Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра спеціалізованих комп'ютерних систем

Цюпа Іван Іванович

КОМП'ЮТЕРНО-ІНТЕГРОВАНА СИСТЕМА ОБЛІКУ РОБОЧИХ ГОДИН ПРАЦІВНИКА НА ПІДПРИЄМСТВІ / A COMPUTER-INTEGRATED SYSTEM FOR RECORDING THE WORKING HOURS OF AN EMPLOYEE AT THE ENTERPRISE

спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
магістерська програма – Автоматизація та комп'ютерно-інтегровані технології

Магістерська робота

Виконав студент групи АКІТм-21
І.І. Цюпа
_____
Науковий керівник:
к.т.н.,доц. І.Б. Албанський
_____

Магістерську роботу допущено до захисту:
"____"_____20___ р.

Завідувач кафедри
_____ А.І. Сегін

Тернопіль 2022

Факультет комп’ютерних інформаційних технологій
Кафедра спеціалізованих комп’ютерних систем
Освітній ступінь "магістр"
спеціальність:151 – Автоматизація та комп’ютерно-інтегровані технології
освітньо-професійна програма – Автоматизація та комп’ютерно-інтегровані технології

ЗАТВЕРДЖУЮ
Завідувач кафедри СКС
_____ А.І. Сегін
“____”_____20__ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА
Цюпа Івана Івановича
(прізвище, ім’я по-батькові)

1. Тема кваліфікаційної роботи
Комп’ютерно-інтегрована система обліку робочих годин працівника на підприємстві / A computer-integrated system for recording the working hours of an employee at the enterprise

Керівник роботи: к.т.н., доцент І.Б. Албанський

Затверджені наказом по університету від 31 грудня 2021 р. № 606

2. Строк подання студентом закінченої кваліфікаційної роботи
16 листопада 2022р.

3. Вихідні дані до кваліфікаційної роботи:
1. Види автоматизованих систем контролю доступу на підприємство.
2. Конструктивні особливості систем автоматичного обліку.
3. Засоби контролю робочих годин працівників та їх критичне значення.
4. Вимоги до покращення систем автоматизоваоного обліку.

4. Основні питання, які потрібно розробити
1. Дослідження автоматизованих систем автоматизованого обліку робочих годин.
2. Обґрунтування вибору елементної бази та засобів автоматизації при розробці систем автоматизованого обліку.
3. Розробка інформаційної системи для роботи в мережах збору і обробки інформації.

5. Перелік графічного матеріалу у роботі
_____
_
_____
_

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26 жовтня 2020р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	АНАЛІЗ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ ЯК ОБ'ЄКТА КЕРУВАННЯ	12.2021р. – 02.2022р.	
2	ОБҐРУНТУВАННЯ ВИБОРУ ЕЛЕМЕНТНОЇ БАЗИ ДЛЯ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ	03.2022р. – 06.2022р.	
3	ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СЕРВЕРА СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБЛІКУ РОБОЧИХ ГОДИН	07.2022р. – 11.2022р.	

Студент _____ І.І. Цюпа
(підпис)

Керівник роботи _____ к.т.н.,доцент І.Б. Албанський
(підпис)

РЕФЕРАТ

Робота виконана на 76 сторінках та містить 39 рисунків, 1 таблиць, 1 додаток, 36 джерела за переліком посилань.

Мета кваліфікаційної роботи: є розробка комп'ютерно-інтегрованої системи обліку робочих годин працівників на підприємстві.

Результати роботи: система системи обліку робочих годин працівників на підприємстві, яка складається з двох частин терміналу і серверу. Дана система забезпечує надійну, безперебійну та якісну роботу в процесі обліку, також оперативне реагування та само балансування на нестандартних ситуаціях роботи. Це в свою чергу дозволяє мінімізувати витрати фізичної праці, що значно підвищує надійність роботи системи. Також ця система дозволяє мінімілізувати навантаження на мережу, що значно підвищує надійність роботи подібної систем та мережі в цілому.

Рекомендації по використанню результатів роботи: розроблена автоматизована система обліку на основі клієнт серверної архітектури, застосування якої дозволяє покращити економічність, надійність та гнучкість роботи подібних систем.

Ключові слова: ТЕХНОЛОГІЧНИЙ ПРОЦЕС, АВТОМАТИЗОВАНА СИСТЕМА, ТЕХНІЧНІ ЗАСОБИ АВТОМАТИЗАЦІЇ, СИСТЕМА ОБЛІКУ РОБОЧИХ ГОДИН.

ABSTRACT

The work is completed on 76 pages and contains 39 figures, 1 table, 1 additions, 36 sources after the list of references.

The purpose of the qualification work is to develop a computer-integrated system for recording the working hours of employees at the enterprise.

Research results. The system of accounting for the working hours of employees at the enterprise, which consists of two parts: a terminal and a server. This system ensures reliable, uninterrupted and high-quality work in the accounting process, as well as prompt response and self-balancing in non-standard work situations. This, in turn, allows you to minimize the costs of physical labor, which significantly increases the reliability of the system. Also, this system allows you to minimize the load on the network, which significantly increases the reliability of the operation of such systems and the network as a whole.

Recommendations for the use of work results: an automated accounting system based on a client-server architecture has been developed, the use of which allows improving the economy, reliability and flexibility of the operation of similar systems.

Keywords: TECHNOLOGICAL PROCESS, AUTOMATED SYSTEM, TECHNICAL MEANS OF AUTOMATION, WORKING HOUR ACCOUNTING SYSTEM.

ЗМІСТ

ВСТУП

Актуальність теми. Актуальність магістерської роботи полягає в необхідності дослідження питань реєстрації та обліку доступу до різних виробничих приміщень. Переваги таких систем незліченні. Вони не лише точно відраховують час для роботодавців, але й зберігають дані для команди відділу кадрів, які можна використовувати під час нарахування заробітної плати. Для працівників, якщо на робочому місці реалізована подібна система це означає, що вони матимуть чітке уявлення про свій робочий день, час початку, час закінчення та перерви. Якщо роботодавець вимагає від працівника понаднормових годин роботи він повинен отримати справедливу компенсацію. Ці чіткі розмежування надають працівникам прозорий графік та оплату праці, який включає дані про всі відпрацьовані додаткові години. Оптимізована система відстеження часу дозволяє співробітникам легко та ефективно записувати години перерв. Якщо заклад покладається на ручний облік часу перерв, тоді може виникнути плутанина. Але система документує вхід і вихід працівника, а також фіксує тривалість їхніх перерв на обід тощо. Це зменшує непотрібну плутанину та дозволяє працівникам ефективно керувати власним розкладом. Крім того, це дає роботодавцям зручне рішення для моніторингу часу співробітників без будь-яких сторонніх проблем.

Час є одним із обмежених ресурсів для будь-якої установи. Сучасні системи інтегруються з мобільними пристроями. Вони збирають та відстежує дані про введення та виведення даних і повідомляє керівництво про розбіжності. Більшість із цих програм мають вбудовані функції для виявлення лазівок у відвідуваності співробітників. Це допомагає керівництву впровадити нову політику або змінити її відповідно до вимог компанії.

У даній роботі буде розглянуто створення комп'ютерно інтегрованої системи обліку робочих годин працівника на підприємстві. Після проведення аналізу об’єкта управління було прийнято рішення про розбивку об’єкта управління на дві підсистеми. Кожен із контурів виконує своє завдання,

спрямоване на виконання загальної задачі системи збору часу входу-виходу працівника із робочого місця.

Мета і завдання дослідження. Метою створення системи реєстрації та обліку, яка дозволить реєструвати і аналізувати час входу і виходу працівника із підприємства, реєструвати понаднормові, недопрацьовані та вихідні години. Що в свою чергу дозволить правильно нараховувати заробітну плату. Також система повинна поділятися на дві частини: термінал і сервер, що дозволить знизити нагрузку на менш потужний термінал і зробить систему в загальному більш гнучкою.

Для реалізації поставленої мети були вирішені наступні питання: проведено дослідження та аналіз існуючого підходу до управління системою обліку.

Об’єктом дослідження. Робота комп'ютерно-інтегрованої системи обліку на підприємстві, принципи регулювання обліку та основні компоненти системи.

Предметом дослідження є засоби та компоненти автоматизованої системи контролю при необхідних параметрах ефективної роботи.

Наукова новизна одержаних результатів: реалізований програмний код сервера, що зменшує навантаження на апаратну частину. Що в свою чергу підвищує швидко дію всієї системи.

Практичне значення одержаних результатів в тому, що запропоновані практичні рекомендації по проектуванню систем автоматизованого управління дозволяють підвищити їх надійність та ефективність.

Апробація. За результатами досліджень підготовлено та опубліковано 2 тез доповідей на наукових конференціях (додаток А).

