; Методика розробки системи моніторингу технічного стану мобільних роботів; Розробка системи моніторингу технічного стану мобільних роботів; Експериментальні дослідження; Висновки; Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій _____

Графічний демонстративні матеріали в форматі PowerPoint (.ppt) формату*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз технічного завдання	10.10 – 15.10.23	Виконано
2	Визначення концепції та об'єктів дослідження	18.10 – 01.11.23	Виконано
3	Проектування програмного забезпечення	05.11 – 15.12.23	Виконано
4	Розробка програмного забезпечення	20.11 – 29.12.23	Виконано
5	Проведення експериментальних досліджень	07.01 – 14.01.24	Виконано
6	Подання роботи на перевірку Інтернет-сервісом Unicheck	16.01.24	Виконано
7	Оформлення пояснювальної записки	16.01.24	Виконано
8	Подання роботи на рецензію	17.01.24	Виконано
9	Подання роботи на підпис зав. кафедри	18.01.24	Виконано
11	Подання кваліфікаційної роботи в ЕК	за день до захисту	Виконано

Дата видачі завдання 10.10.2023

Студент _____ Кравченко С.В
(підпис) (прізвище, ініціали)

Керівник роботи _____ доцент Бронніков А.І
(підпис) (посада, прізвище, ініціали

РЕФЕРАТ

Пояснювальна записка: 75 с., 15 табл., 36 рис., 3 дод., 26 джерел посилання.

АВТОМАТИЗАЦІЯ, ВИРОБНИЦТВО, МОБІЛЬНИЙ, МОНІТОРИНГ, РЕВОЛЮЦІЯ, РОБОТ, СЕРВІС, СИСТЕМА.

Об’єктом дослідження є моніторинг робототехнічних систем.

Предметом досліджень є методи оптимізації технологічних процесів.

Мета досліджень: створення системи моніторингу станів мобільних роботів, використовуючи концепцію Індустрії 5.0.

Методи дослідження: системний підхід, методи структурного аналізу, використання сучасних технологій розробки.

Галузь застосування розробки: дану систему можна використовувати, як на підприємствах, так і в особистих цілях, як основу для розробки рішень моніторингу станів робототехнічних систем.

У кваліфікаційній роботі розроблено систему моніторингу і досліджено її поведінку в різних режимах роботи, а також визначено параметри для оптимізації якості трансляції даних.

Для цього, за допомогою середовища розробки Visual Studio, розроблено відповідне програмне забезпечення для трансляції даних та генерації умов трансляції.

Результати досліджень приведені в форматі знімків екрану у звіті.

Результати досліджень апробовані у науковій статті.

ABSTRACT

Explanatory note: 75 p., 15 tabl., 36 pic., 3 app., 26 sources.

AUTOMATION, MOBILE, MONITORING, PRODUCTION, REVOLUTION, ROBOT, SERVICE, SYSTEM.

The object of research is the monitoring of robotic systems.

The subject of research is methods to optimizing technological processes.

Research goal: creation of a system for monitoring the states of mobile robots, using the concept of Industry 5.0.

Research methods: systematic approach, methods of structural analysis, use of modern development technologies.

Field of application of the development: this system can be used both at enterprises and for personal purposes as a basis for developing solutions for monitoring the states of robotic systems.

In the qualifying work, a monitoring system was developed and its behavior in different operating modes was studied, and parameters for optimizing the quality of data transmission were determined.

For this, with the help of the Visual Studio development environment, appropriate software for data translation and generation of translation conditions has been developed.

The results of the research are given in the format of screenshots in the report.

The research results are tested in a scientific article.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БДРЧ – база даних реального часу;

СУБД – система управління базами даних;

API – application programming interface.

ВСТУП

Впродовж свого існування людство створило багато речей для комфортного, безпечного та ефективного досягнення своїх цілей. В ході еволюції ідей для покращення життя та їх реалізації, еволюціонували і проблеми, складність вирішення яких виростає з кожним днем. Деякі з цих проблем людина самостійно вже не може вирішити без допоміжних засобів. Це спонукає нас знаходити нові підходи та методи для знаходження рішення сучасних викликів, що призводить до видозмінення сприйняття світу і цей процес нескінчений, бо кожна вирішена проблема генерує нову.

Одним із результатів цього процесу є поява та розвиток систем автоматизації, що допомагають людям реалізовувати різні процеси швидше, точніше, безпечніше та зручніше. Зараз складно уявити якусь сферу діяльності без таких систем. Наприклад, сучасне виробництво без автоматизації процесів не зможе існувати, бо воно буде не конкурентноспроможним. Виробництво, що використовує, автоматичні конвеєри, маніпулятори, верстати з програмним управлінням та інші робототехнічні пристрої, зможе виготовити продукт в набагато більших кількостях і більш якісніше. Це означає, що впровадження роботів в сфери діяльності є важливою складовою. Але з цього випливає питання обслуговування. Такі системи є досить складними для налаштування, контролю, ремонту та і самої розробки. Тому невід’ємною частиною робототехнічних систем є моніторинг, що робить цей напрям актуальним в наш час. Моніторинг дозволяє аналізувати роботу автоматичних систем, оперативно реагувати на зміни в них і оптимізувати процеси.

Метою даної роботи є створення системи моніторингу станів мобільних роботів, використовуючи концепцію Індустрії 5.0. Об’єктом дослідження є моніторинг робототехнічних систем. Предметом досліджень є методи оптимізації технологічних процесів. Робота оформлена згідно методичних вказівок [1] та ДСТУ 3008-2015 [2]. Матеріали роботи апробовані у [3].

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

Розвиток технологій та глобалізація світового ринку призвели до появи декількох промислових революцій – переходів до нових технологічних способів виробництва із застосуванням нових інноваційних засобів. Всього налічується 5 таких революцій. П’ята промислова революції ще повністю не прийшла до нас, але плавно ми робимо цей перехід.

1.1 Індустрія 5.0, як вектор розвитку для сучасник виробництв

Індустрія 5.0 – це нова модель виробництва, в якій загальна увага приділяється взаємодії між людиною та машинами [4].

Сьогодні ми переживаємо етап, що називається Індустрія 4.0. Особливостями цього етапу є використання цифрових технологій в виробництві. Він інтегрував у діяльність такі інновації як штучний інтелект, інтернет речей, аналітику великих даних, кібер-фізичні системи та інші сучасні технології. Індустрія 4.0 направлена на повну автоматизацію процесів і мінімізацію ролі людини. В технічному аспекті, ця парадигма покращила ефективність виробництв, але виявилась не стійкою до глобальних потрясінь, наприклад пандемії, вона не є екологічно безпечною, за рахунок збільшення технічних засобів та масштабу виробництва, і заміна людей машинами призвела до протестів серед працівників. Це спричинило появу ідей про Індустрію 5.0. Варто зазначити, що Індустрія 5.0 не є заміною Індустрії 4.0. Вона є більше доповненням до неї, що вирішує вказані недоліки і додає нові корисні характеристики.

Індустрія 5.0 зосереджена на вміннях, навичках і знаннях працівників співпрацювати з машинами та роботами. Інтерфейс людина-машина є ключовою, як на мене, характеристикою цього етапу. Ця комбінація є потужним інструментом для отримання максимальної ефективності. Вона

використовуватиме людський мозок для гнучких задач, наприклад швидкого аналізу та оцінки ситуації, прийняття рішень, тощо, а для виконання процесів будуть залучені потужності цифрових технологій. Виходить, що людина і машина будуть доповнювати один одного, що виключає конкуренцію між ними. Наприклад, людина може придумати більш творче і нестандартне вирішення проблеми, а машина може швидше за людину обробити велику кількість даних для реалізації цього рішення. Зараз людство вже має перші результати в вигляді ChatGPT або Bard, що інтегруються майже в усі сфери діяльності.

Важливою складовою цього етапу є постійний саморозвиток та самонавчання працівників, що змінює їх концепцію із «вартості» на «інвестицію». Це означає, що керівники підприємств, давши умови працівникам навчатись новим навичкам і знанням, відкривають розвиток не тільки людям, а і підприємству в цілому.

Також фокус уваги Індустрії 5.0 направлений на гнучкість виробничих процесів і вплив на навколишнє середовище. Перехід на неї в якомусь сенсі може врятувати нас в перспективі, так як вона штовхає людство на розвиток екологічних технологій. Вже зараз стратегії великих корпорацій спрямовані на залишку мінімального вуглецевого сліду. Як приклад, на рисунку 1.1 представлено прогноз по зменшенню вуглецевого сліду компанії Apple [5].

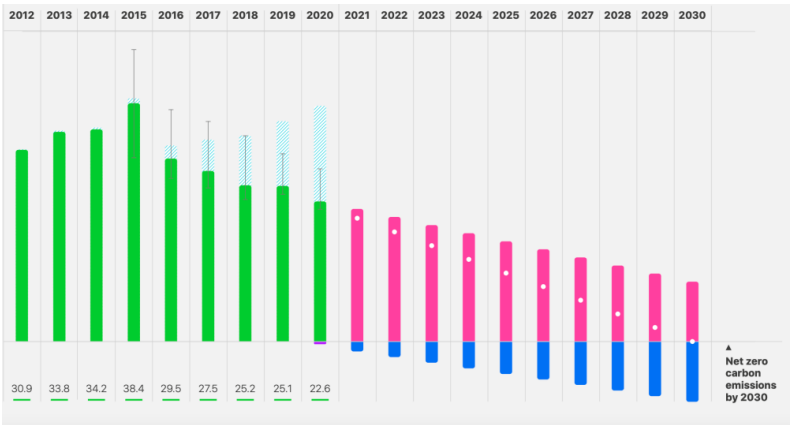

Рисунок 1.1 – Прогноз компанії Apple по зменшенню вуглецевого сліду до 2030 року [4]

Це є сучасним трендом і він призведе не тільки до порятунку навколишнього середовища, а й до розвитку технологій, які ще декілька років тому здавалися неможливими.

В ході обговорень на низці міжнародних самітів було визначено 6 категорій технологій, що повинні підтримуватись п’ятою промисловою революцією [6]:

– індивідуалізована взаємодія людини та машини;

– біоінформаційні технології та інтелектуальні матеріали;

– цифрові двійники та моделювання;

– технології передачі, зберігання та аналізу даних;

– штучний інтелект;

– технології енергоефективності, відновлюваних джерел енергії, зберігання та автономії.

Для більш наглядного розуміння, що ми маємо зараз і до чого нам потрібно прийти, в таблиці 1.1 наведено ключові відмінності Індустрії 4.0 та Індустрії 5.0.

Таблиця 1.1 – Ключові відмінності між Індустрією 4.0 та Індустрією 5.0 [7]

Індустрія 4.0	Індустрія 5.0
1	2
Робить процеси більш автоматизованими. Використовуються роботи та автономні машини для повторюваних, небезпечних або точних завдань.	Зосереджується на клієнтському досвіді та індивідуальному підході. Розвиток нових навичок і компетенцій серед працівників сфери послуг.
Використовує цифрові двійники та інструменти моделювання для оптимізації виробничих процесів.	Застосовує передові нано- та біотехнології для створення нових матеріалів і продуктів.

Продовження таблиці 1.1

1	2
Прогнозоване обслуговування, дистанційний моніторинг та аналіз даних у режимі реального часу для підвищення ефективності та зниження витрат.	Пріоритетність сталого розвитку та етичних виробничих практик для мінімізації відходів і зменшення впливу на навколишнє середовище.
Робоча сила зазвичай знаходиться далеко від заводу або задіюється мінімально.	Робоча сила має бути на заводах, щоб включати людський фактор у процеси.
Продукт має бути інтелектуальним і легко відстежуватися протягом виробничого процесу. Застосовує інтернет речей, штучний інтелект, машине навчання для автоматизації завдань і рішень.	Продукти повинні створюватися з поєднанням передових технологій, людських навичок і креативності.
Більше уваги приділяє технологіям, автоматизації та аналізу даних.	Більше уваги приділяє кастомізації та інноваціям у виробництві.

Як вказувалося, Індустрія 5.0 є розширенням Індустрії 4.0, тому щоб виконати перехід, спочатку слід розібратись з основними технологіями, що використовуються зараз. Це дозволить привести їх до стандартів нової промислової революції. Розглянемо, що з себе представляють кібер-фізичні системи та інтернет речей.

1.2 Поняття кібер-фізичних систем та інтернет речей

Кібер-фізичні системи – це інтелектуальні системи, у які входять мережі фізичних та обчислювальних компонентів, що інженерно взаємодіють [8].

Фізичні процеси, що протікають в системі впливають на обчислення і навпаки, обчислення можуть змінювати характери процесу на основі отриманих даних. Кібер-фізичну систему можна назвати цифровим двійником фізичної системи, так як вона збирає дані з усіх датчиків, які відображають повний стан системи. Цей стан можна використати для прийняття рішень або для моделювання процесів, що допомагає в їх оптимізації, за допомогою спеціального програмного забезпечення.

Розглянемо архітектуру кібер-фізичних систем більш детально. Вона складається з 5 основних модулів, що поєднуються інтернетом наступного покоління (рисунок 1.2) [3].

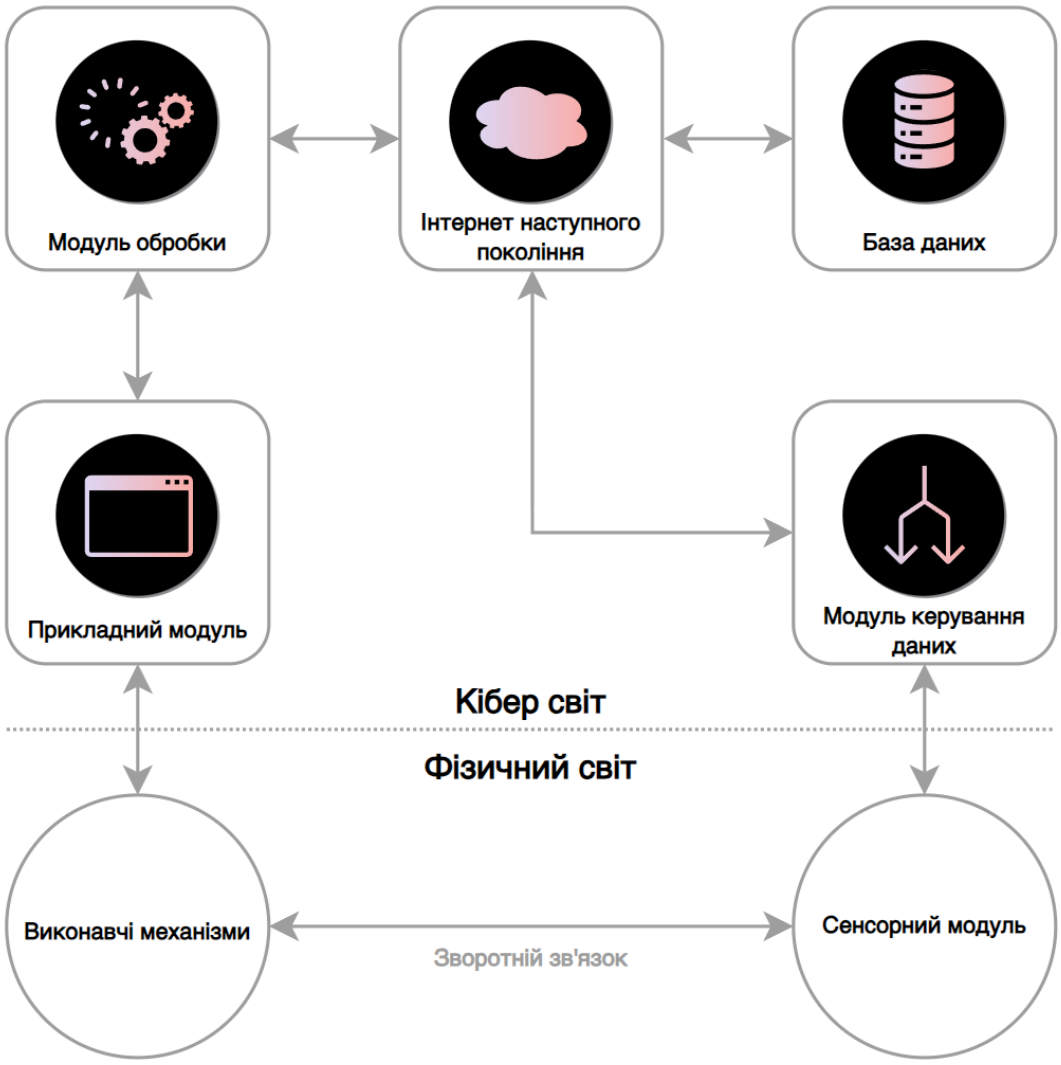

Рисунок 1.2 – Архітектура кібер-фізичної системи [3]

Сенсорний модуль необхідний для збору даних із фізичного світу за допомогою датчиків. Основна функція роботи цього модуля це обізнаність навколишнього середовища, яка досягається шляхом попередньої обробки даних. Вихідні дані надаються модулю керування даними.