1. АНАЛІЗ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ ЯК ОБ'ЄКТА КЕРУВАННЯ

1.1 Загальна характеристика системи обліку

Метою системи обліку робочих годин є електронний контроль і реєстрація відвідуваності для точності нарахування заробітної плати. За допомогою системи обліку робочого часу співробітників компанія створює докладні звіти з інформацією про години роботи співробітників, відгули, понаднормову роботу, відсутність тощо. Це допомагає автоматизувати розрахунок зарплати та забезпечити дотримання організацією трудового законодавства. Система реєструє робочі години та понаднормовий час, щоб ефективно контролювати діяльність співробітників. На даний момент такі системи можна поділити (рисунок 1.1):

- із використанням перфокарти;
- із використання біометричної ідентифікації;
- цифрові системи;
- веб системи;
- мобільні системи.

Системи із використанням перфокарти вважаються застарілими. Працівники забезпечуються паперовою картою обліку робочого часу, яка вставляється в машину для обліку робочого часу після прибуття та відходу з робочого місця [2]. Машина записує або перфорує картку часу, щоб вказати час початку та закінчення. Ця система неефективна для робочих місць, де працюють віддалені працівники.

Системи з біометричною ідентифікацією використовують апаратний пристрій, який зазвичай встановлюється біля входу в офіс, використовують біометричні дані, такі як відбитки пальців, долонь, розпізнавання обличчя тощо, щоб підтвердити особу людини для реєстрації входу та виходу.

Цифрові системи використовують картку з магнітною смугою, щоб зареєструвати дату та час виходу та входу. Цей вид системи може

використовувати сенсорну панель, як альтернативу карти. В свою чергу для реєстрації даних вводяться унікальні ідентифікатори, створені роботодавцями.

Рисунок 1.1 – Системи обліку робочих годин

Веб системи дозволяють працівникам відзначати свою присутність через веб-браузер на комп'ютері за допомогою унікальних облікових даних для обліку часу початку/закінчення робочого дня або перерви.

Мобільні системи стали популярними останнім часом. У світі який змінила пандемія коронавірусу, де дистанційна робота стала нормою, перевагу надають мобільним системам. Це дозволяє співробітникам відзначати присутність за допомогою мобільного пристрою, наприклад мобільного телефону або планшета. Кращі версії інтегровані з системою GPS і дозволяють відстежувати місцезнаходження [3]. Мобільні системи є незамінним рішенням для таких галузей, як ресторани та роздрібна торгівля, де потрібно мати багато партнерів-доставки. Це полегшує відстеження переміщень співробітників.

Переваги таких систем незліченні. Вони не лише точно відраховують час для роботодавців, але й зберігають дані для команди відділу кадрів, які можна використовувати під час нарахування заробітної плати (рисунок 1.2). Для працівників, якщо на робочому місці реалізована подібна система це означає, що вони матимуть чітке уявлення про свій робочий день, час початку, час закінчення

та перерви. Якщо роботодавець вимагає від працівника понаднормових годин роботи він повинен отримати справедливу компенсацію.

Рисунок 1.2 – Система обліку робочих годин

Ці чіткі розмежування надають працівникам прозорий графік та оплату праці, який включає дані про всі відпрацьовані додаткові години. Оптимізована система відстеження часу дозволяє співробітникам легко та ефективно записувати години перерви. Якщо заклад покладається на ручний облік часу перерв, в такому випадку може виникнути плутанина. Але система документує вхід і вихід працівника, а також фіксує тривалість їхніх перерв на обід тощо. Це зменшує непотрібну плутанину та дозволяє працівникам ефективно керувати власним розкладом. Крім того, це дає роботодавцям зручне рішення для моніторингу часу співробітників без будь-яких проблем. Час є одним із обмежених ресурсів для будь-якої установи. Сучасні системи інтегруються з мобільними пристроями. Вони збирають та відстежують дані про введення та виведення даних і повідомляє керівництво про розбіжності. Більшість із цих програм мають вбудовані функції для виявлення лазівок у відвідуваності співробітників. Це допомагає керівництву впровадити нову політику або змінити її відповідно до вимог компанії.

У таких сферах, як роздрібна торгівля чи ресторанний бізнес, де працівники повинні бути присутніми в певні години, точне відстеження їхнього часу допомагає закладу. Система визначає неробочий час і гарантує, що роботодавці не

виплачують працівникам години, які вони не відпрацювали. За певний період часу це може заощадити значні гроші для закладу. Коли працівники використовують систему для відстеження часу, пристрій створює автоматичний запис, що полегшує відстеження регуляторних і понаднормових годин. Це вигідно як для працівників, так і для роботодавців. Працівники можуть справедливо вимагати оплатити відпрацьовані ними понаднормові годин. Роботодавці скорочують несанкціоновану понаднормову роботу та забезпечують дотримання закону. Однією з найважливіших переваг системи це завдяки постійному відстеженню часу вона дозволяє роботодавцю справедливо вирішувати, яким працівникам потрібно платити більше. Зрештою, подібні системи — це не лише облік, але й відстеження ефективності співробітників і розуміння того, хто добре оптимізує свій час [4]. Це допомагає збільшити продуктивність і прибуток. На основі отриманих даних керівники можуть вирішити, чи цінувати працьовитих співробітників.

1.2 Обґрунтування вибору мови програмування

Згідно з багатьма дослідженнями, Java пропонує різні переваги, тому є чудовим вибором для IoT. Однією з таких переваг є те, що на відміну від інших мов, Java дозволяє написати код один раз, а потім використовувати його та запускати всюди. Фонд Eclipse з 2015 року проводить опитування розробників IoT у всьому світі, щоб зрозуміти екосистему, тенденції та проблеми в цій галузі. Згідно з опитуванням 2018 року, IoT з використанням Java очолила чарти з колосальними 66,5% (рисунок 1.3), що робить її найпопулярнішою мовою програмування для розробки IoT. Це революційна технологія, яка захопила світ. Після Інтернету IoT є найбільш затребуваним винаходом. Ринок Інтернету речей зростає не по днях, а по годинах, оскільки всюди з’являється багато нових комп’ютерів, пристроїв і технологій [5]. За даними IDC, до 2024 року загальна кількість пристроїв IoT у всьому світі перевищить 35 мільярдів. Все більше компаній інвестують значні кошти в ці ініціативи. Кожен ринок або галузь має

широкі можливості для використання IoT з Java завдяки її гнучкості. На ній легко писати та використовувати, компілювати, вивчати та налагоджувати. Ця мова програмування призначена для зв'язку речей. Взаємодія або скоординована робота кількох пристроїв є характерною рисою Інтернету речей, і Java робить ставку саме на це. Java відносно менш складна та проста у використанні порівняно з іншими мовами, що робить її кращим вибором для розробників.

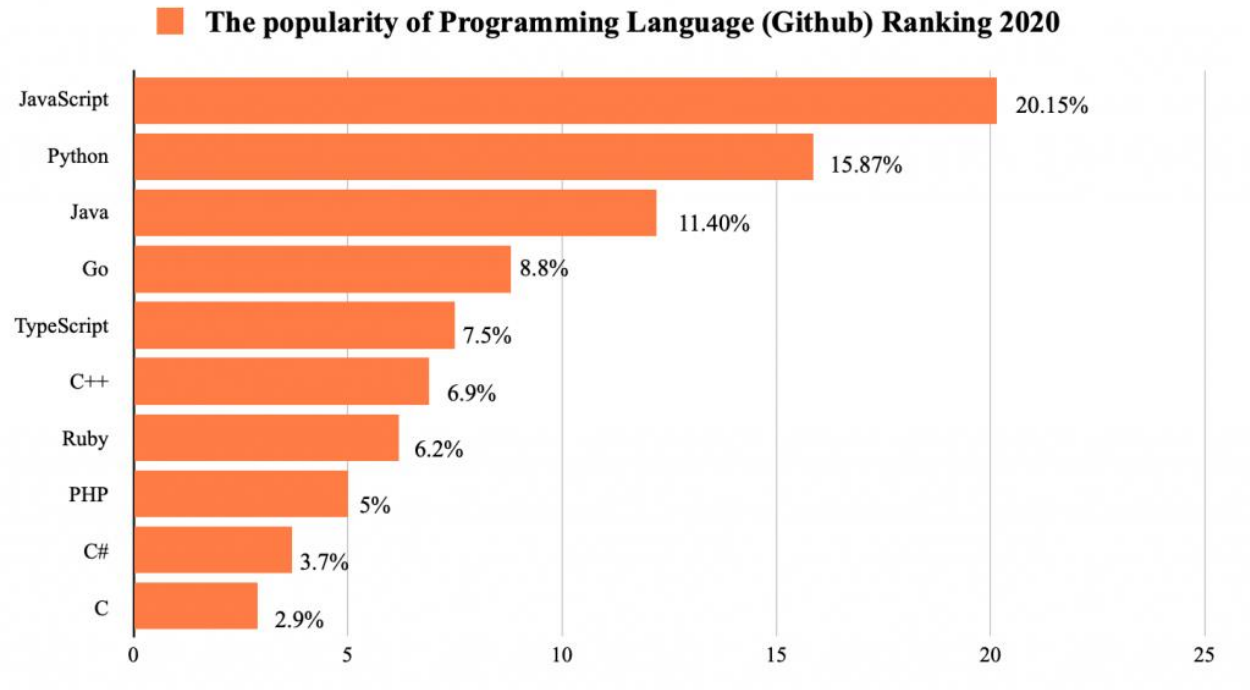

Рисунок 1.3 – Популярність мов програмування

Ця мова усуває проблеми з безпекою даних, що робить її придатною для Інтернету. Існує багато пояснень вибору Java як кращого рішення для розробників. Розробляючи вбудовану програму, вам потрібно враховувати ОС реального часу, процесор і різні протоколи для підключення пристроїв. Той факт, що така незалежна від платформи мова, як Java узагальнює всі ці фактори, допомагає розробникам написати код один раз і запустити його будь-де. Розроблений додаток Java IoT може працювати на різних апаратних пристроях і платформах без зміни коду [6]. Це робить розробку IoT з Java більш ефективною і менш громіздкою. Гнучкість і доступність Java роблять її найкращою платформою для Інтернету речей. Програми Java IoT легко перенести на нову

платформу, що зменшує ймовірність помилок і проблем. Java була розроблена для роботи в середовищах з обмеженими ресурсами. Таким чином, розробка IoT з Java використовуватиме мінімум ресурсів і забезпечуватиме максимальну ефективність.

Програмування на Java пропонує своїм користувачам переваги понад 4000 бібліотек, що охоплюють усі вимоги до програмування IoT, починаючи від паралелізму до роботи в мережі. API вимагає мінімального переписування або взагалі не потребує, що призводить до швидкого виконання коду та розробки додатків. У Java використовуються неявні покажчики, якими неможливо маніпулювати через надійний код програми. Буде менше ймовірностей проблем, таких як несанкціонований доступу до пам'яті або переповнення буфера, що підкреслює важливість Java [7]. Java легше вивчити, ніж більшість інших мов програмування. Вона пропонує численні функції, такі як безпека та масштабованість. Особливістю є те, що ви можете використовувати існуючі API, а не створювати нові коди з нуля під час створення вбудованих програм. Java API є обширним. Стандартний JDK постачається з понад 200 вбудованими пакетами, що містять Java API, які дозволяють виконувати будь-що, починаючи від аналізу XML і закінчуючи перекладом між часовими поясами. Коли розробники додають API Jakarta EE, вони отримують ще більшу бібліотеку API, яка дозволяє розробляти складні програми середнього рівня та хмарні мікросервіси.

Однією з головних причин популярності Java є те, що вона дуже гнучка та надійна, і до неї можна отримати доступ з будь-якого місця. Інтернет речей — це не єдина технологія; це поєднання технологій, включаючи великі дані, хмарні обчислення, співпрацю M2M, датчики, обчислювальні та апаратні пристрої, що робить Java найкращим вибором для об'єднання всіх пристроїв. Ще один важливий фактор, який робить програмування додатків Java настільки корисним у програмах IoT, що воно містить величезну бібліотеку інтерфейсів прикладних процесів. Крім цього Java програма стикається з меншою кількістю проблем під час оновлення програми. Можна сказати, що Java народилася у вбудованому середовищі. У той час доступна обчислювальна потужність була іншою, ніж

сьогодні. Java є однією з найбільш використовуваних мов у світі та найуспішнішою мовою в сфері IT. Своїм неперевершеним успіхом ця мова програмування завдячує принципу “напиши один раз, запусти де завгодно”, важливій функції, яка усуває залежності від платформи під час виконання програми.

Інтернет речей — це ідея підключення пристроїв до Інтернету та їхню взаємодію певним чином. Інтернет речей — це величезна мережа пов’язаних точок, обчислювальних пристроїв, об’єктів, машин тощо. Усі ці речі мають власні унікальні ідентифікатори і здатні збирати та обмінюватися інформацією в різних мережах без необхідності взаємодії людини з комп’ютером або людини з людиною. Інтернет речей набув величезної популярності завдяки зростанню аналітики даних і штучного інтелекту та машинного навчання. Від фітнес-пристроїв до кухонних приладів, таких як мікрохвильові печі, від безпілотних автомобілів до самостійних роботів IoT є всюди.

Як було сказано вище, пристрої IoT обмінюються та збирають інформацію через різні точки дотику [8]. Ці точки дотику створюються за допомогою вбудованої програми, яка використовується для кращого підключення. Технологія Інтернету речей зараз далеко не модне слово. Це стало реальністю і компанії веб-розробників Java роблять великі кроки у використанні технології, щоб запропонувати клієнтам винятковий досвід. Завдяки IoT можливо трансформувати досвід клієнтів, що призведе до трансформування цінностей у додатки, що було неможливо уявити деякий час тому. Уряди використовують технологію IoT, щоб будувати розумніші міста з численними додатками для мешканців. Для використання технології IoT у всіх цих програмах потрібні надійні програми, які можна розробляти відповідно стандартам даних у різних країнах та з дотриманням стандартів безпеки. Таким чином, можна з упевненістю сказати, що Java — це мова програмування, яка використовується програмістами для створення програм IoT.

Oracle вважає перевагою номер один, яка привела до цих результатів, надійність коду програми, розробленого на Java. C - це мова, яка використовує

явні покажчики на еталонну пам'ять. Але Java робить усі посилання на об’єкти неявними покажчиками, які не можна змінити за допомогою коду програми. Ця функція в Java виключає будь-яку можливість порушення доступу до пам’яті, яке зазвичай призводить до раптової зупинки програм. Ще одна особливість мови програмування Java, яка спадає на думку, це виконання коду на декількох платформах. Що стосується C, розробникам знадобиться багато часу, щоб перенести програми на C на іншу платформу. Тоді як Java може працювати будь-де після того, як код був написаний один раз. Функція, яка стає в нагоді, коли ми говоримо про IoT, — це портативність. Стів Джобс вважав, що Java — це мертва мова, і нею ніхто не користується. Але цифри говорять нам, що Java дуже жива і в тренді. Java також має можливість підключення на рівні програми. Це досягається за допомогою набору API, які є стандартними та вільно доступними в проектах з відкритим кодом. Крім того, додайте до цього той факт, що Java була народжена для вбудованих програм.

Новий випуск Java з довгостроковою підтримкою вже не за горами. Оскільки Java перетинає 25-річний рубіж, давайте відступимо назад і розглянемо деякі причини, чому Java залишається найкращою мовою програмування для сучасної розробки програмного забезпечення [9]. Кожна мова програмування потребує певного навчання, але Java має багато спільного з C, C++ і JavaScript . Кожен хто має досвід роботи з будь-якою з цих мов швидко зрозуміє синтаксис Java. Java також має дуже жорсткий і передбачуваний набір правил, які керують структурою коду. Це різко контрастує з іншими, нетиповими мовами сценаріїв, де здається, що все підійде. Коли ви намагаєтесь освоїти нову мову програмування, чіткий набір правил, які постійно дотримуються, полегшує навчання. Крім того, коли щось не має сенсу, програмісти, які новачки у Java, можуть знайти потужну мережу підтримки відео на YouTube, веб-сайти, такі як StackOverflow, і онлайн-форуми, такі як CodeRanch , щоб знайти відповіді на безліч запитань. За понад 25 років еволюції Java має багато еволюційних удосконалень якими можна пишатися. Від куленепробивної модульної системи, яка була надана як частина проекту Jigsaw, до нещодавно доданої можливості функціонального

програмування Java з лямбда-функціями, Java продовжує впроваджувати великі зміни яких вимагає спільнота. Поступові доповнення у версіях без LTS, такі як додавання нового типу даних Record і збирачів сміття для покращення керування пам'яттю демонструють, що JDK також постійно розвивається. Але у світі розробки корпоративного програмного забезпечення зворотна сумісність так само важлива, як і додавання нових функцій. Java завжди робила це головним пріоритетом для розпорядників мови. Дуже рідко комплексне оновлення чи додавання функцій створює проблеми з кодом, написаним на основі старіших випусків.