Модуль керування даними складається з обчислювальних пристроїв і носіїв інформації. Це забезпечує неоднорідну обробку даних, таку як нормалізація, зменшення шуму, зберігання даних та інші подібні функції. Модуль розглядається як міст між динамічним середовищем і сервісами, оскільки він збирає дані від датчиків і пересилає дані модулям, які відповідають за обслуговування, використовуючи інтернет наступного покоління.

Загальною рисою інтернету наступного покоління є можливість для програм вибирати шлях або шляхи, якими їхні пакети проходять між джерелом і одержувачем. Цей динамічний характер інтернет-сервісу необхідний для розробки кібер-фізичної системи. На відміну від поточної архітектури інтернету, де протоколи маршрутизації знаходять єдиний шлях між джерелом і одержувачем, майбутні протоколи маршрутизації інтернету повинні будуть надавати програмам можливість вибору шляхів.

Модуль обробки забезпечує типові функції всієї системи, включаючи прийняття рішень, аналіз завдань, розклад завдань тощо. Після отримання даних цей модуль розпізнає та надсилає дані до доступних сервісів.

Прикладний модуль надає певну функціональність для користувача або іншого модуля. Він може використовувати дані з сенсорів та приводів для контролю фізичного світу або надання інформації про нього. Також модуль призначений для виконання координації дій або обміну даними.

Виконавчі механізми та датчики є двома різними електронними пристроями, які взаємодіють із фізичним середовищем. Виконавчий механізм отримує команди від прикладного модуля та виконує їх.

Зворотній зв’язок служить для мінімізації обробки даних шляхом зв’язку між датчиком і приводом для безпосереднього виконання необхідних дій.

Можливість створювати кібер-фізичні системи з’явилась з приходом інтернету речей. Інтернет речей – це мережа фізичних об'єктів, які мають вбудовані технології, що дозволяють здійснювати взаємодію з зовнішнім середовищем, передавати відомості про свій стан і приймати дані ззовні [9].

Ідеї, що покладені в інтернет речей та кібер-фізичних системах можна застосувати до системи моніторингу мобільних роботів, так як загальні компоненти і технології підходять для її реалізації.

1.3 Загальні відомості про мобільні роботи

Мобільні роботи є відносно новим напрямом робототехніки. Це автономні роботи різного призначення з вбудованим бортовим комп’ютером.

Широке впровадження таких роботів пов’язане з підвищенням продуктивності праці та зменшенням небезпеки для життя людини. Мобільні роботи можуть працювати в агресивних середовищах, що шкідливі для здоров’я людини.

При розробці мобільних роботів використовуються компактні та енергоємні джерела енергії, високонадійні транспортні засоби високої прохідності, високоточні навігаційні системи та інше. Якщо порівняти декілька мобільних роботів різного призначення, то можна виділити 3 базові частини з яких вони складаються: рухомий виконавчий агрегат, пульт дистанційного керування та агрегат живлення. Базова модель може бути розширена різними функціональними системами. Ось декілька з них [10]:

− супутникова навігаційна система;

− інерціальна навігаційна система;

− радіаційно-дозиметрична система;

− система гідродинамічного руйнування;

− система рентгеноскопічного контролю;

− система аналізу газових середовищ;

− система кріогенного охолодження;

– система виявлення мін.

Насправді список функціональних комбінацій мобільних роботів практично нескінчений, але є декілька недоліків, які поки не виключають використання стаціонарних роботів. Першою проблемою є не вирішена проблема довгої автономності. Сучасні батареї поки не дійшли до того рівня, щоб тримати заряд дуже великий термін без заряджання, однак в людства є певні покращені в даному питані. Зараз розвиваються технології швидкого заряджання, а не збільшення ємності батареї. Другою проблемою є велика ціна деяких мобільних роботів. Це пов’язано з тим, що такі роботи в основі використовують найновітніші технології, що стануть дешевими лише з часом за рахунок серійного або масового виробництва.

Розглянемо декілька мобільних роботів в якості прикладу.

На рисунку 1.3 представлено мобільний робот RB-KAIROS+ від компанії Robotnoik. Це мобільний маніпулятор, розроблений для інтеграції маніпуляторів Universal Robots e-Series. Його програмне та апаратне забезпечення повністю підготовлені для встановлення маніпулятора OEM DC e-Series і, таким чином, його можна перетворити на мобільний маніпулятор. Це дозволяє необмежено розширювати робочий простір кобота, який може працювати з великими деталями або виконувати операції підбору та розміщення на великих площах. У цьому сенсі він забезпечує широкий спектр нових застосувань і є чудовим доповненням для поточних користувачів маніпуляторів URe [11].

На рисунку 1.4 зображено мобільний робот MiR100 компанії MiR. Це безпечний, економічно ефективний мобільний робот, який швидко автоматизує внутрішній транспорт і логістику. Робот оптимізує робочі процеси, звільняючи ресурси персоналу, щоб підвищити продуктивність і зменшити витрати [12].

Рисунок 1.3 – Мобільний маніпулятор RB-KAIROS+

Рисунок 1.4 – Мобільний робот MiR100

Як видно, такі роботи дійсно дуже корисні на підприємствах, але, як і все обладнання, вони потребують моніторингу для їх контролю. Проаналізуємо особливості сучасних системи моніторингу більш детально.

1.4 Особливості сучасних системи моніторингу автоматизованих систем

Сучасні системи моніторингу представляють з себе цілий комплекс технологій. До них відносяться бази даних, інформаційно-комунікаційні технології, інтелектуальні інформаційні технології, тощо. В сукупності це забезпечує автоматизований моніторинг роботи обладнання та персоналу в режимі реального часу. Однією з головною вимогою до сучасних моніторингових систем є мобільність. Система повинна забезпечувати можливість доступу до інформації будь де та будь коли з різних засобів.

Розробляючи систему моніторингу автоматизованих систем, необхідно розглядати дві частини, а саме програмну та апаратну.

В апаратну частину входять пристрої протилежних напрямів, одні інформацію збирають – сенсори, а інші цю інформацію використовують і генерують – виконуючі пристрої. Для взаємодії цих пристроїв використовуються дротові та бездротові технології зв’язку. Апаратні компоненти, що використовуються для збору та обробки даних, повинні забезпечувати розв’язання задач у реальному часі. Також є ряд вимог до апаратних компонентів, що пов’язані з масогабаритністю, споживанням та надійністю. Що стосується програмної частини, вона повинна стабільно підтримувати високонавантажені обчислення, працювати за принципом асинхронності та бути побудованою з використанням відкритого програмного забезпечення.

Якщо брати до уваги сучасні типові технологічні процеси, то можна виділити такі особливості для розробки моніторингових систем [13]:

– бажано застосовувати технології, що просто розгортаються та підтримуються;

– архітектура рішення має бути з можливістю масштабування, так як параметри, що описують стан системи можуть змінюватись;

– апаратна частина системи повинна розроблятися на основі сучасних типових рішень для забезпечення простоти та низької вартості;

– необхідне широке застосування базових телекомунікаційних протоколів.

1.5 Постановка задачі та висновки до розділу 1

Проаналізувавши поточний стан розвитку автоматизованих систем та промисловості в цілому, було сформовано основне бачення даного проекту.

Система моніторингу, що розробляється повинна відповідати вимогам мобільності, тобто доступ до інформації повинен здійснюватися в будь-який час та в будь-якому місці на різних засобах. В ході розробки необхідно використовувати технології, які прості в розгортанні і підтримці та слідувати принципам масштабованості системи. Збір та обробка даних повинна виконуватись в режимі реального часу за допомогою базових телекомунікаційних протоколів. Розроблена система моніторингу має враховувати всі особливості мобільних роботів і надавати інтерфейс для користувача за стандартами Індустрії 5.0. Це означає, що комунікація між людиною та роботом повинна бути влаштована таким чином, щоб вони працювали разом, доповнюючи один одного, для отримання максимально швидкого та ефективного результату моніторингу.

2 МЕТОДИКА РОЗРОБКИ СИСТЕМИ МОНІТОРИНГУ ТЕХНІЧНОГО СТАНУ МОБІЛЬНИХ РОБОТІВ

2.1 Концепція та межі проекту

Метою даного проекту є розробка сервісу для надання інформації про технічний стан мобільного робота в режимі реального часу. Він допомагає в вирішені проблеми моніторингу простих та складних робото-технічних систем в рамках підприємства або менших масштабах. Рішення дає змогу зручно з будь-якого пристрою в форматі чат-боту отримувати різну інформацію від мобільних роботів різного призначення. Гарною реалізацією є розробка відповідного програмного забезпечення, що дасть змогу відстежувати будь-який технічний стан мобільного робота в режимі реального часу і представляти інформацію в звичайному текстовому та графічному вигляді. Зв’язок з системою повинен встановлюватись через зручний інтерфейс, використовуючи базові протоколи.

Рішення направлення на користувачів, які створюють і втілюють в роботу свої мобільні роботи і бажають стежити за їх поточним і минулим станом за допомогою одного універсального інструменту, не створюючи власний.

Розроблювальна система повинна задовольняти такі бізнес правила:

– кожен користувач може зареєструвати обліковий запис;

– кожен обліковий запис має унікальний ідентифікатор;

– ідентифікатор облікового запису не повинен дорівнювати 0;

– обліковий запис повинен мати псевдонім користувача;

– псевдонім користувача повинен бути унікальним;

– псевдонім користувача не менше 3 символів;

– псевдонім користувача не перевищує 50 символів;

– обліковий запис повинен мати ім’я користувача;

– ім’я користувача не менше 3 символів;

– ім’я користувача не перевищує 50 символів;

– обліковий запис користувача може мати прізвище;

– прізвище користувача не менше 3 символів;

– прізвище користувача не перевищує 50 символів;

– обліковий запис повинен мати пароль;

– пароль обліковий запису не менше 8 символів;

– пароль обліковий запису не перевищує 20 символів;

– користувач повинен бути авторизованим;

– авторизація користувача проводиться за псевдонімом та паролем облікового запису;

– кожен обліковий запис має свій список чатів з роботами;

– кожен користувач може створити безліч чатів;

– чат повинен мати ім’я;

– ім’я чату не менше 3 символів;

– ім’я чату не перевищує 30 символів;

– кожен чат має унікальний ідентифікатор;

– ідентифікатор чату не повинен дорівнювати 0;

– кожен чат повинен мати токен;

– токен чату повинен бути унікальним;

– токен чату 20 символів;

– авторизація роботу відбувається по токену чату;

– чат може відображати текстову інформацію та графіки з даними, що передав робот;

– максимальний затримка передачі даних 500 мс.

Серед бізнес ризиків можна виділити велике навантаження на систему. Із-за можливої великої кількості одночасно підключених роботів до сервісу, навантаження на нього може перебільшувати допустимі обрахункові можливості серверів. Це може привести до того, що знадобиться заключати

контракти з хостинг-провайдерами або будувати власну серверну великих масштабів.

Спираючись на виділену концепцію і враховуючи бізнес правила, побудуємо архітектуру проекту.

2.2 Архітектура проекту

Архітектура системи моніторингу технічного стану мобільних роботів базується на мікро-сервісній архітектурі і включає такі компоненти:

– сервіс авторизації;

– сервіс обробки даних;

– сервіс збору та відправки даних;

– СУБД;

– БДРЧ;

– web-додаток.

Взаємозв’язок цих компонентів представлено на рисунку 2.1.

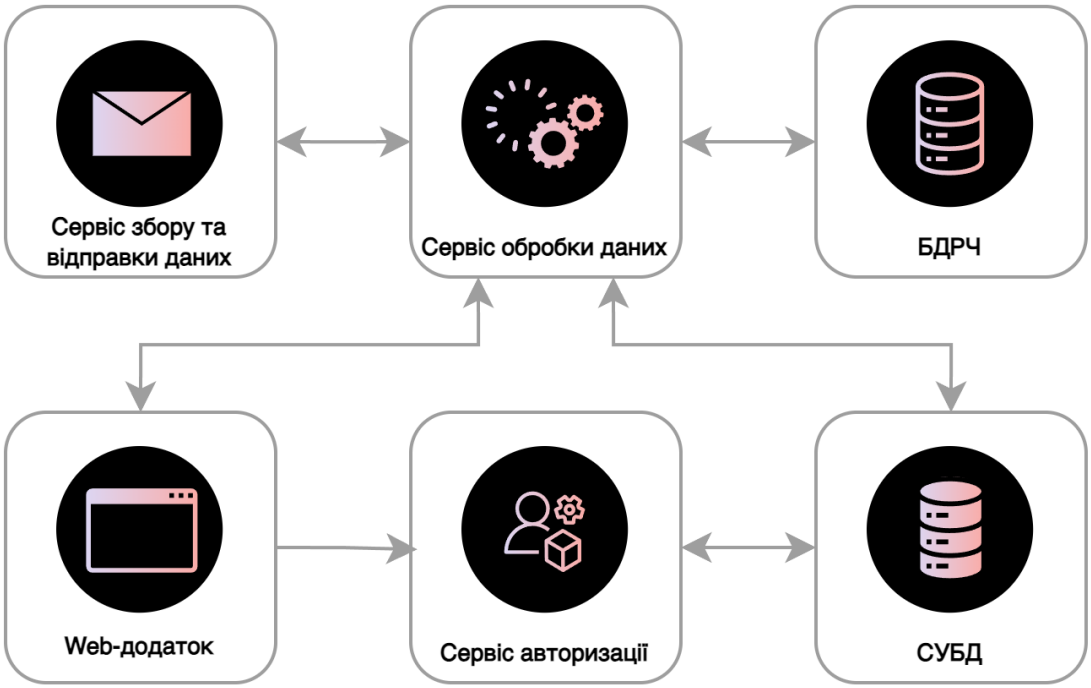

Рисунок 2.1 – Архітектура системи моніторингу технічного стану мобільних роботів

Сервіс авторизації призначений для керування обліковими записами користувачів. До функцій цього сервісу відносяться:

– створення облікового запису;

– видача ключів доступу до системи;

– управління даними користувача.

Сервіс обробки даних являється основним сервісом наведеної архітектури, так як там проходять процеси збору, обробки та розсилки даних з роботів користувачам. В функції сервісу входять:

– прийом даних з роботів;

– надання даних користувачу за запитом;

– трансляція даних користувачам в реальному часі;

– зберігання інформації в базі даних.

Для забезпечення надійного збереження та неперервного потоку даних, цей сервіс використовує в якості сховища інформації СУБД – набір пов’язаних між собою даних і програм, що забезпечують доступ до них [14]; та БДРЧ – сховище пар ключ-значення, які ще називають тегами, що постійно оновлюються в режимі реального часу [15].

Сервіс збору та відправки даних розгортається в бортовому комп’ютері мобільного робота. Він є мостом, що інтерпретує дані, що зчитуються безпосередньо з апаратної частини робота в дані, що транслюються сервісом обробки.

Web-додаток призначений для відображення отриманої інформації від роботів і використовується користувачем. Користувач за допомогою нього може авторизуватись, виконати налаштування в системі і відправити запити роботам.

2.3 Механізм трансляції та збору даних

Крім загальної архітектури проекту, в ході розробки моніторингової системи слід акцентувати увагу на механізмі трансляції та збору даних, так як

це є важлива частина в будь-якій системі такого типу. Цей механізм повинен забезпечувати передачу даних між роботом та користувачем з мінімально можливою затримкою, а також їх відтворення. В більшості програмних систем для роботи з даними використовуються СУБД, але вони не призначенні до оновлення даних в реальному часі і це може призвести до різних відмов, наприклад, коли перевищений ліміт підключень до СУБД. Із-за цього дані можуть бути втрачені, що не припустимо. Для зменшення ймовірності відмов і втрати даних, було вирішено використовувати СУБД в комбінації з БДРЧ. БДРЧ дозволяють швидко зберігати велику кількість інформації в вигляді пар ключ-значення, тому БДРЧ можна використати як кеш, з якого періодично будуть переноситись дані в СУБД. Це значно знизить навантаження на СУБД.

На рисунку 2.2 зображена схема, що описує життєвий цикл даних в системі.