Для збереження даних існує проект JBoss Hibernate. Для розробки хмарних мікросервісів на Java є повний набір Spring Boot API. І звісно, існує безліч проектів API Apache з відкритим кодом, які стосуються широкого спектру випадків використання програмного забезпечення, від агрегування повідомлень про помилки за допомогою log4j до вирішення складних проблем за допомогою HashMaps і безвідмовних ітераторів через Apache Commons Collections API. Ландшафт розробки додатків наповнений інструментами розробки програмного забезпечення, написаними мовою Java які розроблені для спрощення та оптимізації розробки, розгортання та навіть виведення з експлуатації додатків Java. Кілька прикладів інструментів, написаних на Java, включають:

- gradle: неймовірно потужний інструмент збірки з відкритим кодом;
- maven: інструмент з відкритим вихідним кодом, призначений для вирішення проблем керування залежностями;
- jenkins: інструмент безперервної інтеграції та доставки на основі Java.

Це далеко не вичерпний список екосистеми інструментів Java. Інші приклади інструментів і технологій написаних на Java включають сервери додатків, як-от Tomcat, до популярного Kubernetes стека Java під назвою Quarkus Red Hat.

Android є найпопулярнішою у світі ОС для мобільних телефонів, а Java є де-факто мовою програмування для розробки додатків Android. Хоча версія Java для Android не зовсім та сама, що можна знайти в JDK, Google дійсно скопіював

понад 11 500 рядків коду з Java Standard Edition, коли створював свій клон Java. У результаті розробники можуть очікувати, що версія Java, яку вони бачать на Android, буде досить близькою до оригіналу. Якщо ви вмієте писати код Java для настільних або серверних програм, ви дуже швидко можете почати розробляти для Android. Усі низькорівневі відмінності між JVM і Android Runtime будуть приємно абстраговані від вас після короткого навчання. Коли розробники вивчать Java у їхньому розпорядженні буде вся екосистема Android. Java розвивається повільно, але вона розвивається.

Програмування на Java є одночасно зручним і гнучким, що робить її очевидною мовою програмування для розробників веб-додатків і експертів з управління програмами. Під гнучкістю в цьому випадку ми маємо на увазі, що програма, розроблена в системі кодування, може працювати послідовно в будь-якій операційній системі, незалежно від ОС, в якій вона була спочатку розроблена. Незалежно від того, чи потрібна вам мова для чисельних обчислень, мобільних обчислень чи настільних комп'ютерів Java допоможе вам. ЇЇ код легко зрозуміти та легко усунути неполадки. Крім безперебійної роботи на комп'ютерах Mac, Linux або навіть Windows, JRE також сумісний із мобільними телефонами. Оскільки ви можете запускати Java як на комп'ютерах, так і на мобільних пристроях, можна сказати, що діалект мови є універсальним. Більше того, він ідеально працює на таких пристроях, як Raspberry Pi. З іншого боку, ви можете запускати Java як у великому, так і в малому масштабі, а це означає, що її коди надійні та стабільні. Як було зазначено, у Java немає жодних обмежень ви навіть можете розробити програмне забезпечення для перекладу, використовуючи цю мову. Однак для отримання найкращих результатів завжди доцільно тісно співпрацювати з професійним постачальником послуг перекладу.

Spring — це найпопулярніший фреймворк розробки корпоративних додатків з використанням мови Java. Мільйони розробників у всьому світі використовують Spring Framework для створення високопродуктивного коду, який легко тестувати та використовувати повторно. Spring framework — це платформа Java з відкритим кодом. Спочатку він був написаний Родом Джонсоном і вперше був випущений

під ліцензією Apache 2.0 у червні 2003 року. Базова версія Spring Framework становить близько 2 МБ. Основні функції Spring Framework можна використовувати для розробки будь-якої програми Java, проте існують розширення для створення веб-програм на основі платформи Java EE. Spring Framework має на меті зробити розробку J2EE простішою у використанні та популяризує ефективні практики програмування, використовуючи модель програмування на основі POJO.

Нижче наведено список кількох великих переваг використання Spring Framework. Spring дозволяє розробникам розробляти програми корпоративного класу за допомогою POJO. Перевага використання лише POJO полягає в тому, що вам не потрібно EJB, наприклад сервер додатків, оскільки ви можете використовувати лише надійний контейнер сервлетів, наприклад Tomcat або інший комерційний продукт. Фреймворк організовано за модульним принципом. Незважаючи на те, що кількість пакунків і класів є значною, є можливість використовувати лише про ті, які вам потрібні і ігнорувати інші. Spring не винаходить велосипед заново, натомість він використовує деякі з існуючих технологій, як-от кілька фреймворків ORM, фреймворків журналювання, таймерів JEE, Quartz і JDK та інших технологій [10]. Тестування програми написаної за допомогою Spring просте, оскільки код що залежить від середовища, переміщується в цю структуру. Крім того, за допомогою JavaBeanstyle POJO стає легше використовувати ін’єкцію залежностей для ін’єкції тестових даних.

Веб-фреймворк Spring — це веб-фреймворк MVC, який надає чудову альтернативу веб-фреймворкам, таким як Struts або іншим менш популярним веб-фреймворкам. Spring надає зручний API для перекладу специфічних для технології винятків (наприклад JDBC, Hibernate або JDO). Контейнери IoC є досить легкими, особливо в порівнянні, наприклад з контейнерами EJB. Це корисно для розробки та розгортання програм на комп’ютерах з обмеженою пам’яттю та ресурсами ЦП. Spring забезпечує узгоджений інтерфейс керування транзакціями, який можна масштабувати до локальної транзакції (наприклад за

допомогою однієї бази даних) і масштабувати до глобальних транзакцій (наприклад за допомогою JTA).

Технологія з якою Spring найбільше ототожнюють — це інверсія керування залежностями (DI). Інверсія контролю (IoC) є загальною концепцією і її можна виразити різними способами. Ін'єкція залежностей – лише один конкретний приклад інверсії контролю. Під час написання складної програми Java, класи програми повинні бути якомога більш незалежними від інших класів Java, щоб збільшити можливість повторного використання цих класів і тестування їх незалежно від інших класів під час модульного тестування. Ін'єкція залежностей допомагає об'єднати ці класи разом і в той же час зберегти їх незалежність. Впровадження залежностей може відбуватися шляхом передачі параметрів конструктору або шляхом пост-конструкції за допомогою методів установщика. Ін'єкція залежностей є основою Spring Framework.

Одним із ключових компонентів Spring є структура аспектно-орієнтованого програмування (AOP). Функції які охоплюють кілька точок програми, називаються наскрізними проблемами, і ці наскрізні проблеми концептуально відокремлені від бізнес-логіки програми. Існують різні загальні приклади аспектів, включаючи журналювання, декларативні транзакції, безпеку, кешування тощо. Ключовою одиницею модульності в ООП є клас, тоді як в АОП одиницею модульності є аспект. DI допомагає вам відокремити об'єкти програми один від одного, тоді як AOP допомагає відокремити наскрізні проблеми від об'єктів, на які вони впливають. Модуль AOP Spring Framework забезпечує реалізацію аспектно-орієнтованого програмування, що дозволяє вам визначати перехоплювачі методів і точки розрізів, щоб чітко відокремити код, який реалізує функціональність, яку слід розділити.

Spring потенційно може бути єдиною платформою для всіх корпоративних програм. Spring має модульну структуру (рисунок 1.4), що дозволяє вибирати модулі, які вам підходять, без необхідності вводити решту.

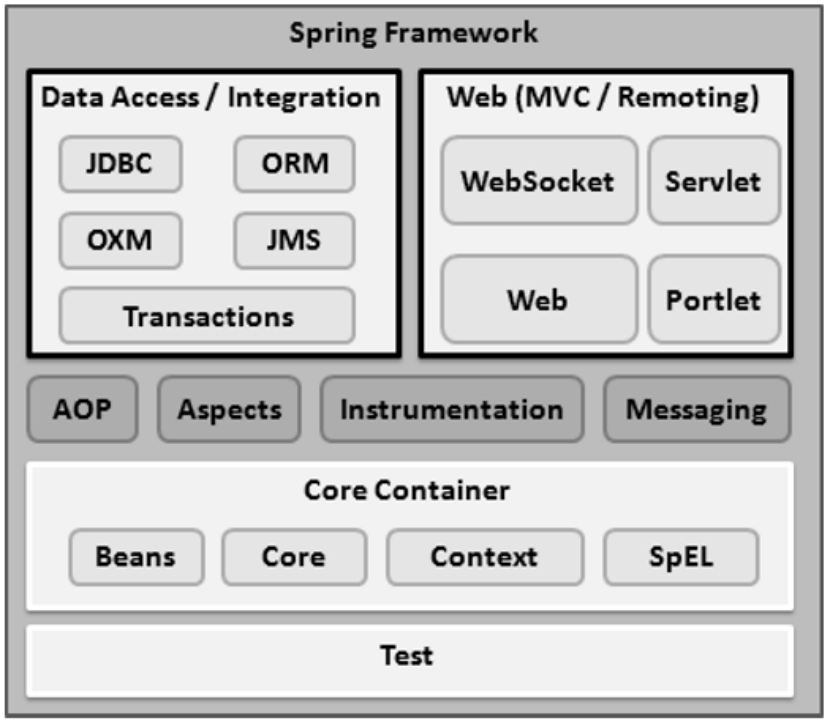

Рисунок 1.4 - Модульна структура Spring

Spring Framework надає близько 20 модулів, які можна використовувати відповідно до вимог програми. Контейнер Core складається з модулів Core, Beans, Context і Expression Language:

- модуль Core забезпечує фундаментальні частини фреймворку, включно з функціями IoC і Dependency Injection;

- модуль Bean забезпечує BeanFactory, який є складною реалізацією фабричного шаблону;

- модуль Context будується на міцній основі, що надається модулями Core і Beans, і є засобом доступу до будь-яких визначених і налаштованих об’єктів. Інтерфейс ApplicationContext є центром модуля Context;

- модуль SpEL надає потужну мову виразів для запитів і маніпулювання графом об’єктів під час виконання.