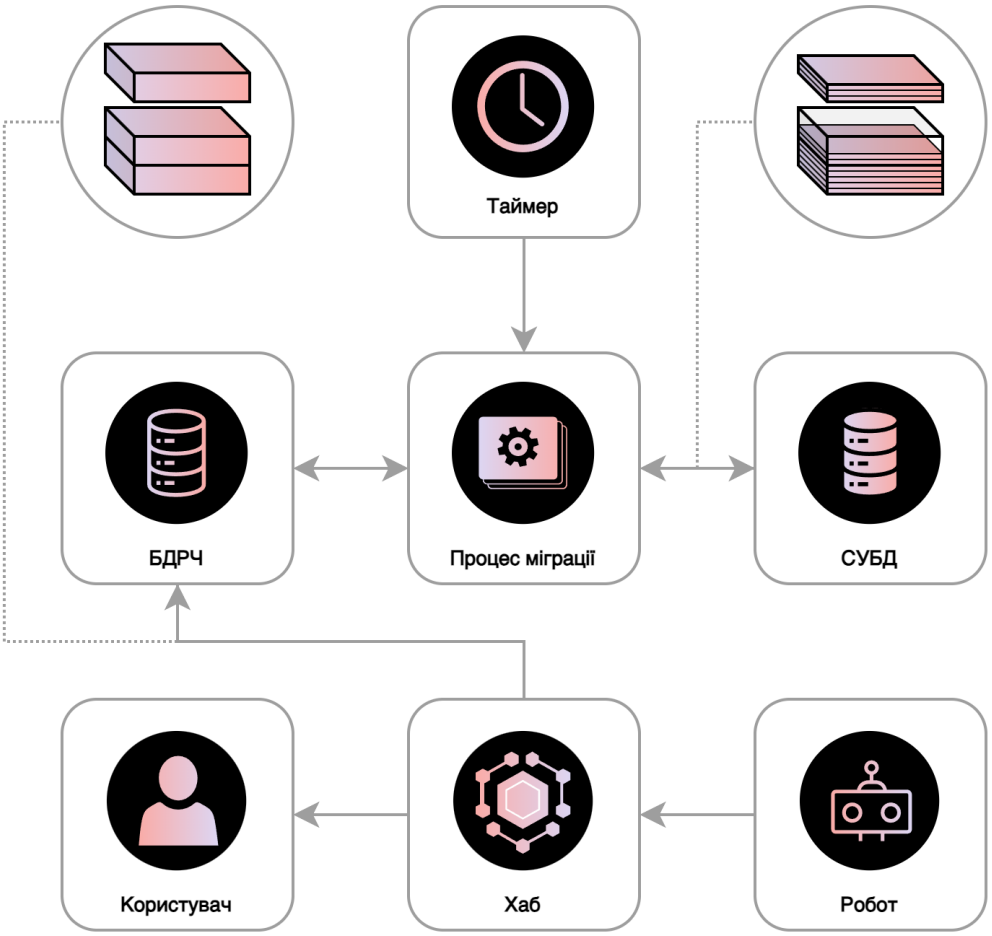

Рисунок 2.2 – Схема життєвого циклу даних в системі

Всі користувачі та роботи підключені до одного хабу, через який проходить обмін інформацією. Життєвий цикл даних стану робота в системі починається з їх надходження від підключеного робота в хаб. Хаб зберігає цей стан в БДРЧ і виконує розсилку користувачам. В цей момент в системі встановлений таймер, який періодично ініціює міграцію з БДРЧ в СУБД. В ході міграції, беруться всі збережені в БДРЧ записи та групуються по ідентифікаторам роботів та ключам станів. Далі вони зберігаються в СУБД партіями. Кожна партія має максимальну кількість станів і одна така партія еквівалентна одному запису в СУБД. Якщо партія не має максимальну кількість станів, до неї будуть додаватися нові і так поки вона не наповниться. Це зроблено для того, щоб зменшити кількість записів в СУБД і швидко викачувати необхідний масив станів в відповідній хронології.

2.4 Вибір технічних засобів для реалізації проекту

Технічні засоби для реалізації проекту можна розділити на 3 категорії:

– інструменти для реалізації «front-end» частини;

– інструменти для реалізації «back-end» частини;

– апаратні засоби.

До front-end частини відноситься web-додаток. В рамках даного проекту він буде представляти з себе сайт. Реалізація додатку буде базуватись на платформі Node.js з використанням засобів HTML, CSS та мову програмування JavaScript з бібліотекою React.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript [16]. Це відносно нова платформа, так як її перший випуск відбувся в 2009 році, а стабільний випуск відбувся в 2020 році. Вона рахується однією з лідируючих платформ для web-розробки, що працюють з JavaScript. На Node.js написані такі відомі сервіси як eBay, Paypal, Netflix та інші. Платформа підтримує систему JavaScript модулів, що дає змогу використовувати React.

React – це відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту web-сторінки, з якими стикаються в розробці односторінкових застосунків [17]. Ця бібліотека є популярною в наш час і ця популярність тільки зростає. На рисунку 2.3 показаний рейтинг бібліотек JavaScript за 2023 рік в сервісі GitHub [18].

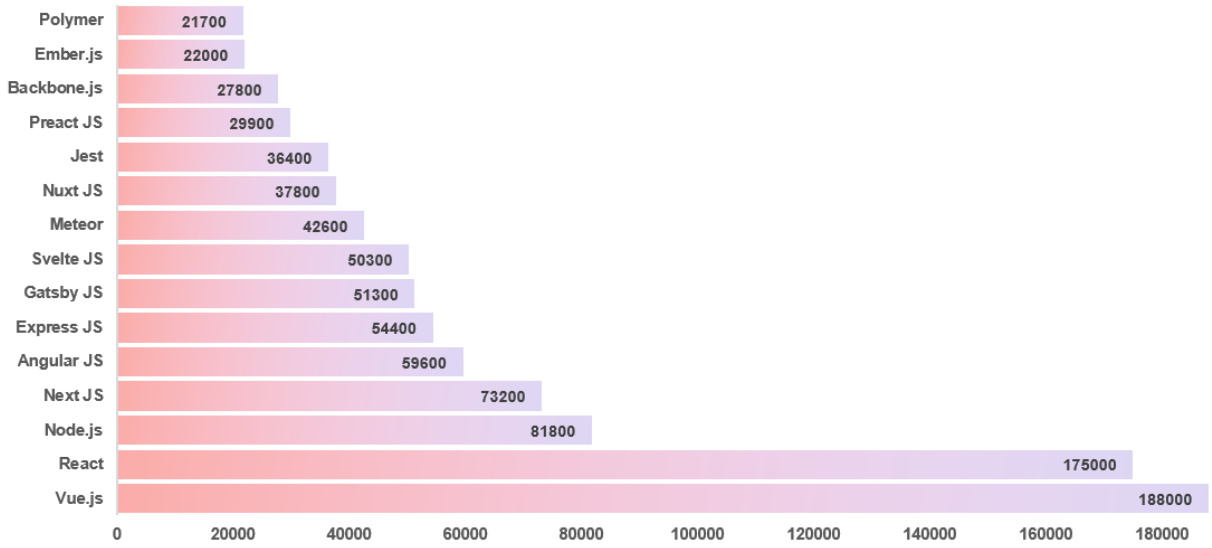

Рисунок 2.3 – Найпопулярніші JavaScript бібліотеки відповідно до зірок в GitHub [18]

Що стосується back-end частини проекту, то вона буде реалізована на платформі .NET. В якості основи буде взятий фреймворк ASP .NET Core. ASP .NET Core є крос-платформовим, високопродуктивним фреймворком з відкритим кодом, випущений компанією Microsoft в 2016 році. Він призначений для створення сучасних хмарних додатків, таких як web-додатки та служби, додатки інтернету речей та API. Для розробки на ASP .NET Core використовується мова програмування C# [19]. Вибір цієї технології обумовлений тим, що вона широко використовується і набір бібліотек підтримується багатьма компаніями, що маюсь свої програмні продукти та сервіси. Це значно пришвидшить розробку проекту. Також ASP .NET Core має бібліотеку SignalR, яка дозволяє серверному коду надсилати асинхронні

сповіщення клієнтським web-додаткам в реальному часі [20]. Це потужний і надійний інструмент, який скоротить час написання коду, так як власні засоби обробки web-sockets не потрібні. Також SignalR підтримується React, що робить web-клієнт сумісним.

В якості сховища даних буде використовуватись СУБД MySQL. MySQL – це вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних [21]. Вона широко використовується в різних сервісах корпоративного та не корпоративного призначення. Вона підтримується бібліотекою Entity Framework для .NET Core, що робить зручне використання її можливостей в коді C#.

Для зберігання даних в реальному часі був вибраний Redis. Redis – розподілене сховище пар ключ-значення, які зберігаються в оперативній пам'яті, з можливістю забезпечувати довговічність зберігання на бажання користувача [22]. За допомогою цього інструменту можна виконати потокове збереження та трансляцію даних в режимі реального часу. Redis також підтримується платформою .NET Core.

Вибрані технології для реалізації back-end частини повністю задовольняють вимогам, що були вказані в розділі 1.4. Всі вони з відкритим кодом і є кросс-платформовими, що дає змогу запускати їх в Docker в вигляді контейнерів, що можуть масштабуватись за допомогою Kubernetes – платформою для управління контейнеризованими робочими навантаженнями та супутніми службами [23].

Останньою категорією технічних засобів є апаратні засоби. Вони безпосередньо не включаються в проект, але вони взаємодіють з ним. Головною вимогою до цих засобів є доступ до інтернету, бо без нього робота моніторингу неможлива. Також можна виділити рекомендації, щодо вибору апаратного забезпечення, які повинні інтегруватись в бортову систему мобільних роботів. В якості цього, можна використати одноплатні комп’ютери типу Raspberry Pi або Orange Pi. Вони представлені на рисунках 2.4 – 2.5.

Рисунок 2.4 – Зовнішній вигляд Raspberry Pi 3 Model B

Рисунок 2.5 – Зовнішній вигляд Orange Pi One

Операційна система таких комп’ютерів дозволяє запускати програмне забезпечення високого рівня, що може приймати і оброблювати дані з апаратного забезпечення і відправляти їх по протоколам TCP та HTTP з підтримкою web-sockets. Програми можуть бути написані на багатьох мовах програмування, включаючи C, C++, C#, Java, тощо.

2.5 Проектування баз даних

В ході розробки баз даних виділено перелік сутностей баз даних сервісів авторизації та обробки даних. Їх опис представлено в таблиці 2.1.

Таблиця 2.1 – Опис сутностей баз даних сервісів

Назва	Сервіс	Опис
1	2	3
User	Сервіс авторизації	Містить інформацію про обліковий запис користувача.
WorkNode	Сервіс обробки даних	Містить інформацію про обліковий запис робота.
UserWorkNode	Сервіс обробки даних	Містить інформацію про екземпляр чату користувача з роботом.
WorkNodeStateData	Сервіс обробки даних	Містить інформацію про партію станів робота.
Message	Сервіс обробки даних	Містить інформацію про повідомлення в чаті з роботом.
MessageContent	Сервіс обробки даних	Містить інформацію про зміст повідомлення в чаті з роботом.

Опис зв’язків між сутностями представлено в таблиці 2.2.

Таблиця 2.2 – Опис зв’язків між сутностями

Сутність 1	Найменування зв’язку	Сутність 2	Тип зв’язку
1	2	3	4
WorkNode	Один до багатьох	UserWorkNode	1:M

Продовження таблиці 2.2

1	2	3	4
WorkNode	Один до багатьох	WorkNodeStateData	1:M
UserWorkNode	Один до багатьох	Message	1:M
Message	Один до багатьох	MessageContent	1:M

Детальний опис зв’язків:

– один обліковий запис робота може мати декілька екземплярів чатів користувачів з роботом, але один екземпляр чату користувача з роботом може відноситись до одного облікового запису робота;

– один обліковий запис робота може мати декілька партій станів робота, але одна партія станів робота може відноситись до одного облікового запису робота;

– один екземпляр чату користувача з роботом може мати декілька повідомлень, але одне повідомлення може відноситись до одного екземпляру чату користувача з роботом;

– одне повідомлення може мати декілька змістів, але один зміст може відноситись до одного повідомлення.

На основі цих сутностей виділим побудуємо реляційну модель. Реляційна модель – це модель даних, що орієнтована на представлення даних в вигляді набору таблиці та зв’язків між ними, де кожен рядок в таблиці описує, екземпляр об’єкта, а стовпець описує його властивість [24]. Кожен запис в таблиці відрізняється унікальним ключем, що також називається первинним. Зв’язки між таблицями реалізовуються за допомогою зовнішніх ключів, що містять значення первинного ключа зв’язаної таблиці.

Опис атрибутів таблиць баз даних сервісів авторизації та обробки даних представлений в таблиці 2.3.

Таблиця 2.3 – Опис атрибутів таблиць баз даних сервісів

Таблиця	Назва атрибуту	Опис атрибуту
1	2	3
User	Id	Ідентифікатор облікового запису користувача.
	Firstname	Ім’я користувача.
	Lastname	Прізвище користувача.
	Username	Псевдонім користувача.
	Password	Пароль від облікового запису користувача.
Message	Id	Ідентифікатор повідомлення.
	UserWorkNodeId	Ідентифікатор екземпляру чату користувача та робота.
	SenderType	Тип відправника (користувач або робот).
	CreationDateTime	Дата та час відправки повідомлення.
WorkNode	Id	Ідентифікатор облікового запису робота.
	Name	Ім’я робота.
	Token	Токен облікового запису робота.
UserWorkNode	Id	Ідентифікатор екземпляру чату користувача та робота.
	WorkNodeId	Ідентифікатор облікового запису робота.
	UserId	Ідентифікатор облікового запису користувача.
	Permission	Рівень прав доступу до облікового запису робота.

Продовження таблиці 2.3

1	2	3
WorkNodeStateData	Id	Ідентифікатор партії станів робота.
	WorkNodeId	Ідентифікатор облікового запису робота.
	Key	Ключ стану робота.
	EntriesCount	Кількість станів в партії.
	Data	Дані партії.
	CreationDateTime	Дата та час створення партії.

Опис типів атрибутів таблиць баз даних сервісів авторизації та обробки даних представлений в таблиці 2.4.

Таблиця 2.4 – Опис типів атрибутів таблиць баз даних сервісів

Таблиця	Назва атрибуту	Тип даних	За замовченням	Унікальний	Ключ PK|FK
1	2	3	4	5	6
WorkNode	Id	BIGINT	NOT NULL	+	PK
	Name	VARCHAR	NOT NULL		
	Token	VARCHAR	NOT NULL	+	
User	Id	BIGINT	NOT NULL	+	PK
	Firstname	VARCHAR	NOT NULL		
	Lastname	VARCHAR	NULL		
	Username	VARCHAR	NOT NULL	+	
	Password	VARCHAR	NOT NULL		
WorkNodeStateData	Id	BIGINT	NOT NULL	+	PK
	WorkNodeId	BIGINT	NOT NULL		FK
	Key	VARCHAR	NOT NULL		
	EntriesCount	INT	NOT NULL		
	Data	VARCHAR	NOT NULL		
	CreationDateTime	DATETIME	NOT NULL		

Продовження таблиці 2.4

1	2	3	4	5	6
UserWorkNode	Id	BIGINT	NOT NULL	+	PK
	WorkNodeId	BIGINT	NOT NULL	+	FK
	UserId	BIGINT	NOT NULL		
	Permission	INT	NOT NULL		
Message	Id	BIGINT	NOT NULL	+	PK
	UserWorkNodeId	BIGINT	NOT NULL		FK
	SenderType	INT	NOT NULL		
	CreationDateTime	DATETIME	NOT NULL		
MessageContent	Id	BIGINT	NOT NULL	+	PK
	MessageId	BIGINT	NOT NULL		FK
	Type	INT	NOT NULL		
	Data	VARCHAR	NOT NULL		

ER-діаграма баз даних сервісів авторизації та обробки даних представлена на рисунку 2.6.

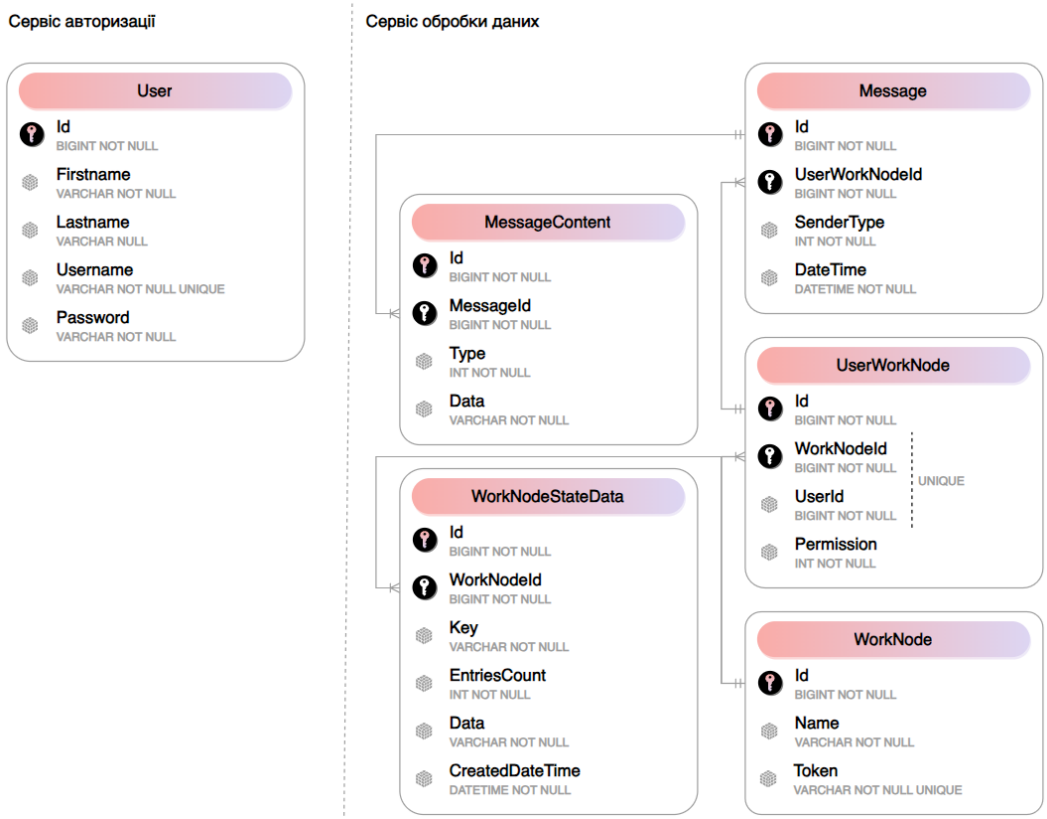

Рисунок 2.6 – ER-діаграма баз даних сервісів

При проектуванні реляційної бази даних розмірність типів даних не була розглянута, так як був вибраний Code-First підхід розробки сервісів для зручності подальшого внесення змін в проект. Це означає, що відповідальний за створення бази даних та за дані, що записуються, є програмний код сервісів.