Рівень інтеграції складається з модулів JDBC, ORM, OXM, JMS і Transaction:

- модуль JDBC забезпечує рівень абстракції JDBC, який усуває потребу у виснажливому кодуванні, пов’язаному з JDBC;

- модуль ORM забезпечує рівні інтеграції для популярних API об’єктно-реляційного відображення, включаючи JPA, JDO, Hibernate та iBatis;

- модуль OXM забезпечує рівень абстракції, який підтримує реалізацію відображення Object/XML для JAXB, Castor, XMLBeans, JiBX і XStream;

- модуль JMS це служба обміну повідомленнями Java містить функції для створення та споживання повідомлень;

- модуль Transaction підтримує програмне та декларативне керування транзакціями для класів, які реалізують спеціальні інтерфейси, і для POJO класів.

Веб-рівень складається з модулів Web, Web-MVC, Web-Socket і Web-Portlet:

- web модуль надає базові функції веб-орієнтованої інтеграції, такі як функція завантаження файлів та ініціалізація контейнера IoC за допомогою слухачів сервлетів і контексту веб-орієнтованої програми;

- модуль Web-MVC містить реалізацію Model-View-Controller (MVC) Spring для веб-одатків;

- модуль Web-Socket забезпечує підтримку двостороннього зв’язку на основі WebSocket між клієнтом і сервером у веб-додатках;

- модуль Web-Portlet забезпечує реалізацію MVC для використання в середовищі портлетів і відображає функціональність модуля Web-Servlet.

Існує кілька інших важливих модулів, таких як AOP, Aspects, Instrumentation, Web і Test modules:

- модуль AOP забезпечує реалізацію аспектно-орієнтованого програмування, що дозволяє визначати перехоплювачі методів і точки розрізів для чіткого відокремлення коду, який реалізує функціональність, яку слід розділити;

- модуль Aspects забезпечує інтеграцію з AspectJ, який також є потужною структурою AOP;

- модуль Instrumentation забезпечує підтримку інструментів класу та реалізацію завантажувача класів для використання на певних серверах додатків;

- модуль обміну повідомленнями забезпечує підтримку STOMP як під протоколу WebSocket для використання в програмах. Він також підтримує модель

програмування анотацій для маршрутизації та обробки повідомлень STOMP від клієнтів WebSocket;

- модуль Test підтримує тестування компонентів Spring за допомогою фреймворків JUnit або TestNG.

Ви можете завантажити останню версію SDK із сайту Oracle Java − Java SE Downloads. Ви знайдете інструкції щодо встановлення JDK у завантажених файлах. Дотримуйтеся наведених інструкцій, щоб встановити та налаштувати параметри. В кінці потрібно встановити змінні середовища PATH і JAVA_HOME для посилання на каталог, який містить java і javac, це як правило java_install_dir/bin і java_install_dir відповідно.

Крім того, у Windows NT/2000/XP вам доведеться клацнути правою кнопкою миші на Мій комп'ютер, вибрати Властивості, Додатково, Змінні середовища. Потім вам доведеться оновити значення PATH і натиснути кнопку OK. Крім того, якщо ви використовуєте інтегроване середовище розробки (IDE), наприклад Borland JBuilder, Eclipse, IntelliJ IDEA або Sun ONE Studio, вам доведеться скомпілювати та запустити просту програму, щоб підтвердити, що IDE знає, де ви встановили Java. В іншому випадку вам доведеться виконати належне налаштування, як зазначено в документації IDE(рисунок 1.5).

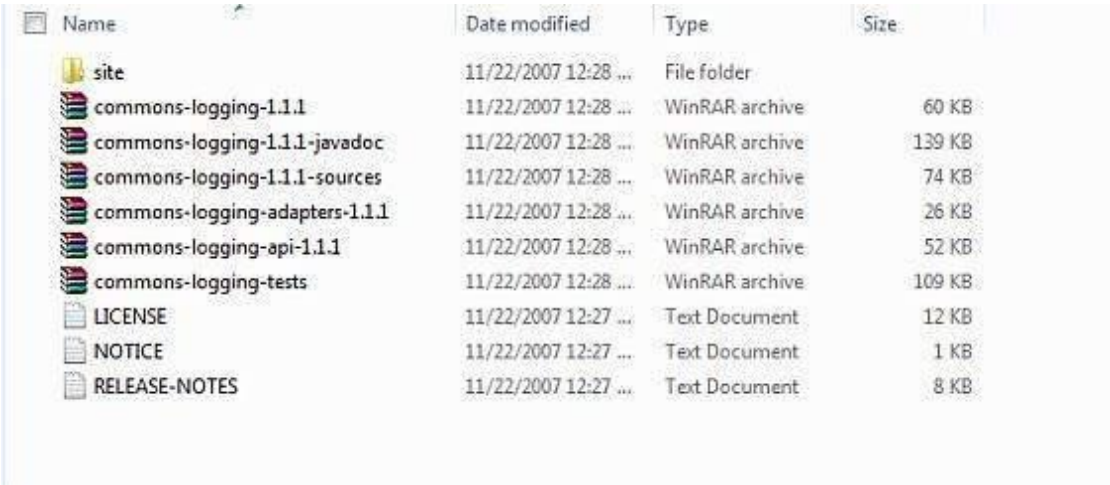

Рисунок 1.5 - Пакет логування

Ви можете завантажити останню версію Apache Commons Logging API. Після завантаження інсталяції розпакуйте двійковий дистрибутив у зручне місце.

Цей каталог міститиме наступні файли jar та інші супровідні документи тощо. Переконайтеся, що ви правильно встановили свою змінну CLASSPATH у цьому каталозі, інакше ви зіткнетеся з проблемою під час роботи програми. Щоб інсталювати Eclipse IDE, завантажте останню версію Eclipse з офіційного сайту. Після завантаження інсталяції розпакуйте двійковий дистрибутив у зручне місце і належним чином встановіть змінну PATH. Після успішного запуску, якщо все гаразд, має відображатися такий результат (рисунок 1.6).

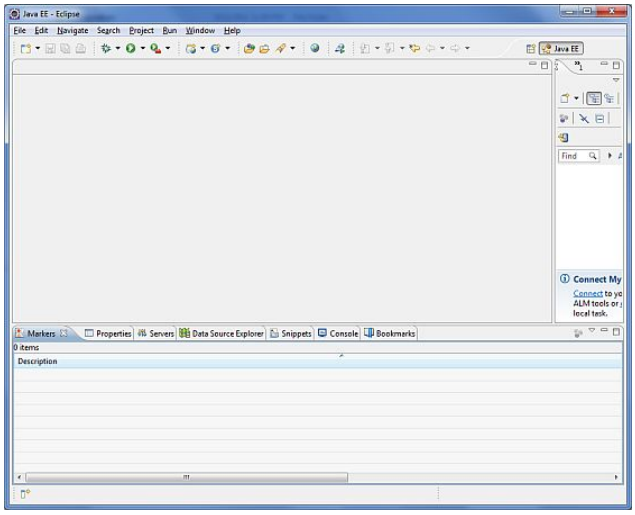

Рисунок 1.6 - Середовище розробки

Нижче обґрунтовано вибір Spring фреймворку:

- інтуїтивна функціональність;
- безпека;
- простота спілкування з базами даних;
- модульний дизайн;
- можна інтегрувати з іншими фреймворками;
- ін'єкція залежності;
- тестування.

Spring легко освоїти та реалізувати, він складається з модулів, за допомогою яких можемо писати програми Spring з інтерфейсами та абстрактними класами, як програми Java. Spring допомагає з’єднати компоненти, що робить роботу зі Spring легшою та простішою, оскільки ми можемо більше зосередитися на нашій програмі, а не на її реалізації. Він легкий, оскільки ми можемо вводити залежності

відповідно до наших потреб без необхідності включати кожну з них, таким чином позбавляючи проект від непотрібного використання пам’яті. Spring забезпечує безпеку, якщо Spring Security знаходиться на шляху до класів, і ми можемо додатково налаштувати параметри безпеки для базової автентифікації та запобігання вразливостям у нашому проекті. MVC — це шаблон і методологія розробки програмного забезпечення, що розшифровується як Model View Controller, який допомагає відокремити реалізацію та бізнес-логіку, щоб розробники могли зосередитися на своєму коді для кращої продуктивності програми. Простіше кажучи, представлення — це місце, де спочатку надходять запити, які контролер передає до відповідної моделі, які потім передаються для відображення в браузері. Це систематичне отримання та вирішення запитів робить фреймворки на основі MVC хорошими рішенням для роботи. Spring підтримує шаблон MVC для розробки проектів, які сприяють поділу проблем, що є однією з ключових особливостей потенційного програмного забезпечення.

Бази даних є важливою частиною програми, оскільки без плавного та легкого зв’язку з базою даних наша програма може стати складною. Spring забезпечує легкий і ефективний зв’язок із базами даних оскільки має функціональні можливості DAO (Data Access Object), які призначені для читання та запису даних у базу даних. Завдяки підтримці DAO у Spring пов’язані з доступом до даних технології, такі як Hibernate, JDBC і JPA, спрощують спілкування з базами даних [11]. Наприклад, Hibernate у Spring полегшує виконання операцій CRUD, усуваючи рядки коду та використовуючи лише прості функції для створення, читання, оновлення або видалення даних із бази даних. Однією з яскравих особливостей Spring є його модульність. Це не цілий великий фреймворк, який містить усе як тісно пов’язаний пакет, а фреймворк із різними файлами JAR, які можна використовувати незалежно один від одного. Він розділений на такі компоненти, як основний контейнер , доступ/інтеграція даних , веб і тест, які працюють разом але незалежно. Реалізацію також можна розділити на Spring відповідно до шаблону MVC, де представлення взаємодіє з користувачем і відповідає за отримання запитів і надсилання відповідей

користувачеві, контролер працює над наданням запитів правильному методу та моделі, що працює з базами даних і даними.

Наявність широкої мережі спеціалістів вважається одним із ключових моментів успіху в сучасному світі. Spring успішний, оскільки він має хорошу мережу і може використовуватися з іншими фреймворками, такими як Struts і Hibernate . Це робить його більш затребуваним, оскільки можна полегшити операції CRUD для баз даних, коли Hibernate інтегровано з Spring. Залежність не корисна ні для людей, ні для класів у проектах. Впровадження залежностей допомагає зменшити зв'язок і залежність між класами в проектах Spring, щоб програму можна було підтримувати та використовувати повторно. Модулі одного проекту можна ефективно використовувати в іншому проекті, як-от сторінка входу та реєстраційна сторінка, що не лише економить час, але й сприяє багаторазовому використанню коду, що є одним із важливих аспектів розробки програмного забезпечення. Аспектно-орієнтоване програмування дозволяє нам по-іншому думати про структуру програми, уможливлюючи модульність проблем. Це допомагає розбити логіку на частини, відомі як концерни, а концерни допомагають розділити бізнес-логіку програми та підвищити модульність. Порівняно з ООП , АОП не має порівняння, оскільки АОП виник із парадигми ООП. Так само, як клас є ключем до модульності ООП, аспект є ключем до модульності ООП.

Незалежно від того, подобається нам це чи ні, тести є частиною нашого життя, будь то в школі чи університеті. Щоб оцінити програми, проводиться тестування, щоб дізнатися, чи працюють вони відповідно до очікувань і документації. Функція впровадження залежностей робить фреймворки більш придатними для тестування, і це одна з причин, чому Spring підходить для тестування. Слабкий зв'язок також допомагає в модульному тестуванні, оскільки таким чином класи можна тестувати незалежно, не залежачи один від одного. Складно тестувати складні проекти за один раз, тому Spring є хорошим варіантом для розробки проектів, оскільки легко перевірити його функціональність. Spring ефективно обробляє не лише внутрішні ресурси, але й зовнішні ресурси, такі як

файли властивостей, файли зображень і файли XML. Resource і ResourceLoader — це інтерфейси, наявні у Spring для обробки зовнішніх ресурсів.

Spring Boot є частиною популярного фреймворку Spring, який використовується для швидкого запуску програм Spring. Micronaut — це платформа на основі JVM, створена для усунення деяких недоліків Spring/Spring Boot. Буде порівняння легкості створення нової програми (рисунок 1.7), підтримку мов та інші параметри конфігурації.

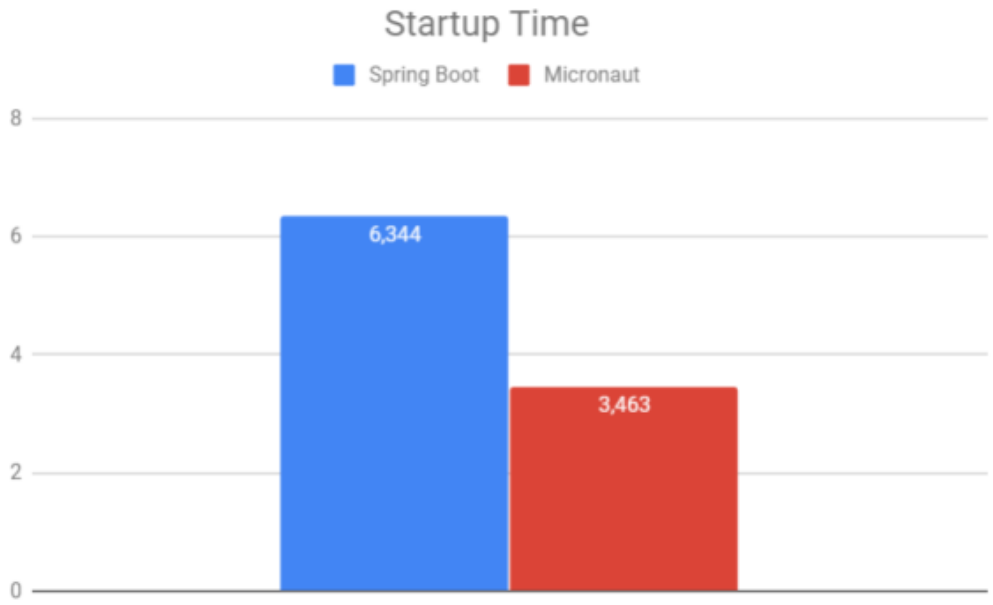

Рисунок 1.7 – Порівняння часу запуску

Micronaut, і Spring Boot пропонують кілька зручних методів для створення нових програм. Наприклад ми можемо створити нову програму, використовуючи будь-яку структуру з інтерфейсом командного рядка. Крім того, ми можемо використати Spring Initializr для Spring Boot або подібний інструмент для Micronaut під назвою Launch. Що стосується підтримки IDE, ми можемо використовувати плагіни Spring Boot для більшості популярних IDE включаючи його різновид Eclipse, Eclipse Spring Tools Suite. Якщо ми використовуємо IntelliJ, у нас є плагін Micronaut.

Коли поглянути до підтримки мови, ми побачимо, що вона майже ідентична для Spring Boot і Micronaut. Для обох фреймворків ми можемо вибрати між Java,

Groovy або Kotlin . Якщо ми обираємо Java, обидва фреймворки підтримують Java 8, 11 і 17. Крім того, ми можемо використовувати Gradle або Maven з обома фреймворками, як системи збірки.

Використовуючи Spring Boot, наша програма за умовчанням використовуватиме Tomcat (рисунок 1.8). Однак ми можемо налаштувати Spring Boot на використання Jetty або Undertow .