Також визначимо пари ключ-значення для даних що будуть зберігатися в БДРЧ (таблиця 2.5).

Таблиця 2.5 – Опис пар ключ-значення для даних БДРЧ

Ключ	Опис ключа	Тип значення
1	2	3
Id	Ідентифікатор стану.	STRING
WorkNodeId	Ідентифікатор облікового запису робота.	LONG
Key	Ключ стану.	STRING
Value	Значення стану.	FLOAT
DateTime	Дата та час стану.	DATETIME

2.6 Проектування API сервісів

Виділимо методи, що будуть надаватися для роботи сервісами. Для цього деталізуємо функції кожного сервісу.

Сервіс авторизації повинен надавати такі функціональні можливості:

– створення облікових записів користувачів;

– надавання ключа доступу до облікового запису користувача;

– надавання інформації про обліковий запис користувача;

– оновлення інформацію про обліковий запис користувача.

Сервіс обробки даних повинен надавати такі функціональні можливості:

– створення облікових записів роботів;

– оновлення інформацію про обліковий запис робота.

– надавання ключів доступу до облікового запису робота;

– видалення облікових записів роботів;

– надання масиву чатів користувача з роботами;

– надання масиву повідомлень чату користувача з роботом;

– надання масиву станів робота;

– відправка повідомлень роботу користувачем;

– відправка повідомлень користувачу роботом;

– відправка стану користувачам роботом.

На основі цього отримаємо набір методів, які будуть доступні іншим компонентам в системі. Опис методів для сервісу авторизації представлено в таблиці 2.6.

Таблиця 2.6 – Опис набору методів до API сервісу авторизації

Назва методу	Аргументи	Опис
1	2	3
Login	username, password	Повертає ключ доступу до облікового запису користувача на основі псевдоніму користувача та паролю.
CreateAccount	username, firstname, lastname, password	Створює новий обліковий запис користувача.
UpdateUserInfo	username, firstname, lastname	Оновлює інформацію облікового запису користувача.
ChangeUserPassword	password, newPassword	Змінює пароль від облікового запису користувача.
GetMe		На основі ключа доступу повертає інформацію облікового запису користувача виключаючи пароль.

Опис методів для сервісу обробки даних представлено в таблиці 2.7.

Таблиця 2.7 – Опис набору методів до API сервісу обробки даних

Назва методу	Аргументи	Опис
1	2	3
CreateWorkNode	workNodeName	Створює обліковий запис робота з вказаною назвою.
UpdateWorkNodeInfo	workNodeId, workNodeName	Оновлює інформацію облікового запису робота
GetWorkNodeToken	workNodeId	Надає токен облікового запису робота.
ResetWorkNodeToken	workNodeId	Перегенеровує токен облікового запису робота.
DeleteWorkNode	workNodeId	Видаляє обліковий запис робота.
GetUserWorkNodes		Надає масив чатів користувача з роботами.
GetWorkNodeAccessToken	token	Надає ключ доступу до облікового запису робота по токену.
GetUserMessages	workNodeId, count, offset	Надає масив повідомлень чата користувача та робота з вказаною кількістю та зсувом.
GetWorkNodeStates	workNodeId, key, count, offset	Надає масив стану робота з вказаною кількістю та зсувом.
GetUserMessages	workNodeId, count, offset	Надає масив повідомлень чата користувача та робота з вказаною кількістю та зсувом.

Продовження таблиці 2.7

1	2	3
GetWorkNodeStates	workNodeId, key, count, offset	Надає масив стану робота з вказаною кількістю та зсувом.
OnUpdate	workNodeId, userId, senderType, type, message, frame	Виконує задачу розсилки оновлень між користувачами та роботами. Оновленнями можуть бути повідомлення та нова інформацію про стани роботів.

2.7 План проведення досліджень поводження системи

Розроблювальна моніторингова система повинна розгортатися в вигляді контейнерів в Docker. Це дозволить ізолювати кожен компонент від зовнішніх впливів і масштабувати його за необхідністю. Це означає, що якщо навантаження на систему буде перевищувати допустиме, можна запустити ще один екземпляр компоненту і розподілити трафік між ними. В автоматичному режимі це дозволяє робити Kubernetes, але його налаштування в даній роботі розглядатись не буде. Дослідимо поведінку одного екземпляру контейнера сервісу обробки даних при різних умовах і визначемо параметри при яких слід його масштабувати для забезпечення виконання поставлених вимог до системи. В контексті цього сервісу, є 2 параметра, що дозволяють визначити, коли потрібно запускати новий екземпляр компоненту, а саме:

– максимальна кількість користувачів та роботів, що підключено одночасно U_{max};

– максимальна допустима затримка передачі даних від робота до користувача D_{max}.

U_{max} можна визначити експериментально, підключаючи користувачів до тих пір, поки система не відмове. Цей дослід дозволить також оцінити навантаження на оперативну пам’ять, за допомогою чого можна виділити рекомендовані параметри для серверної машини.

Рекомендована міскість оперативної пам’яті буде дорівнювати (2.1):

$$C = \sum_i (c_i \cdot k \cdot 1{,}5), \quad (2.1)$$

де c_i – максимальна затрачена пам’ять і-м компонентом системи, МБ;

k – кількість екземплярів компоненту.

Для прикладу, припустимо, що об’єми пам’яті, що затрачаються 1 екземпляром компонент, і кількість запущених екземплярів такі, як в таблиці 2.8.

Таблиця 2.8 – Кількість запущених екземплярів та максимальні затрати оперативної пам’яті компонентів моніторингової системи

Компонент	k	c, МБ
1	2	3
Сервіс авторизації	1	100
Сервіс обробки даних	2	300
Web-додаток	1	50
Redis	2	500
MySql	1	500

Тоді значення рекомендованої оперативної пам’яті для серверної машини, де буде розгортатися система, буде дорівнювати:

$C = (100 \cdot 1 \cdot 1{,}5) + (300 \cdot 2 \cdot 1{,}5) + (50 \cdot 1 \cdot 1{,}5) + (500 \cdot 2 \cdot 1{,}5) + (500 \cdot 1 \cdot \cdot 1{,}5) = 150 + 900 + 75 + 1500 + 750 = 3375$ МБ.

Що стосується затримки передачі даних то, в розділі 2.1 було вказано, що D_{max} = 500 мс, тому в ході досліджень потрібно визначити параметри при яких буде виконуватись умова (2.2):

$$D_{max} \leq D, \tag{2.2}$$

де D – затримка передачі даних.

D може бути розрахована за формулою (2.3):

$$D = T_r - T_s, \tag{2.3}$$

де T_r – час прийому даних, мс;

T_s – час відправки даних, мс.

Для прикладу візмемо умови роботи сервісу обробки даних з таблиці 2.9. T_s дорівнює 0, так як цей параметр фіксується відносно певного часу, тому припустимо, що різниці між фіксованим часом та T_s немає. В мовах програмування ця фіксація підтримується.

Таблиця 2.9 – Значення часу відправки та прийому даних при кількості підключених користувачів та роботів до сервісу обробки даних

№	Кількість підключених користувачів	Кількість підключених роботів	T_s 1-го робота, мс	T_r 1-го користувача 1-го робота, мс
1	2	3	4	5
1	1	1	0	50
2	1	10	0	100
3	10	10	0	200

Продовження таблиці 2.9

1	2	3	4	5
4	20	20	0	400
5	30	30	0	600

Тоді D для кожного сценарію дорівнює:

D_1 = 50 мс;

D_2 = 100 мс;

D_3 = 200 мс;

D_4 = 400 мс;

D_5 = 600 мс.

З цього можна зробити висновок, що при 30 підключених користувачах, що моніторять 30 роботів слід масштабувати компонент сервісу обробки даних.

Для виконання дослідів необхідно розробити додаткове відповідне програмне забезпечення, яке буде симулювати різні сценарії роботи системи.

Виділимо загальні сценарії, що повині симулюватись:

– відправка вказаної кількості станів робота користувачу;

– одночасне підключення вказаної кількості користувачів та роботів до системи з можливістю трансляції даних між ними.

Також для знімання параметрів дослідження, програмне забезпечення повинно показувати ці параметри.

2.8 Висновки до розділу 2

Декомпозиція проекту і виділення його меж показали, що система моніторингу містить 6 основних взаємопов’язаних компонентів, 4 з яких повинні бути реалізовані програмно.

Front-end частина буде розроблятись на платформі Node.js з використанням бібліотеки React, а back-end повинен базуватись на ASP .NET

Core. Для пришвидшення написання серверного коду в реалізації слід використовувати бібліотеку SignalR, що імплементує в собі інструменти асинхронного обміну повідомленнями між сервером та клієнтом.

Сховище слід розділити на 2 частини – СУБД та БДРЧ. СУБД буде зберігати дані, що використовує сервіс для функціювання. Для цього була вибраний засіб MySQL. Дані що будуть зберігатися в вибраній СУБД описують 6 сутностей. БДРЧ потрібна для потокового збереження даних станів, що відправляють роботи. Для цього було вибрано Redis. Дані в вибраній БДРЧ достатньо описати 6 парами ключ-значення. Комбінація використання СУБД та БДРЧ повинне забезпечувати збалансованість в швидкості та якості зберігання даних для гарантування ефективної роботи системи.

Рекомендовано, щоб апаратна частина бортової системи мобільних роботів використовувала одноплатні комп’ютери типу Raspberry Pi або Orange Pi, так як вони підтримують запуск програм високого рівня і мають доступ до даних низького рівня при своїх невеликих габаритах.

Після реалізації системи моніторингу згідно розробленої методики слід провести дослідження поведінки одного екземпляру Docker контейнера сервісу обробки даних в різних умовах роботи системи для визначення параметрів, що впливають на масштабування цього компоненту. Результати дослідження повинні надавати дані, які допоможуть попередньо оцінити об’єм ресурсів, які необхідні для стабільної роботи системи.

3 РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ СТАНІВ МОБІЛЬНИХ РОБОТІВ

3.1 Розробка web-сервісів

Першим реалізуємо сервіс авторизації. Для цього створимо .NET проект за допомогою середовища розробки Visual studio і підключимо всі необхідні залежності для роботи з MySQL та JWT авторизацією. Також створимо та підключемо проект, що представляє з себе бібліотеку, де будуть зберігатися загальний код для усіх сервісів. В результаті маніфест проекту AuthService.csproj має такий зміст:

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="7.0.11" />
    <PackageReference Include="MySql.EntityFrameworkCore" Version="7.0.5" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="..\Common\Common.csproj" />
  </ItemGroup>
</Project>
```

Після цього створимо класи для роботи з базою даних, а саме модель для таблиці User та DbContext для роботи з базою даних.

Для таблиці User модель має вигляд:

```
public class User
{
   public long Id { get; set; }
   public string Firstname { get; set; }
   public string? Lastname { get; set; }
   public string Username { get; set; }
   public string Password { get; set; }
}
```

DbContext містить в собі колекцію моделі User та її налаштування для бази даних:

```
using Microsoft.EntityFrameworkCore;
using MrMonitor.AuthService.Models.Db;

namespace MrMonitor.AuthService;

public class AuthDbContext : DbContext
{
   public AuthDbContext(DbContextOptions<AuthDbContext> options) :
        base(options)
   {
      Database.EnsureCreated();
   }

   public DbSet<User> Users { get; set; }

   protected override void OnModelCreating(ModelBuilder modelBuilder)
   {
      modelBuilder.Entity<User>().HasKey(u => u.UserId);
      modelBuilder.Entity<User>().Property(u => u.Username).IsRequired();
```

```
        modelBuilder.Entity<User>().HasIndex(u => u.Username).IsUnique();
        modelBuilder.Entity<User>().Property(u => u.Firstname).IsRequired();
        modelBuilder.Entity<User>()
          .Property(u => u.Lastname)
          .HasDefaultValue(null);
        modelBuilder.Entity<User>().Property(u => u.Password).IsRequired();
    }
}
```

Коли робота з базою даних реалізована, створимо API контролер з методами, що описані в таблиці 2.6. З кодом контролера можна ознайомитись в додатку Б.

Для запуску сервісу використаємо код в Program.cs:

```
using MrMonitor.AuthService.Extensions;
using MrMonitor.Common.Extensions;

var builder = WebApplication.CreateBuilder(args);

var configuration = builder.Configuration;

builder.Services.AddAuthDbContext(configuration.GetSection("Database"));
builder.Services.AddAuthStores();
builder.Services.AddAuth(configuration.GetSection("Auth"));
builder.Services.AddAuthOptions(configuration.GetSection("Auth"));
builder.Services.AddControllers();

var app = builder.Build();

app.UseCors(builder =>
{
    builder.AllowAnyMethod();
```

```
    builder.AllowAnyHeader();
    builder.SetIsOriginAllowed(origin => true);
    builder.AllowCredentials();
});
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

В алгоритмі запуску ми створюємо будівельника web-програми, додаємо в його сервіси всі необхідні залежності, що використовуються в різних частинах проект, будуємо програму і ініціюємо її запуск.

По аналогії реалізуємо сервіс обробки даних. З кодом можна ознайомитись в додатку Б.

3.2 Розробка web-додатку

Для створення проекту використаємо команду npx create-react-app. Код web-додатку наведено в додатку Б.

В результаті реалізації, web-додаток має такий перелік сторінок:

– сторінка авторизації;

– сторінка створення облікового запису користувача;

– головна сторінка;

– сторінка редагування облікового запису користувача;

– сторінка створення облікового запису робота;

– сторінка редагування облікового запису робота.

Web-додаток в реальному часі може обмінюватись повідомленнями з роботами і моніторити їх стан в текстовому вигляді та в вигляді графіків.

Зовнішній вигляд web-додатку представлений на рисунках 3.1 – 3.11.

Рисунок 3.1 – Сторінка авторизації користувача

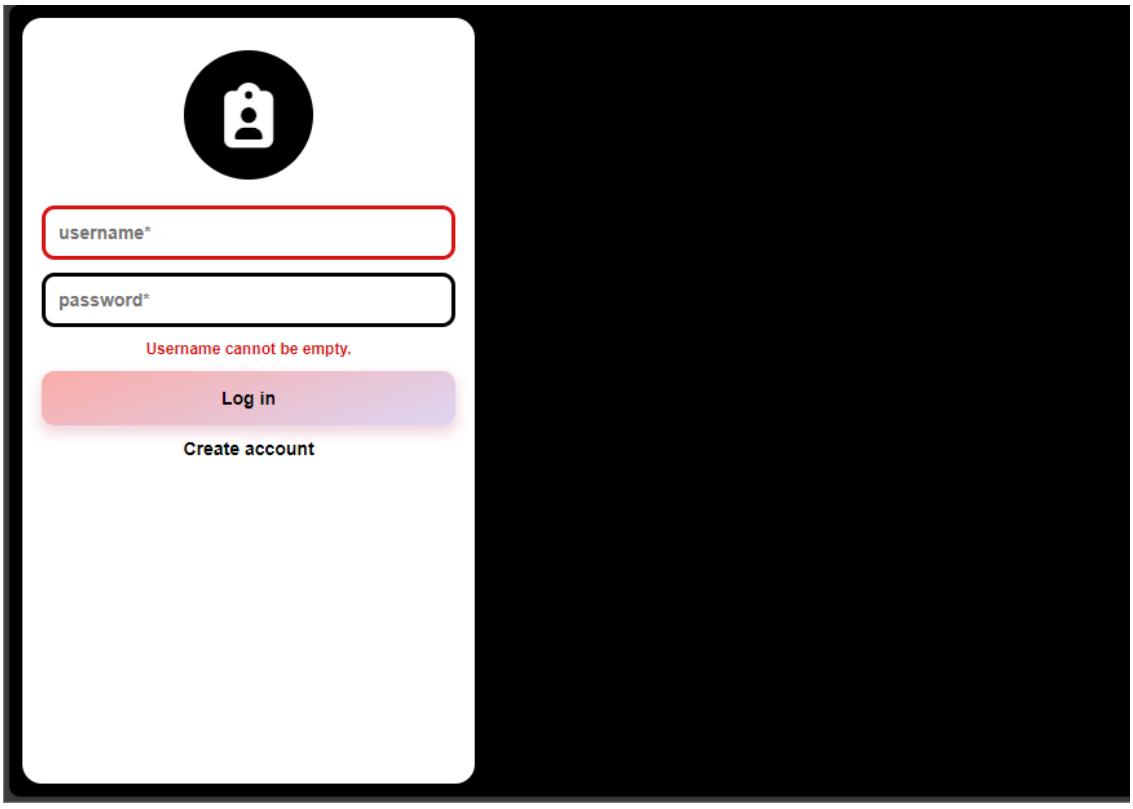

Рисунок 3.2 – Приклад валідації полів форми

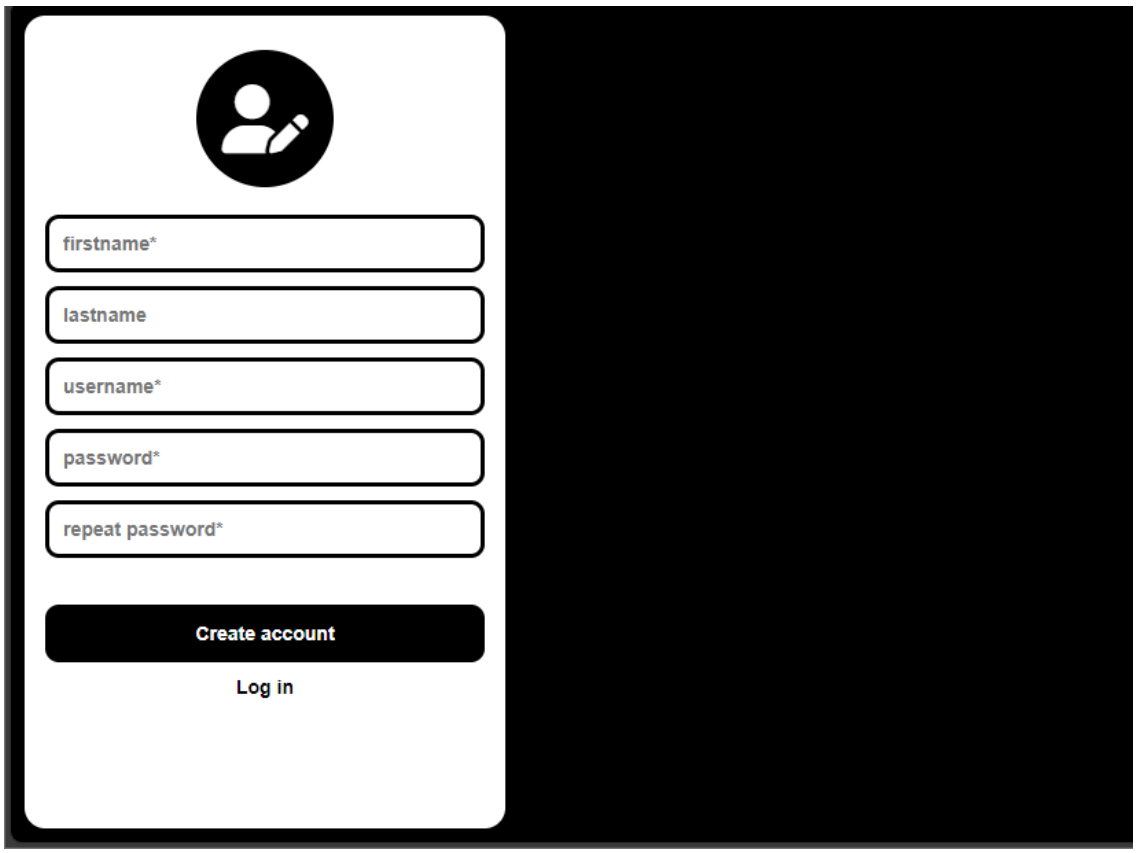

Рисунок 3.3 – Сторінка створення облікового запису користувача

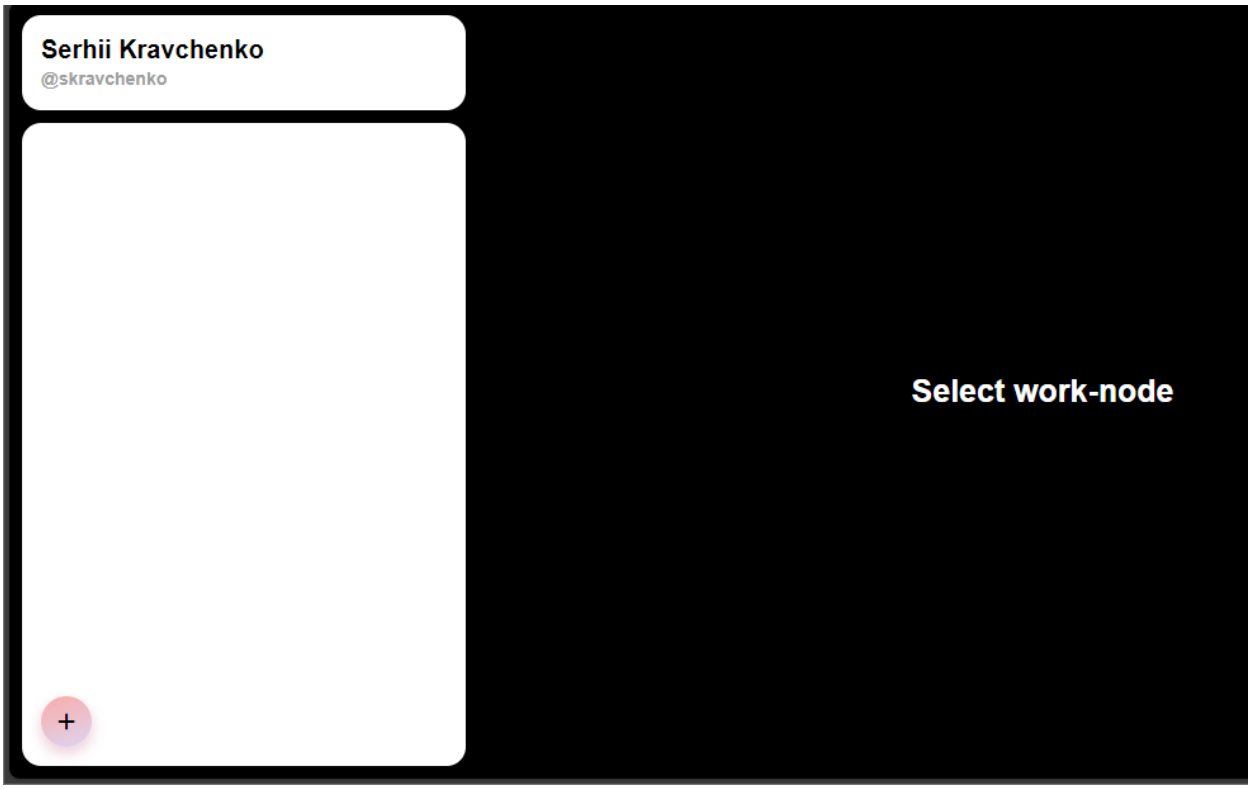

Рисунок 3.4 – Головна сторінка

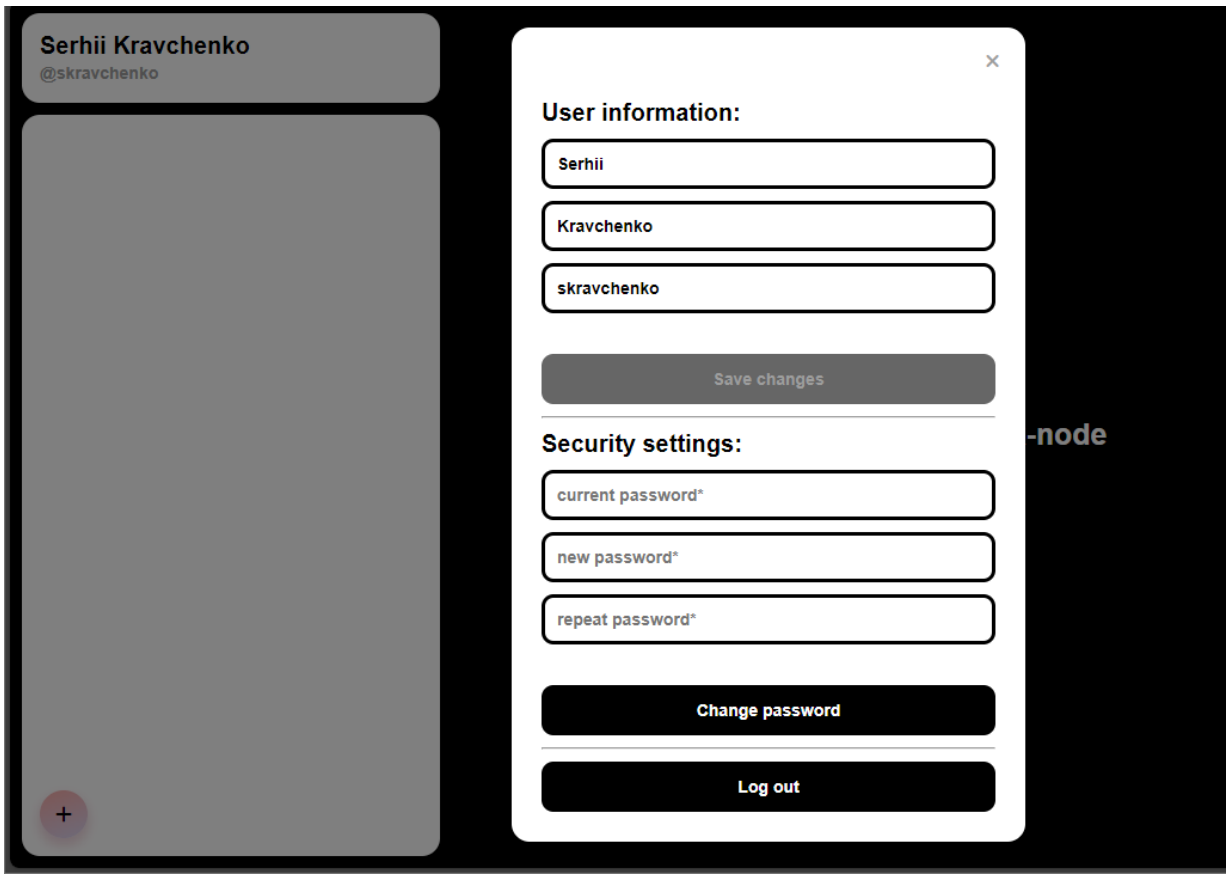

Рисунок 3.5 – Сторінка налаштування облікового запису користувача

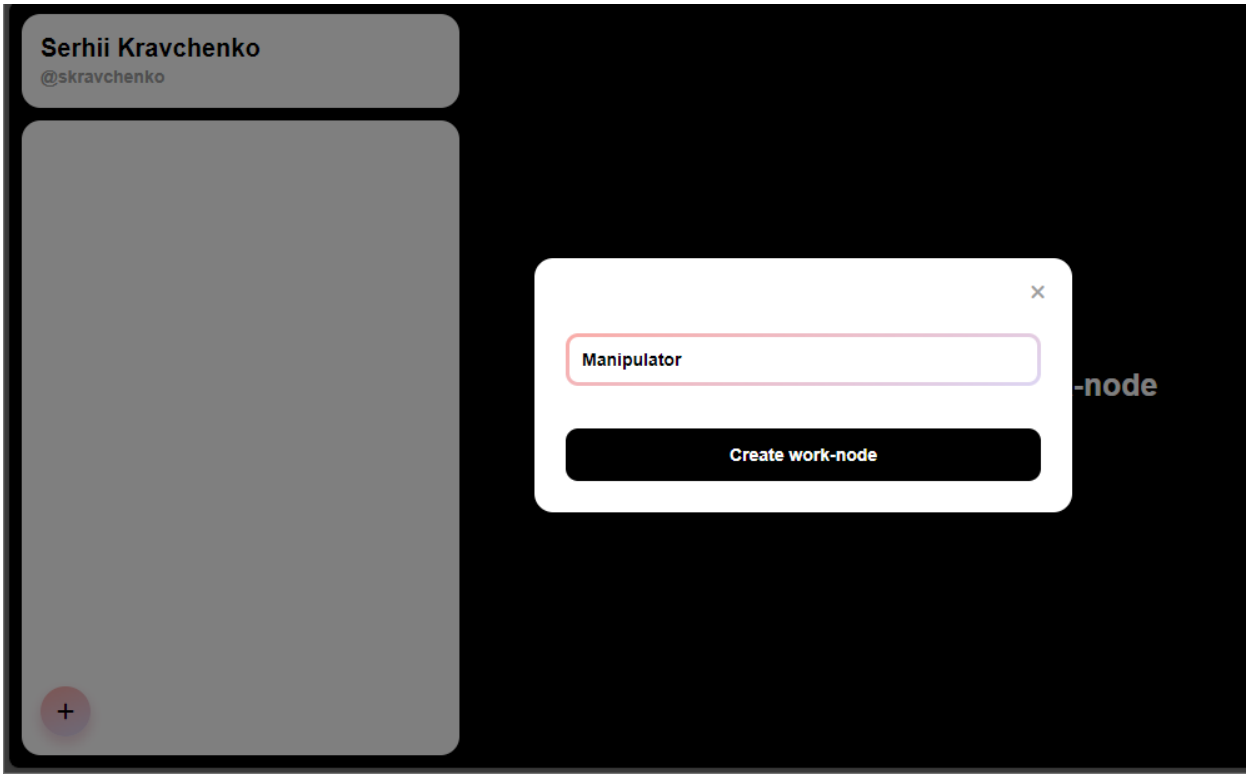

Рисунок 3.6 – Сторінка створення облікового запису робота

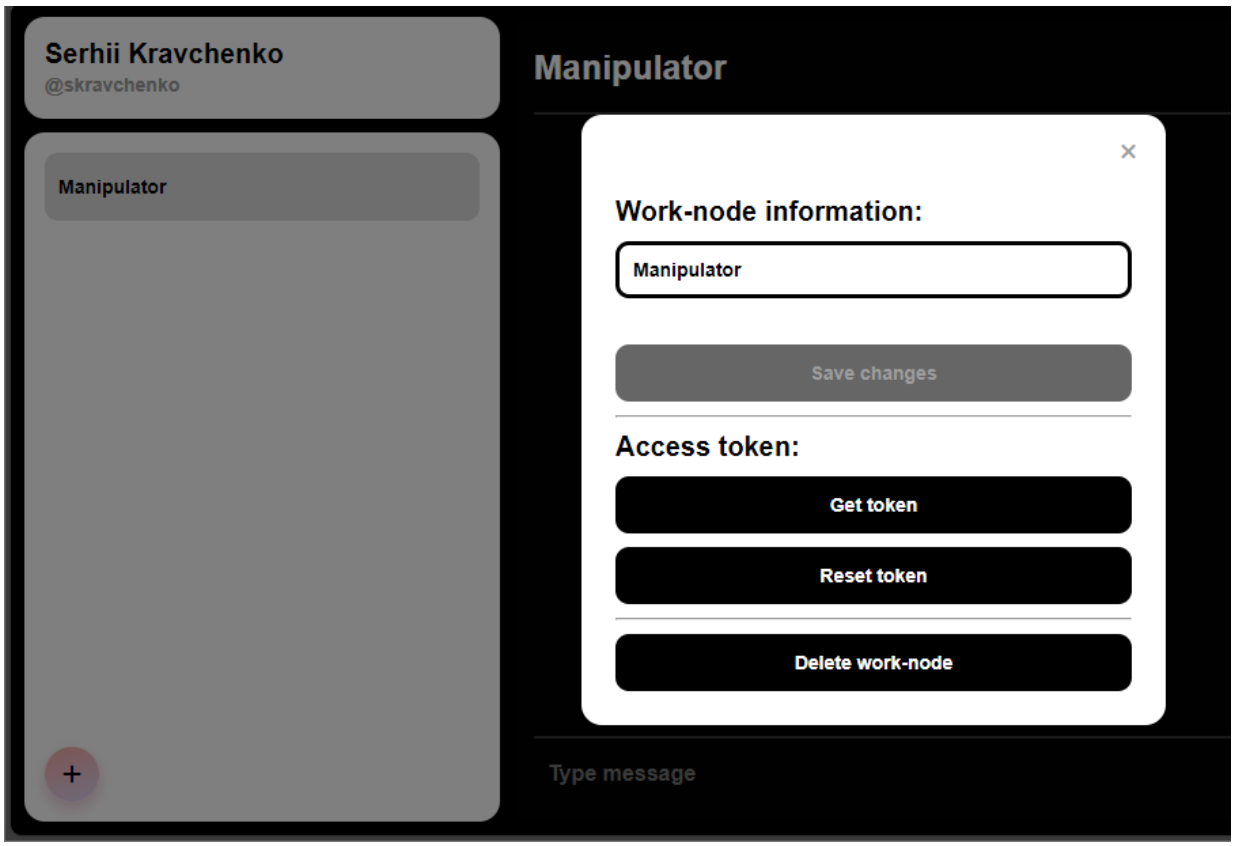

Рисунок 3.7 – Сторінка налаштування облікового запису робота

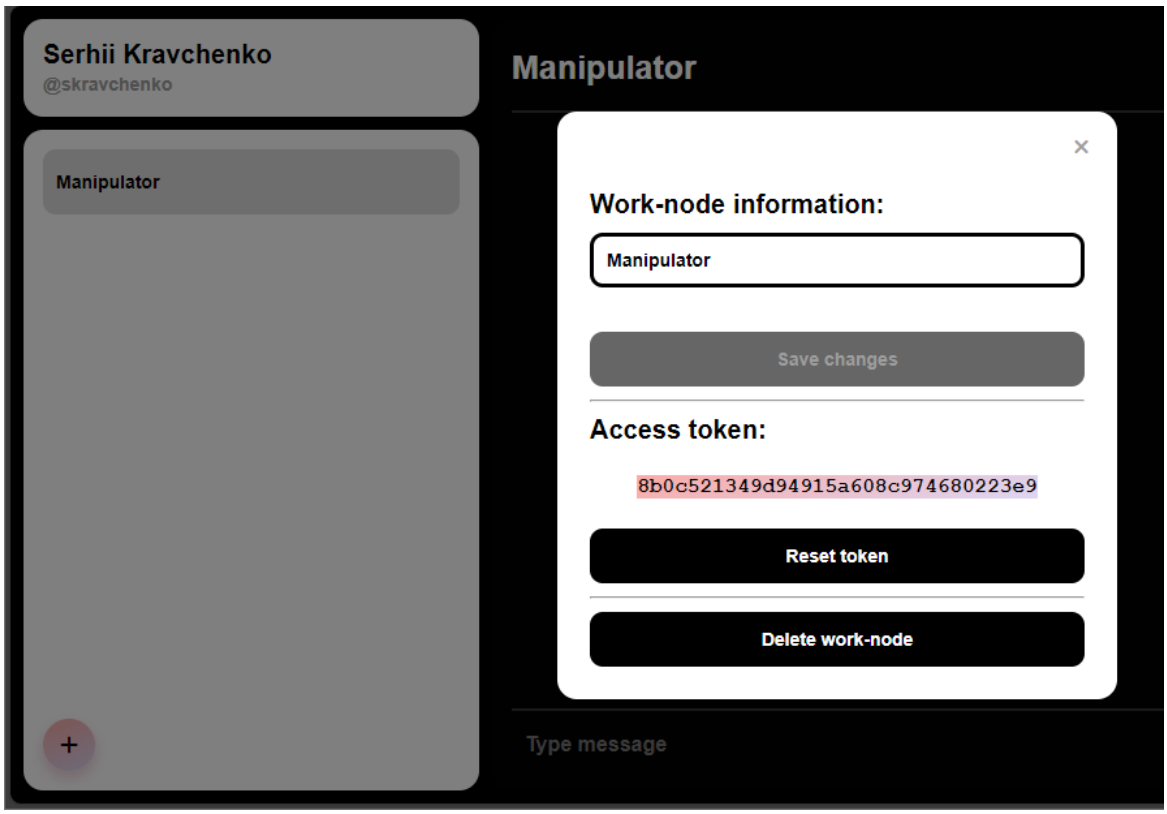

Рисунок 3.8 – Приклад отримання токену доступа до облікового запису робота для його бортової системи

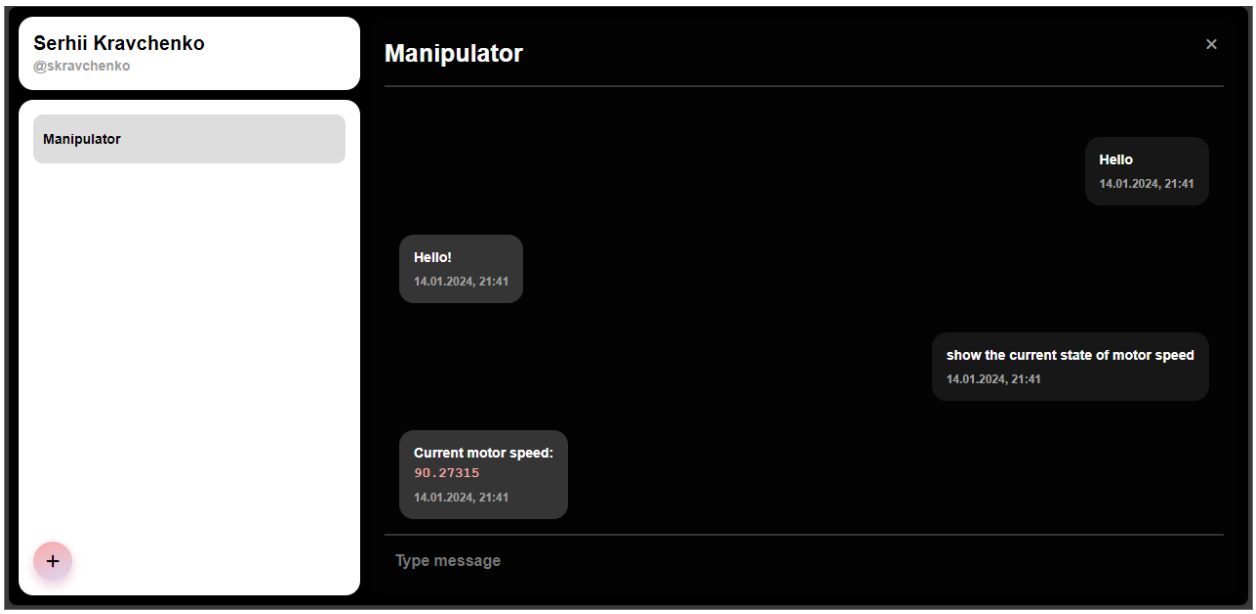

Рисунок 3.9 – Приклад отримання поточного стану робота за допомогою листування з ним

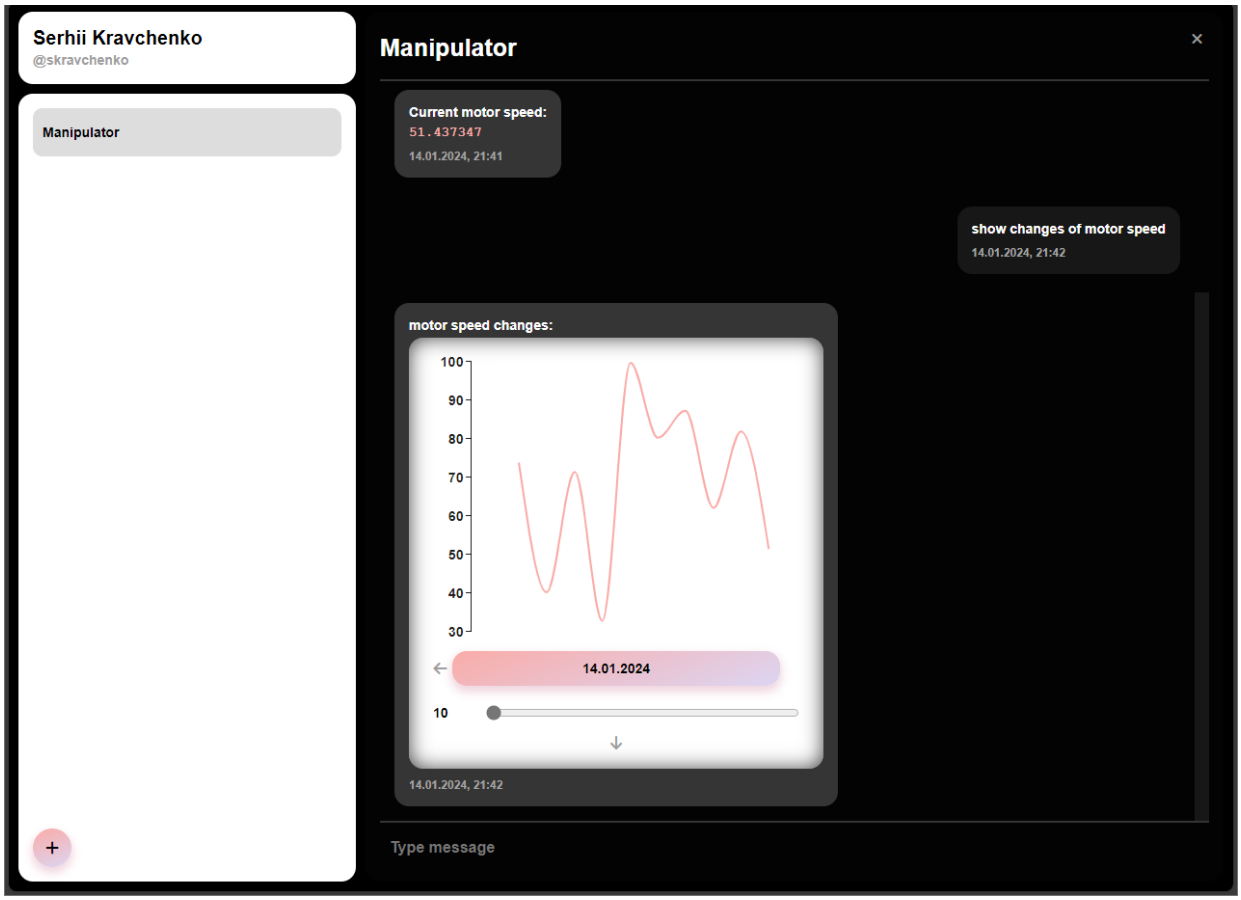

Рисунок 3.10 – Приклад отримання змін стану робота в вигляді графіків за допомогою листування з ним

Рисунок 3.11 – Приклад виводу розширених даних про зміни стану робота

3.3 Розгортання системи в Docker

Процес розгортання системи є важливим етапом, що потребує аналізу та планування з врахуванням особливостей середовища, де розгортається система та самої системи.

При розгортанні в Docker потрібно враховувати мережеві властивості контейнерів. Кожен контейнер ізольовано один від одного, що означає, що вони не знають про існування один одного, але вони можуть спілкуватися між собою, якщо їх об’єднати в одну внутрішню мережу в Docker. Якщо контейнери в одній мережі, то до них можна звернутися за посиланням, типу http://назва_контейнеру. Також контейнери ізольовані від загального середовища де запущено Docker. Це означає, що доступ до web-додатку та до API сервісів є неможливим без додаткових налаштувань. Для вирішення цієї проблеми слід відкрити порти контейнера при його запуску.

Для розгортання в Docker, спочатку додамо в кожен компонент системи Dockerfile, за допомогою якого буде зібрано образ компоненту. На основі образу буде створено екземпляр контейнеру.

Dockerfile для сервісу авторизації та сервісу обробки даних буде мати однакову структуру з різними назвами директорій та виконувальних файлів:

```
# Сервіс авторизації
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /source
COPY ./AuthService ./AuthService
COPY ./Common ./Common
RUN dotnet publish ./AuthService/AuthService.csproj -c release -o /build
FROM mcr.microsoft.com/dotnet/aspnet:7.0
WORKDIR /service
COPY --from=build /build ./
EXPOSE 80
ENV ASPNETCORE_URLS=http://+:80
ENTRYPOINT ["dotnet", "AuthService.dll"]

# Сервіс обробки даних
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /source
COPY ./WorkerService ./WorkerService
COPY ./Common ./Common
RUN dotnet publish ./WorkerService/WorkerService.csproj -c release -o /build
FROM mcr.microsoft.com/dotnet/aspnet:7.0
WORKDIR /service
COPY --from=build /build ./
EXPOSE 80
ENV ASPNETCORE_URLS=http://+:80
ENTRYPOINT ["dotnet", "WorkerService.dll"]
```

Цей образ будується за допомогою декількох образів. Один збирає .NET проект, а інший запускає його. Також в образах відкритий 80 порт і встановлена змінна середовища, що налаштовує роботу API сервісів на цей порт.

Dockerfile для web-додатку має іншу структуру:

```
FROM node:18 AS build
WORKDIR /app
COPY ./package*.json ./
RUN npm install
COPY ./public ./public
COPY ./src ./src
RUN npm run build
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Тут також використовуються декілька образів для збирання проекту та запуску його в web-сервері Nginx, використовуючи 80 порт.