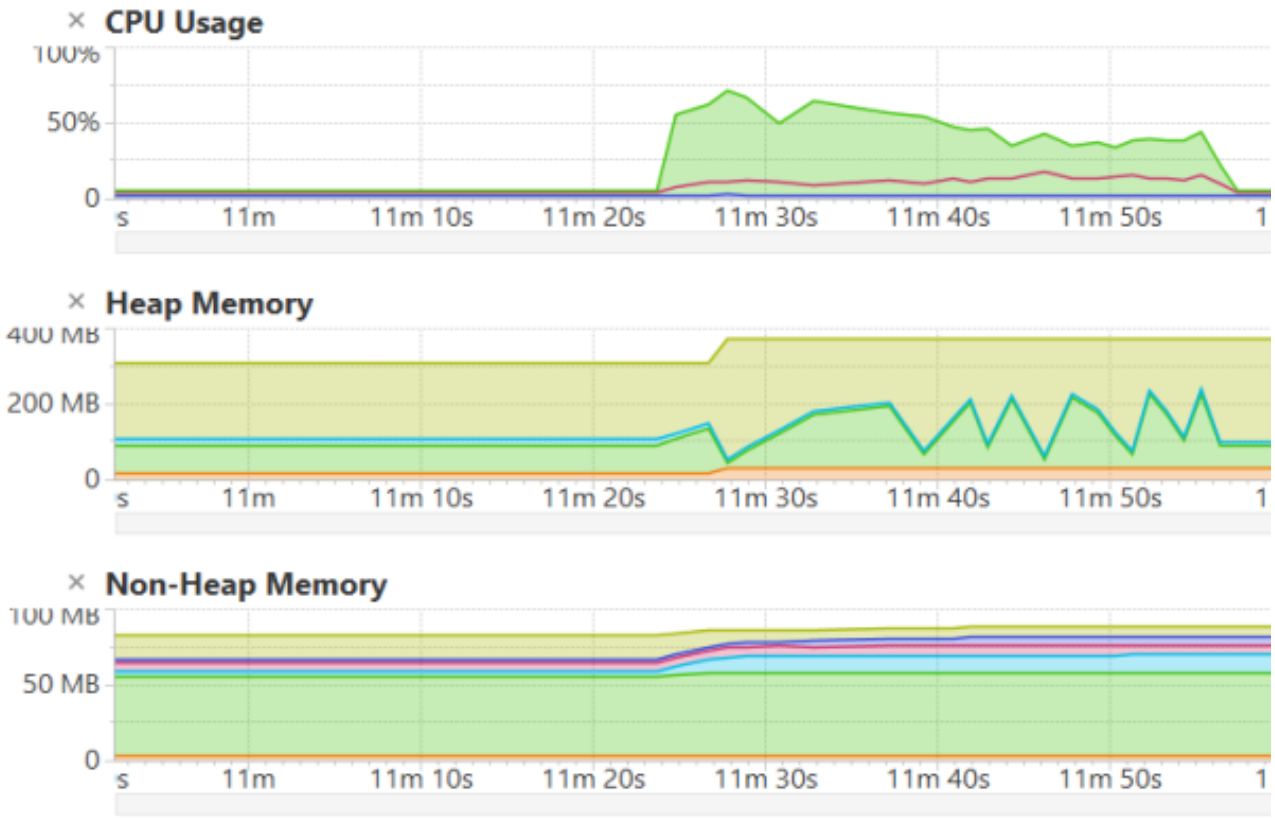

Рисунок 1.8 – Використання ресурсів Spring

Наші програми Micronaut за замовчуванням працюватимуть на HTTP-сервері Netty. Однак ми можемо переключити нашу програму на Tomcat, Jetty або Undertow. Для Spring Boot ми можемо визначити наші властивості в application.properties або application.yml. Ми можемо використовувати конвенцію application-{env}.properties для надання різних властивостей для різних середовищ. Крім того, ми можемо перевизначити ці надані властивості файлів програм за допомогою системних властивостей, змінних середовища або

атрибутів JNDI. В Micronaut можемо використовувати application.properties, application.yml і application.json для наших файлів властивостей. Якщо нам потрібно змінити будь-які властивості, ми можемо використовувати системні властивості або змінні середовища.

Якщо ми використовуємо обмін повідомленнями за допомогою Spring Boot, нам доступні Active MQ, Artemis, Rabbit MQ і Apache Kafka. На стороні Micronaut у нас є Apache Kafka, Rabbit MQ і Nats.io як варіанти. Spring Boot пропонує п’ять стратегій авторизації: базову, авторизацію через форму, JWT, SAML і LDAP. Якщо ми використовуємо Micronaut, ми маємо ті самі параметри тільки без SAML. Обидва фреймворки забезпечують підтримку OAuth2. З точки зору фактичного застосування безпеки, обидва фреймворки дозволяють нам використовувати анотації для захисту методів.

Обидва фреймворки дають нам можливість відстежувати різні показники та статистику в наших програмах. Ми можемо визначити власні кінцеві точки в обох фреймворках. Ми також можемо налаштувати безпеку кінцевої точки в обох фреймворках. Однак актуатор Spring Boot забезпечує кілька вбудованих кінцевих точок на відміну від Micronaut.

Ми можемо створювати повні додатки з стеком з обома фреймворками, використовуючи надані мови шаблонів для візуалізації інтерфейсу. Для Spring Boot ми маємо вибір між Thymeleaf, Apache Freemarker, Mustache і Groovy. Ми також можемо використовувати JSP, хоча це не рекомендується. У Micronaut маємо вибір між варіантами Thymeleaf, Handlebars, Apache Velocity, Apache Freemarker, Rocker, Soy/Closure і Pebbles.

Було всебічно порівняно різні функції фреймворків Micronaut і Spring Boot. Spring Boot все ще лідирує в багатьох аспектах завдяки підтримці величезного досвіду, який він набув у роботі з микросервисами протягом багатьох років. Але Micronaut (Young chap) є перспективним і достатньо перспективним, щоб скласти жорстку конкуренцію Spring Boot в майбутньому.

1.3 Обґрунтування вибору бази даних

База даних - це структурована система для зберігання та організації даних. База даних зберігає дані відповідно до певних правил. В залежносності від потреб проекту, проблематики змінюються правила організації даних, що визначають певну базу даних, тому важливо правильно вибрати потрібну систему. При виборі потрібно врахувати характеристики системи та характеристики проекту. До характеристиків проету можна віднести наступне розмір даних, навантаження, час роботи [12]. До характеристиків системи потрібно віднести наступні характеристики безпека, цілісність даних, здатність отримати до них швидкий доступ, надійність, можливість обслуговування багатьох клієнтів одночасно також можливість правильно переживати збої та проблеми з обладнанням без пошкодження даних. Для проекту важливим чинником являється збереження цілісності даних та можливість опрацьовувати збої системи без спотворення даних.

У сучасному технологічному суспільстві цілісність і безпека мають пріоритетне значення.Особливо якщо горити в контексті великих проектів, які вимагають зберігання великої кількості блоків інформації в одному надійному місці [13]. Таким чином вибір відповідної бази даних для магістерської роботи є необхідним кроком. І щоб зробити це правильно потрібно знати всі можливі варіанти, їх плюси і мінуси. У цьому розділі буде описано основні вимоги до системи управління бази даних, буде порівняно обрану базу даних із основними конкурентами, буде обґрунтовано мій вибір.

В проекті використовується СУБД PostgreSQL. Протягом кількох останніх десятиліть лідером серед варіантів систем управління баз даних (СУБД) є Postgresql. Це передова об'єктно реляційна СУБД з відкритим кодом, яка використовує мову SQL. Postgres дозволяє безпечно зберігати великі та складні дані. Він допомагає розробникам створювати надзвичайно складні програми, виконувати адміністративні завдання та створювати цілісні середовища.

Починаючи з 1986 року, року створення PostgreSQL, у цієї бази даних було багато прихильників і критиків. Щоб зберегти актуальність СУБД, регулярно оновлюється новими функціями. Кількість конкурентів Postgres також зростає. Щороку користувачам стають доступні нові бази даних.

На даному етапі є вибір між реляційними базами даних і не реляційними. Отже почнемо із самої суті системи Postgres - це її мова SQL. Це декларативна мова програмування для створення та роботи з даними в реляційній базі даних. У той же час NoSql, це скоріше набір підходів до зберігання даних в інших спосіб, ніж це робить sql. Не реляційні бази даних були створені в 1960 роках. Проте вони зуміли отримати справжню популярність тільки останніми часом. Цьому сприяла поява таких баз даних як MongoDB, CouchDB, Redis і Apache Cassandra. Ці два типи систем мають наступні відмінності:

- формат зберігання даних;
- спосіб отримання даних;
- безпека та практичність;
- масштабованість;
- наявність/відсутність транзакцій;
- цілісність даних.

Не реляційні бази даних переважно зберігають дані в json форматі, тобто ключ-значення форматі. Більше того, будь які подібні файли можна організувати в колекції. Структура NoSQL заснована на документі, всередині якого колекція ключ-значення, документи, бази даних графів або сховища широких стовпців, які не мають стандартної схеми, це також називається базою даних без зв’язків. База даних NoSQL горизонтально масштабована. Вона дозволяє легко додати ще декілька серверів у свою інфраструктуру бази даних NoSQL, щоб обробляти великий трафік, для простих запитів це забезпечує хорошу продуктивність, оскільки всі пов’язані дані містяться в одному документі, що усуває операції об’єднання. У наступних випадках ідеальним буде використання NoSQL DB:

- для роботи з великою кількістю даних. Дані можуть бути будь-якими: структурованими, напів структурованими чи неструктурованими;

- якщо у вас є локальні транзакції даних, які не повинні бути довговічними;

- коли від вас вимагається дотримання сучасних практик розробки програмного забезпечення;

- якщо вам потрібно займатися об'єктно-орієнтованим програмуванням;

- якщо ви використовуєте дані без схеми;

- якщо ви хочете мати економічну, але ефективну архітектуру;

- якщо реляційна база даних не здатна збільшити трафік у вашому бюджеті.