Для MySQL та Redis використаємо готові образи.

Після цього розробимо CMD скрипт для автоматизованого збирання компонентів та запуску їх в Docker:

```
echo off

# Збирання образів компонентів
docker build -t auth-service -f ./service/src/AuthService/Dockerfile ./service/src/

docker build -t worker-service -f ./service/src/WorkerService/Dockerfile ./service/src/
```

```
docker build -t app -f ./app/Dockerfile ./app/

# Очищення поточних ресурсів
docker network rm mr-monitor-network

docker kill mysql redis auth-service worker-service app

docker rm mysql redis auth-service worker-service app

# Створення мережі для системи
docker network create mr-monitor-network

# Запуск контейнерів
docker run -d --restart always --network mr-monitor-network --name mysql -e MYSQL_ROOT_PASSWORD=pass -p 3306:3306 mysql:8.1.0

docker run -d --restart always --network mr-monitor-network --name redis -p 6379:6379 -p 8001:8001 redis/redis-stack

docker run -d --restart always --network mr-monitor-network --name auth-service -p 5001:80 auth-service

docker run -d --restart always --network mr-monitor-network --name worker-service -p 5002:80 worker-service

docker run -d --restart always --network mr-monitor-network --name app -p 3000:80 app
```

Схема інфраструктури, що розгортається розробленим скриптом представлена на рисунку 3.12.

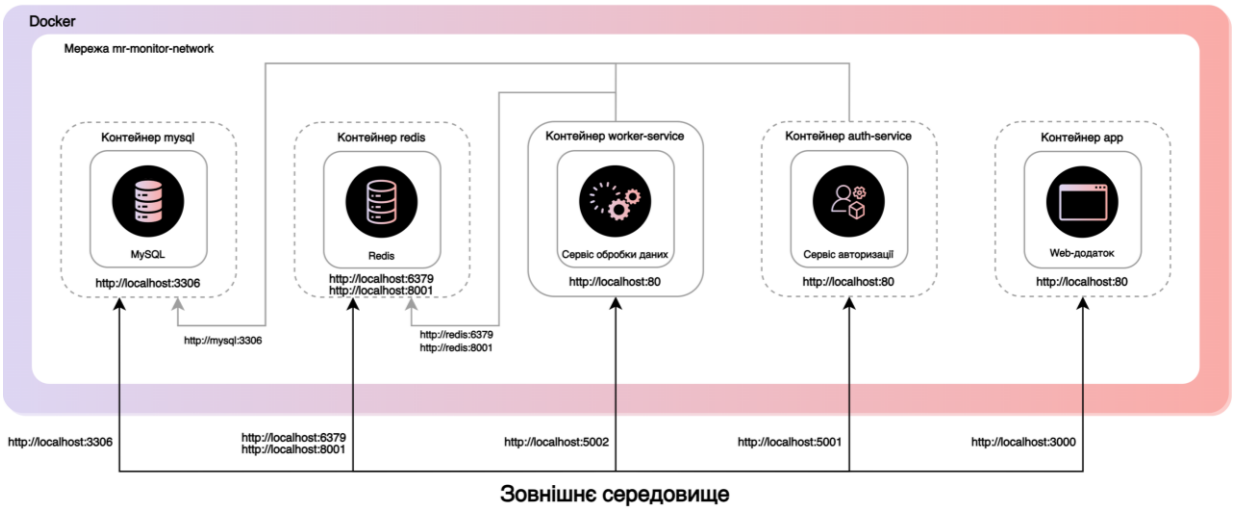

Рисунок 3.12 – Схема розгорнутої інфраструктури скриптом

3.4 Висновки до розділу 3

Всі компоненти системи успішно реалізовані і протестовані. Результат задовільняє поставлену мету і виконує всі вимоги, які були виділені в розділі 2.1.

В ході перевірки, сервіси авторизації та обробки даних показали задовільну роботу. Трансляція даних в web-додаток виконувалась з високою швидкістю і всі дані було збережено з використанням механізму, що описувався в розділі 2.3.

Web-додаток має зручний та ергономічний інтерфейс в вигляді мессенджера, за допомогою якого можна взаємодіяти з роботом шляхом листування. Додаток вміє виводити текстову та графічну інформацію від робота. Основні функції системи моніторингу розроблений web-додаток виконує.

Скрипт для автоматичного розгортання також вийшов зручним, так як не потрібно виконувати багато дій. Розгорнута інфраструктура дозволить проводити дослідження над системою, не потребуючи використання великої кількості запущених програм.

Проект повністю готовий до експериментальних досліджень.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Для проведення досліджень спочатку реалізуємо емулятор, що дозволить зробити умови роботи розробленої системи моніторигну. Створимо новий консольний .NET проект і підключемо до нього проекти компонентів системи для того, щоб мати доступ до типів даних, що використовуються компонентами. Після цього розробимо алгоритм для додавання заданої кількості користувачів та роботів в базу даних:

```
int count = int.Parse(args[0]);
using (var dbContext = new AuthDbContext())
{
        for (int i = 0; i < count; i++)
        {
                dbContext.Users.Add(new User
                {
                        Firstname = "TEST",
                        Password = "password",
                        Username = $"test{i}",
                });
      }
      await dbContext.SaveChangesAsync();
}
using (var dbContext = new WorkerDbContext())
{
        for (int i = 0; i < count; i++)
        {
                dbContext.WorkNodes.Add(new WorkNode
                {
                        Name = $"test{i}",
                        Token = $"{i}",
```

```
                UserWorkNodes = new UserWorkNode[]
                {
                    new UserWorkNode
                    {
                        UserId = i,
                        Permission = WorkNodePermission.Creator
                    }
                }
            });
        }
        await dbContext.SaveChangesAsync();
}
```

В цьому алгоритмі створюються DbContext з сервісу авторизації та сервісу обробки даних і за допомогою них додаються відповідні записи в базу даних.

Додавши записи в базу даних, згенеруємо ключі доступу для підключення до сервісу обробки даних:

```
int count = int.Parse(args[0]);
var httpClient = new HttpClient();
var userJwts = new List<string>();
var workNodeJwts = new List<string>();
 for (int i = 0; i < count; i++)
 {
        var request = new HttpRequestMessage(
            HttpMethod.Post,
            "http://localhost:5001/auth/login");
        request.Content = JsonContent.Create(new LoginRequest
        {
            Username = $"test{i}",
            Password = "password"
```

```
        });
        var response = httpClient.Send(request);
        var userJwt = response.Content.ReadAsStringAsync().Result;
        userJwts.Add(userJwt);
        request = new HttpRequestMessage(
                HttpMethod.Get,
                $"http://localhost:5002/worker/getWorkNodeAccessToken/{i}");
        response = httpClient.Send(request);
        var workNodeJwt = response.Content.ReadAsStringAsync().Result;
        workNodeJwts.Add(workNodeJwt);
        Console.WriteLine(i);
}
File.WriteAllLines("user-jwt", userJwts);
File.WriteAllLines("work-node-jwt", workNodeJwts);
```

Цей алгоритм відправляє http-запити в сервіс, використовуючи дані, які було додано в базу даних на попередньому етапі, і записує отримані ключи для користувачів та роботів в спеціальні файли в вигляді списку.

Далі напишемо алгоритм, що в автоматичному режимі буде підключати користувачів до системи моніторингу:

```
var userJwt = File.ReadAllLines("user-jwt");
for (int i = 0; i < userJwt.Length; i++)
{
        var connection = new HubConnectionBuilder()
                .WithUrl("http://localhost:5002/hubs/chat", ops =>
                {
                        ops.AccessTokenProvider = () => Task.FromResult(userJwt[i]);
                })
                .WithAutomaticReconnect()
                .Build();
```

```
        try
        {
                connection.StartAsync().Wait();
                Console.WriteLine($"{i} {userConnection.State}");
        }
         catch (Exception ex)
        {
                Console.WriteLine(ex.Message);
        }
}
```

Цей алгоритм можна використовувати також для роботів.

Тепер визначимо параметр U_{max}. Результати показали, що максимальна кількість користувачів, що можуть бути підключеними до одного екземпляру сервіса обробки даних дорівнює 255. Результат виконання програми представлено на рисунку 4.1.

```
243 Connected
244 Connected
245 Connected
246 Connected
247 Connected
248 Connected
249 Connected
250 Connected
251 Connected
252 Connected
253 Connected
254 Connected
255 Connected
One or more errors occurred. (Unable to complete handshake with the server due to an error: Handshake was canceled.)
```

Рисунок 4.1 – Результат тестування підключення користувачів до одного екземпляру сервісу обробки даних

Тепер подивимось на використання оперативної пам’яті кожним компонентом в піку навантажень. За допомогою наступного алгоритму можна навантажити Redis, відправляючи псевдостани робота, що згенеровані генератором випадкових чисел, користувачу:

```
while (true)
{
```

```
        var stateKey = "motor speed";
        var stateValue = (float)(random.NextDouble() * 100);

        await connection.SendAsync("OnUpdate", new Update
        {
                Type = UpdateType.Frame,
                SenderType = SenderType.WorkNode,
                 Frame = new Frame
                {
                        States = new[]
                        {
                                new State
                                {
                                        Key = stateKey,
                                        Value = stateValue,
                                        DateTime = DateTime.UtcNow
                                }
                        }
                 }
        });
}
```

Наведений алгоритм симулює відправку роботом стану користувачу з максимально можливим періодом відправки. Сервіс обробки даних почне зберігати ці стани в Redis, тим самим спричинить навантаження на нього.

Інформацію про використання ресурсів в Docker можна за допомогою команди docker stats.

Результат виводу максимольного використання оперативної пам’яті кожним компонентом системи представлено на рисунках 4.2 – 4.6.

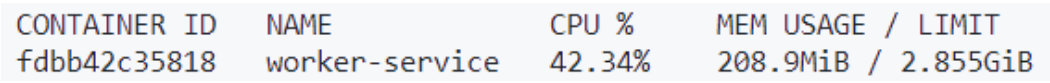

Рисунок 4.2 – Максимальне використання оперативної пам’яті сервісом обробки даних

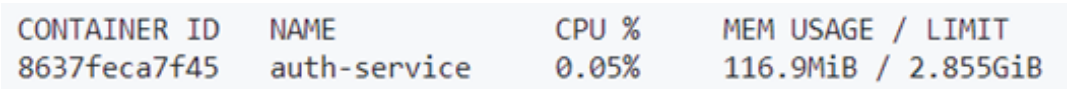

Рисунок 4.3 – Максимальне використання оперативної пам’яті сервісом авторизації

Рисунок 4.4 – Максимальне використання оперативної пам’яті web-додатком

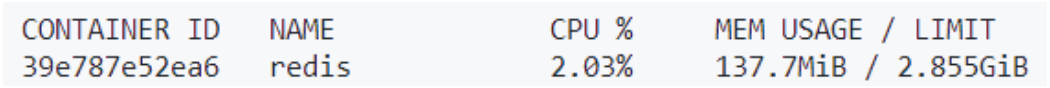

Рисунок 4.5 – Максимальне використання оперативної пам’яті Redis

Рисунок 4.6 – Максимальне використання оперативної пам’яті MySQL

Тепер розрахуємо рекомендовану ємність оперативної пам’яті за формулою (2.1), взявши кількість екземплярів кожного компоненту 1:

$$C = (208{,}9 \cdot 1{,}5) + (116{,}9 \cdot 1{,}5) + (4{,}75 \cdot 1{,}5) + (137{,}7 \cdot 1{,}5) + (289{,}9 \cdot \cdot 1{,}5) = 313{,}35 + 175{,}35 + 7{,}13 + 206{,}55 + 434{,}85 = 1137{,}23 \text{ МБ}.$$

Наступним дослідженням знайдемо середню затримку передачі даних роботом одному користувачу. Для цього використаємо алгоритм для відправки станів роботом користувачу і подивимось вивід інформації на стороні емулятора та в web-браузері, де запущено web-додаток.

Результати дослідження представлені на рисунках 4.7 – 4.9.

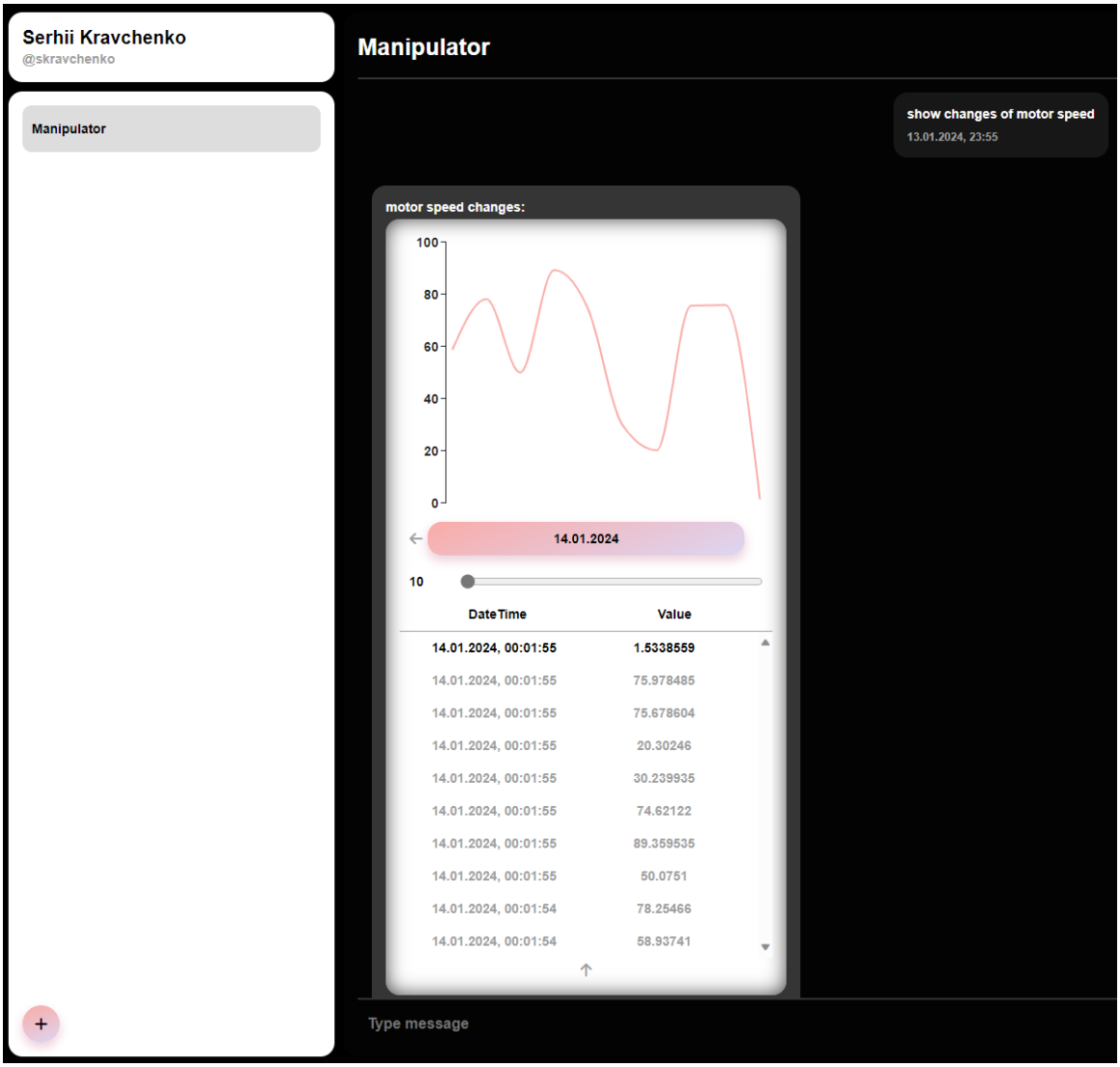

Рисунок 4.7 – Результат отримання даних від емулятору в web-додатку

```
98. Received: 58.93741 (Time: 1, 54, 835)
99. Received: 78.25466 (Time: 1, 54, 939)
100. Received: 50.0751 (Time: 1, 55, 48)
101. Received: 89.359535 (Time: 1, 55, 157)
102. Received: 74.62122 (Time: 1, 55, 264)
103. Received: 30.239935 (Time: 1, 55, 371)
104. Received: 20.30246 (Time: 1, 55, 485)
105. Received: 75.678604 (Time: 1, 55, 588)
106. Received: 75.978485 (Time: 1, 55, 696)
107. Received: 1.5338559 (Time: 1, 55, 804)
```

Рисунок 4.8 – Вивід інформації про отримання даних в web-браузері

```
98. Sended motor speed: 58,93741 (Time: 1, 54, 819)
99. Sended motor speed: 78,25466 (Time: 1, 54, 926)
100. Sended motor speed: 50,0751 (Time: 1, 55, 33)
101. Sended motor speed: 89,359535 (Time: 1, 55, 140)
102. Sended motor speed: 74,62122 (Time: 1, 55, 247)
103. Sended motor speed: 30,239935 (Time: 1, 55, 356)
104. Sended motor speed: 20,30246 (Time: 1, 55, 464)
105. Sended motor speed: 75,678604 (Time: 1, 55, 572)
106. Sended motor speed: 75,978485 (Time: 1, 55, 681)
107. Sended motor speed: 1,5338559 (Time: 1, 55, 790)
```

Рисунок 4.9 – Вивід інформації про відправку даних в емуляторі

Тепер напишемо простий алгоритм, який порахує параметр D за формулою (2.3):

```
var a = new DateTime[]
{
    new DateTime(2024, 1, 1, 0, 1, 54, 835),
    new DateTime(2024, 1, 1, 0, 1, 54, 939),
    new DateTime(2024, 1, 1, 0, 1, 55, 48),
    new DateTime(2024, 1, 1, 0, 1, 55, 157),
    new DateTime(2024, 1, 1, 0, 1, 55, 264),
    new DateTime(2024, 1, 1, 0, 1, 55, 371),
    new DateTime(2024, 1, 1, 0, 1, 55, 485),
    new DateTime(2024, 1, 1, 0, 1, 55, 588),
    new DateTime(2024, 1, 1, 0, 1, 55, 696),
    new DateTime(2024, 1, 1, 0, 1, 55, 804),
};
var b = new DateTime[] {
    new DateTime(2024, 1, 1, 0, 1, 54, 819),
    new DateTime(2024, 1, 1, 0, 1, 54, 926),
    new DateTime(2024, 1, 1, 0, 1, 55, 33),
    new DateTime(2024, 1, 1, 0, 1, 55, 140),
    new DateTime(2024, 1, 1, 0, 1, 55, 247),
    new DateTime(2024, 1, 1, 0, 1, 55, 356),
```

```
        new DateTime(2024, 1, 1, 0, 1, 55, 464),
        new DateTime(2024, 1, 1, 0, 1, 55, 572),
        new DateTime(2024, 1, 1, 0, 1, 55, 681),
        new DateTime(2024, 1, 1, 0, 1, 55, 790),
};
for (int i = 0; i < 10; i++)
{
        Console.WriteLine((a[i] - b[i]).TotalMilliseconds);
}
```

Результати розрахунку представлені таблиці 4.1.

Таблиця 4.1 – Результати розрахунку D при мінімальному навантажені на систему

Пакет даних	1	2	3	4	5	6	7	8	9	10
D, мс	16	13	15	17	17	15	21	16	15	14

Результати показали, що всі отримані пакети надішли з затримкою, яка відповідає умові (2.2), тобто були менше 500 мс.

Тепер повторимо це дослідження, але в умовах повного навантаження на систему. За допомогою наведених вище алгоритмів підкючемо максимальну кількість дозволених користувачів та роботів, що будуть надсилати дані. Максимальна кількість підключень становить 255. З цих підключень 2 виділимо під емулятор та web-браузер. Виходить що нам необхідно підключити 126 користувачів та 126 роботів для відтворення такого сценарію.

Результати представлені на рисунках 4.10 – 4.12.

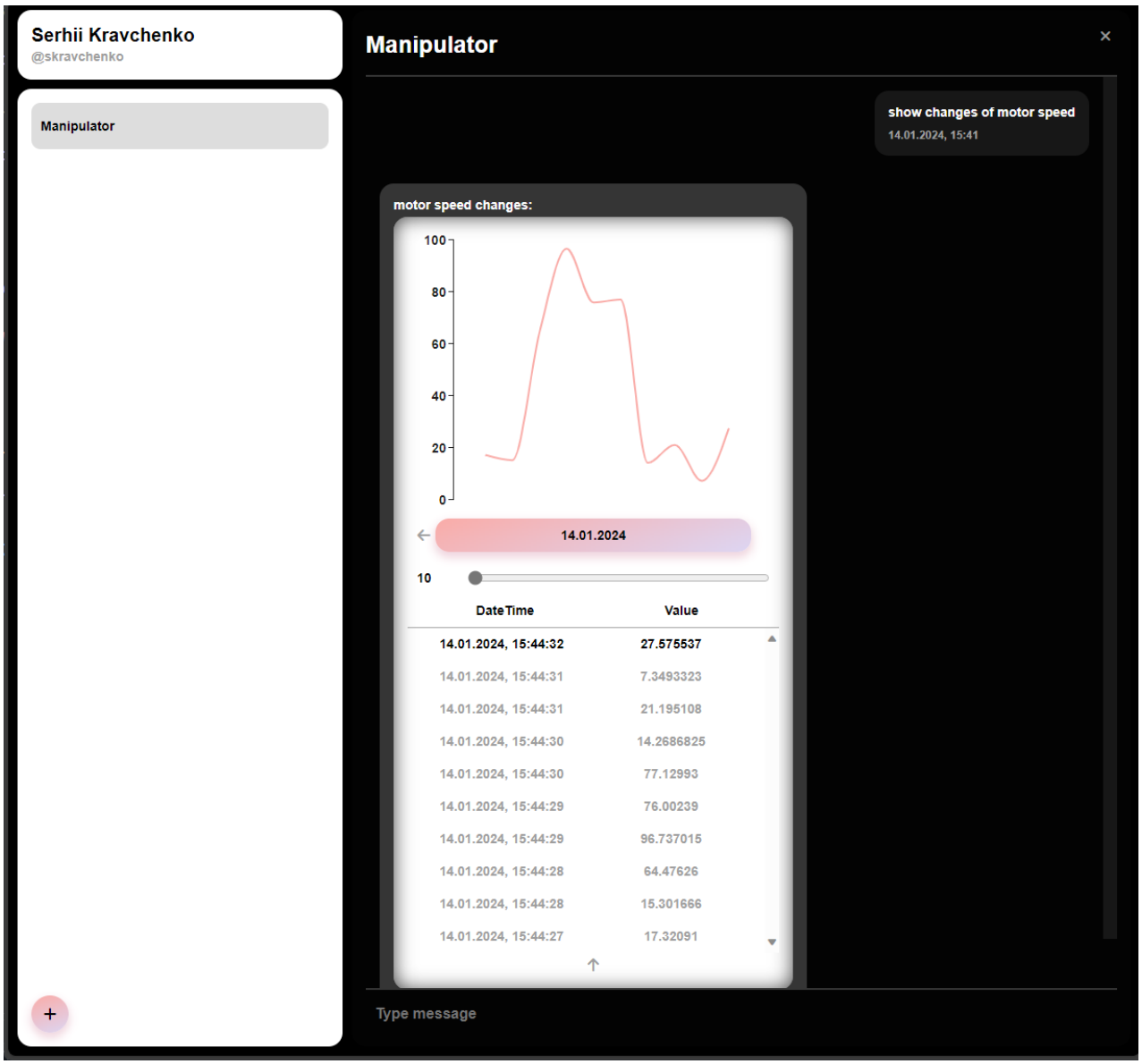

Рисунок 4.10 – Результат отримання даних від емулятору в web-додатку

```
97. Received: 17.32091 (Time: 44, 27, 604)
98. Received: 15.301666 (Time: 44, 28, 120)
99. Received: 64.47626 (Time: 44, 28, 635)
100. Received: 96.737015 (Time: 44, 29, 151)
101. Received: 76.00239 (Time: 44, 29, 662)
102. Received: 77.12993 (Time: 44, 30, 177)
103. Received: 14.2686825 (Time: 44, 30, 691)
104. Received: 21.195108 (Time: 44, 31, 202)
105. Received: 7.3493323 (Time: 44, 31, 722)
106. Received: 27.575537 (Time: 44, 32, 235)
```

Рисунок 4.11 – Вивід інформації про отримання даних в web-браузері

```
97. Sended motor speed: 17,32091 (Time: 44, 27, 597)
98. Sended motor speed: 15,301666 (Time: 44, 28, 111)
99. Sended motor speed: 64,47626 (Time: 44, 28, 626)
100. Sended motor speed: 96,737015 (Time: 44, 29, 140)
101. Sended motor speed: 76,00239 (Time: 44, 29, 653)
102. Sended motor speed: 77,12993 (Time: 44, 30, 169)
103. Sended motor speed: 14,2686825 (Time: 44, 30, 682)
104. Sended motor speed: 21,195108 (Time: 44, 31, 194)
105. Sended motor speed: 7,3493323 (Time: 44, 31, 710)
106. Sended motor speed: 27,575537 (Time: 44, 32, 224)
```

Рисунок 4.12 – Вивід інформації про відправку даних в емуляторі

Розраховані D представлені в таблиці 4.2.

Таблиця 4.2 – Результати розрахунку D при максимальному навантажені на систему

Пакет даних	1	2	3	4	5	6	7	8	9	10
D, мс	7	9	9	11	9	8	9	8	12	11

Отримані значення D також відповідають умові (2.2).

Побудуємо графіки з отриманими значеннями D при максимальному та мінімальному навантажені на систему (рисунок 4.13).

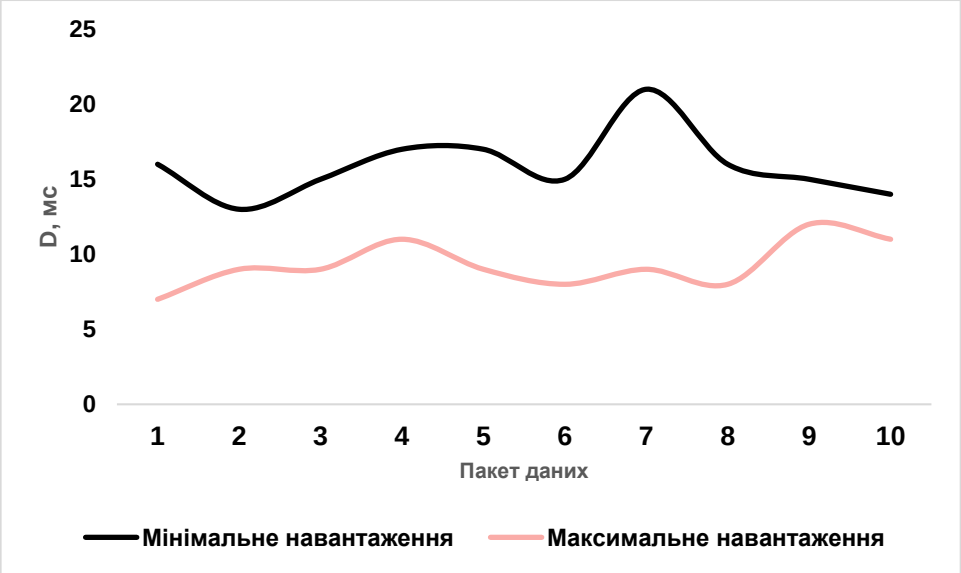

Рисунок 4.13 – Графіки зміни параметру D при мінімальних та максимальних навантаженнях

На графіках видно, що значення затримки при максимальних навантаженнях менші ніж при мінімальних. Якщо побудувати ці графіки, взявши діапазон не мілісекунди, а секунди (рисунок 4.14), то можна побачити, що це майже одна лінія з деякими похибками. З цього випливає висновок, що при мінімальних та максимальних навантаженнях, якість трансляції даних залишається незмінною. А це означає, що розроблена система моніторингу парцює стабільно в різних умовах.

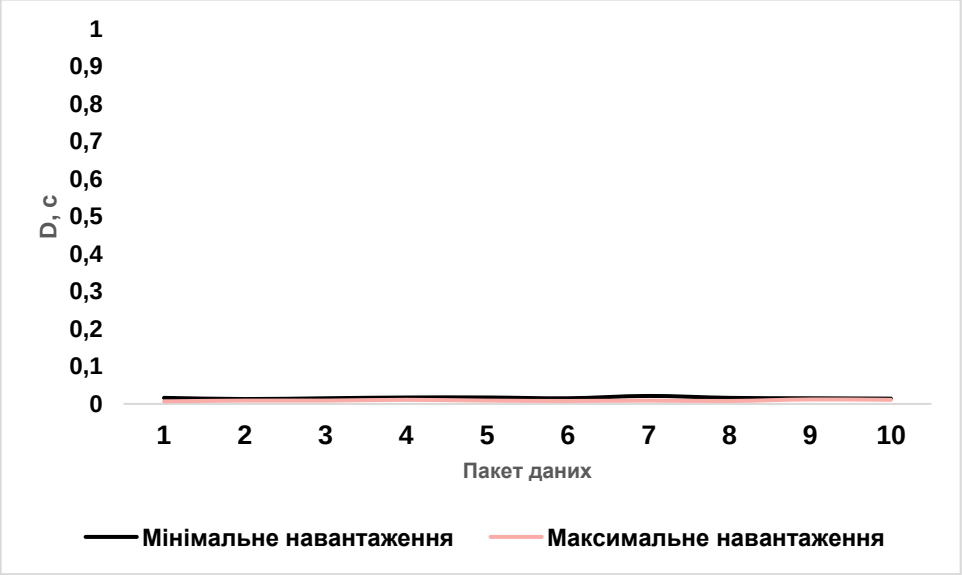

Рисунок 4.14 – Графіки зміни параметру D при мінімальних та максимальних навантаженнях в рамках секунди

4.1 Висновки до розділу 4

В ході проведення досліджень було визначено, що до екземпляру контейнера сервісу обробки даних можна підключити 255 користувачів одночасно, після чого потрібно запускати новий екземпляр. Також встановлено, що при мінімальних та максимальних навантаженях, якість передачі даних від робота до користувача, залишається стабільною і для нормальної роботи системи, рекомендований об’єм оперативної пам’яті на серверній машині становить 1137,23 МБ.

5 ОХОРОНА ПРАЦІ

Охорона праці має важливу роль в процесі розробки та досліджень, бо якими б не були цінними результати, вони не компенсують здоров’я та життя людини.

Згідно до санітарних норм мікроклімату виробничих приміщень ДНС (Державих санітарних норм) 3.3.6-042-99, легкі фізичні роботи, витрата енергії при якії становить 105 – 140 Вт (90 - 120 ккал/год.) що виконуються сидячі, мають категорію Іа [25]. Робота в приміщені з персональним комп’ютером відноситься до цієї категорії робіт. В таблиці 5.1 наведено оптимальні величини температури, відносної вологості та швидкості руху повітря в робочій зоні згідно категорії Іа.

Таблиця 5.1 – Оптимальні величини параметрів мікроклімату робочої зони для категорії робіт Іа

Параметр	Норми
1	2
Температура повітря, С°	22 – 24
Відносна вологість повітря, %	60 – 40
Швидкість руху повітря, м / с	0,1

Згідно ДБН В.2.5-28:2018, характеристика зорової роботи має розряд ІІІ [26]. В таблиці 5.2 наведено норми освітлення для цього розряду.

Таблиця 5.2 – Норми освітлення для ІІІ розряду зорової роботи

Параметр	Норми
1	2
Освітленість, лк	≥ 200
Коефіцієнт природнього освітлення (КПО), %	≥ 1,2

Виміряні параметри робочого приміщення наведені в таблиці 5.3.

Таблиця 5.3 – Виміряні параметри робочого приміщення

Параметр	Фактичне значення	В межах допусимого
1	2	3
Температура повітря, С°	22	+
Відносна вологість повітря, %	50	+
Швидкість руху повітря, м / с	0,1	+
Освітленість, лк	250	+
КПО, %	1,2	+

З результатів можна сказати, що приміщення в якому проводилась робота, повністю відповідає вимогам і здоров’я та життя людей збережено.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проаналізовано актуальність та проблематику теми, що пов’язана з розробленням автоматизованої системи моніторингу станів мобільних роботів.

В ході аналізу було розглянуто Індустрію 5.0 як напрям розвиту сучасних виробництв, сучасні технології, що використовуються на виробництвах, відомості про мобільних роботів і особливості сучасних систем моніторингу автоматизованих систем. Аналіз показав, що тема актуальна і на базі нього було розроблено методику розробки системи моніторингу.

В методиці розробки описано концепцію та виділено загальні вимоги до проекту. На основі концепції була побудована архітектура, що включала 6 компонентів для яких підібрано технічні засоби для реалізації. Також розроблений план досліджень системи після її реалізації.

Використовуючи розроблену методику, було успішно розроблено систему моніторингу в вигляді мессенджеру. Шляхом листування з роботом, можна моніторити його стан в реальному часі. Також успішно розглянуто і виконано процес розгортання системи, що є важливою складовою в роботі проекту.

Дослідження поведінки системи показали ефективну роботу розробленої системи моніторингу і вони перевершили очікувані результати. Проект працює стабільно при різних навантаженнях і для підтримки цього були отримані параметри, за допомогою яких можна оцінити потреби в ресурсах.

Всі поставлені цілі було досягнуто в повному обсязі. Результатом виконаної роботи задоволений. До додаткових функцій, що не було реалізовано відносяться вивід GPS координат та додавання прав доступу на робота іншим користувачам, але для цього була створена база, що дозволить швидко це додати в перспективі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами»; «Комп'ютерно-інтегровані технологічні процеси і виробництва»; «Комп'ютеризовані та робототехнічні системи» / упоряд.: І. Ш. Невлюдов Р. В. Артюх В. В. Безкоровайний Н. П. Демська В. В. Євсєєв О. І. Филипенко О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 21–27 с.

2. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2016.

3. Кравченко С. В. Аналіз автоматизованих ссистем керування технологічними процесами сучасного підприємства [Електронний ресурс] / С. В. Кравченко, А. І. Бронніков // Автоматизація та приладобудування : зб. студ. наук. ст., 1 вип., ел. вид. – Харків : ХНУРЕ, 2023 р. – С. 39 – Режим доступу: https://drive.google.com/file/d/1ZEp4lklhEkWoSUAZ5Pa6od2d12KQfdxe/view. – 11.10.2023.

4. Industry 5.0: the new revolution. NexusIntegra [Електронний ресурс]. – Режим доступу: https://nexusintegra.io/industry-5-0-the-new-revolution. – 11.10.2023.

5. Benjamin Mayo Apple releases 2021 Environmental Progress Report, focus on 2030 carbon neutral target [Електронний ресурс]. – Режим доступу: https://9to5mac.com/2021/04/16/apple-releases-2021-environmental-progress-report-focus-on-2030-carbon-neutral-target. – 20.09.2023.

6. Зубкова Аліна, Майгурова Дар'я, Місюня Руслан Ключові відмінності Індустрії 4.0 та 5.0 [Електронний ресурс] // Управління проєктами цифрової

трансформації міжнародних підприємств / НТУ ХПІ. – С. 3-4. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwin3uy3vpaCAxW7ExAIHeSXCUoQFnoECAwQAQ&url=https%3A%2F%2Fmdes.khmnu.edu.ua%2Findex.php%2Fmdes%2Farticle%2Fdownload%2F170%2F156&usg=AOvVaw1XyiqU4PeaxFD_fBSTkDEq&opi=89978449. – 11.10.2023.

7. Ящук Ірина Індустрія 5.0: Зміна парадигми в промисловому секторі – людина в центрі уваги [Електронний ресурс]. – Режим доступу: https://zn.ua/ukr/TECHNOLOGIES/industrija-50-zmina-paradihmi-v-promislovomu-sektori-ljudina-v-tsentri-uvahi.html. – 11.10.2023.

8. Ван Чунжі, Яцишин С. П., Лиса О. В., Мідик А-В. В. Кіберфізичні системи та їх програмне забезпечення [Електронний ресурс] / Національний університет «Львівська політехніка»; Львівський аграрний університет. – С. 1. – Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2018/sep/14522/0064.pdf. – 11.10.2023.

9. Internet of Things, IoT [Електронний ресурс]. – Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot. – 11.10.2023.

10. Шевченко О. О., Юсупов В. Т., Юрченко О. Д. Мобільні роботи: можливості, перспективи, проблеми [Електронний ресурс]. – Харків: ХНУРЕ. – Режим доступу: https://openarchive.nure.ua/server/api/core/bitstreams/9fdcebe1-3ef5-4f9a-bb75-fbdfe77e4936/content. – 13.10.2023.

11. RB-KAIROS+ Mobile manipulator [Електронний ресурс] / Robotnik. – Режим доступу: https://robotnik.eu/products/mobile-manipulators/rb-kairos. – 13.10.2023.

12. MiR100 [Електронний ресурс] / MiR. – Режим доступу: https://www.mobile-industrial-robots.com/solutions/robots/mir100. – 13.10.2023.

13. І. Г. Цмоць, А. Є. Батюк, А. В. Яворський, Т. В. Теслюк Система моніторингу технологічних процесів «Розумного підприємства». Національний університет «Львівська політехніка», кафедра автоматизованих

систем управління / – 2018, – С. 2-3. [Електронний ресурс]. – Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2019/jan/15441/10-17.pdf. – 24.10.2023.

14. Система управління базами даних [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Система_управління_базами_даних. – 25.10.2023.

15. База даних реального часу [Електронний ресурс]. – Режим доступу: http://edu.asu.in.ua/mod/book/tool/print/index.php?id=78. – 25.10.2023.

16. Node.js [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Node.js. – 25.10.2023.

17. React [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/React. – 25.10.2023.

18. Hiren Dhaduk 15 Best JavaScript frameworks to use in 2023 [Електронний ресурс]. – Режим доступу: https://www.simform.com/blog/javascript-frameworks. – 25.10.2023.

19. Кравченко С. В. Аналіз сучасного фреймворка ASP .NET Core для web-додатків [Електронний ресурс] / С. В. Кравченко, С. В. Сотник // Автоматизація та приладобудування : зб. студ. наук. ст., 1 вип., ел. вид. – Харків : ХНУРЕ, 2023 р. – С. 151 – Режим доступу: https://drive.google.com/file/d/1ZEp4lklhEkWoSUAZ5Pa6od2d12KQfdxe/view. – 25.10.2023.

20. SignalR [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/SignalR. – 25.10.2023.

21. MySQL [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/MySQL. – 25.10.2023.

22. Redis [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Redis. – 25.10.2023.

23. Що таке Kubernetes? [Електронний ресурс]. – Режим доступу: https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes. – 25.10.2023.

24. Бутрил Л., Маєвський О. В. Реляційна модель бази даних. [Електронний ресурс]. / Тернопільський національний технічний університет імені Івана Пулюя. – Режим доступу: https://core.ac.uk/download/60829388.pdf. – 25.10.2023.

25. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень. – Введ. 1999-12-01.

26. ДБН В.2.5-28:2018. Природне та штучне освітлення. Державні будівельні норми України. – Введ. 2019-03-01.