Реляційні системи зазвичай зберігають інформацію в таблицях. Таблиці пов'язані між собою і мають фіксованих шаблон даних. Тобто кожна таблиця має фіксовану кількість стовпців, які є строго типізованими. Така строгість є менш привабливою, але мінімізує кількість помилок.

У реляційних базах даних схема (рисунок 1.9) містить таблиці і тип полів у ній. Це означає, що для запуску проекту потрібно спроектувати базу даних перед будь якою бізнес логікою. Після того як схема створена досить важко внести будь які зміни. Для не реляційних систем таких обмежень немає. Користувачі можуть у будь який час вносити зміни у свою базу даних. Без схемна база даних не потребує попереднього встановлення жорстокого дизайну системи.

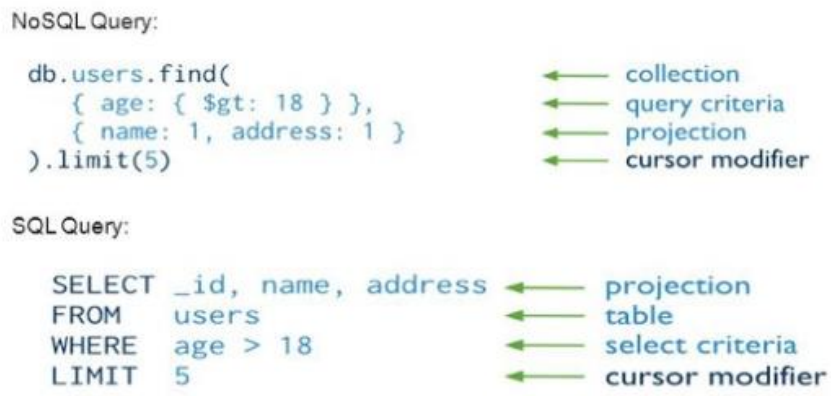

Рисунок 1.9 - Різниця між запитами

Ще однією корисною функцією реляційних СУБД це мова запитів. Реляційні СУБД використовують мову SQL. Це дає змогу отримати всі необхідні дані з обраної таблиці або декількох таблиць одночасно. Не реляційні не маю такої функції, все потрібно робити вручну. Також при роботі з не реляційними системами виникають проблеми пов'язані з безпекою і проблеми з функціонуванням. Проте ці проблеми викликані не системними прогалинами, а скоріш браком знань. Маючи менший досвід роботи з новими системами, розробник часто не дотримується вимог безпеки або допускає певні процедурні помилки.

Масштабованість для реляційних систем може бути досить проблематичною. Проте якщо у майбутньому буде вирішено розподілити пов'язані дані, можна буде розглянути кластеризацію кількох сервісів навколо одного центрального сховища. Для не реляційних систем це буде набагато простіше для реалізації. Масштабованість означає здатність системи обробляти зростаючий обсяг роботи або її потенціал виконувати більшу загальну роботу за той самий час, що минув, коли обчислювальна потужність розширюється для забезпечення зростання. Система називається масштабованою, якщо вона може збільшити своє робоче навантаження та пропускну здатність при додаванні додаткових ресурсів. Вертикальне масштабування — це процес додавання ресурсів, наприклад пам’яті або більш потужних ЦП, до існуючого сервера. Горизонтальне масштабування — це процес додавання додаткового обладнання до системи. Зазвичай це означає додавання вузлів (нових серверів) до існуючої системи (рисунок 1.10).

```
CREATE TABLE IF NOT EXISTS public.terminal
(
    id bigint NOT NULL DEFAULT nextval('terminal_id_seq'::regclass),
    create_date_time timestamp without time zone,
    update_date_time timestamp without time zone,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    location character varying(255) COLLATE pg_catalog."default" NOT NULL,
    name character varying(255) COLLATE pg_catalog."default" NOT NULL,
    company_id bigint,
    CONSTRAINT terminal_pkey PRIMARY KEY (id)
);
```

Рисунок 1.10 – Приклад створення таблиці

Цілісність і точність даних є основним пріоритетом для реляційних систем. Ще одним прикладом цього є механізм транзакцій. Такі бази даних дозволяють вміщувати дві і більше операції оновлення в одну транзакцію. Транзакції дотримуються принципу “все або нічого”, тобто всі оновлення можуть бути прийняті або відхиленні. У будь-якому випадку точність даних збережеться.

На противагу не реляційні не працюють з транзакціями, кожне оновлення приймається або відхиляється окремо. Це може призвести до спотворення даних які зрештою стануть не актуальними. Гарантії безпеки, які надають транзакції, часто описуються відомим абревіатурою ACID, що означає атомарність, консистенція, ізоляція, довговічність. Атомарність гарантує, що всі операції в транзакції розглядаються як єдиний «блок», який або повністю проходить, або повністю провалюється. Консистенція гарантує, що транзакція може лише перевести базу даних з одного дійсного стану в інший, запобігаючи пошкодженню даних. Ізоляція визначає, як і коли зміни зроблені однією транзакцією, стають видимими для іншої. Довговічність забезпечує постійне збереження результатів транзакції в системі. Зміни мають зберігатися навіть у разі втрати живлення або системних збоїв. Щоб база даних підтримувала найсуворіший серіалізований рівень ізоляції, потрібен такий механізм, як глобально впорядковані мітки часу, щоб послідовно впорядкувати всі транзакції. Отже було проаналізовано деякі основні відміності між sql і nosql. Однак не

потрібно поспішати робити припущення на вище сказаному. Не можна однозначно сказати що одна система краща за іншу. Хоча деякі прихильники цих баз даних вважають, що можуть. Це всього лише два різні підходи до одного - зберігання даних. Жодна з них не переважає над іншою. Ці системи не є конкурентними, а скоріше альтернативи. Це означає що вибір бази даних не повинен базуватися тільки на порівнянні, а швидше на проекті. Деякі проекти та компанії потребують баз даних SQL, тоді як NoSQL може бути зручним для інших. Існує навіть можливість використовувати їх як взаємозамінні. Що в деяких випадках являється найкращим варіантом. Також варто зазначити, що існують SQL бази даних які прийняли характеристики NoSQL, що робить перших ще більш цікавими для використання. Для програми із простою структурою даних підходять бази даних NoSQL в іншому випадку потрібно подумати про реляційну базу даних. В будь якому випадку для проектів на стадії ініціалізації реляційна база даних буде розумним рішення, оскільки вона може робити все що робить не реляційна система. Останні є більш гнучкими і добре відповідають потребам бізнесу. Наприклад швидкість, продуктивність і можливість до масштабування. Але потрібно пам'ятати, що іноді вони є занадто гнучкими що вимагає багато коду, для контролю вхідних даних та їх аналізу.

Найбільш розповсюдженою не реляційною базою даних являється MongoDB. Це розподілена база даних на основі документів, яка добре підходить для додатків. MongoDB підтримує документи JSON і пропонує гнучкі і динамічні схеми. Більше того ця система пропонує потужну мову запитів. За її допомогою ви можете фільтрувати та сортувати дані, а також виконувати численні типи пошуків. Нижче наведено основні характеристики MongoDB:

- база даних без схеми — це набір даних, у якому є кілька типів документів. Документ складається з кількох вмістів, які мають різні розміри, ця функція робить MongoDB гнучкою системою для баз даних;

- mongoDB є документоорієнтованою базою даних, mongodb зберігатиме дані у формі документів, а не у формі таблиць, дані зберігаються в полях, а не в рядках чи стовпцях, це забезпечує MongoDB хорошу гнучкістю;

- дані що зберігаються в MongoDB будуть індексовані на основі первинних і вторинних індексів, він спрямований на те, щоб зробити пошук і отримання збережених даних простішим і швидшим, це підвищує продуктивність і робить систему більш продуктивною;

- масштабованість MongoDB надає дві функції, а саме горизонтальну масштабованість із шардингом, шардинг — це розподіл даних через кілька різних серверів, за допомогою цього розподілу великі обсяги даних будуть розділені на менші групи за допомогою сегментного ключа, потім він продовжується, розділяючись на кілька різних серверів;

- реплікація в MongoDB — це функція, яка дозволяє робити копії даних і надсилати їх на різні сервери, таким чином, якщо один із серверів буде пошкоджено, дані залишиться в безпеці, оскільки вони були скопійовано на інший сервер;

- агрегація у MongoDB існує три типи агрегації, а саме: конвеєрна агрегація, функція зменшення карти та одноцільова агрегація, функція агрегації в MongoDB полягає в оперуванні даними для отримання єдиного або обчисленого результату.

Ось деякі з переваг програмного забезпечення MongoDB. Швидкість та ефективність, MongoDB забезпечує зручність завдяки швидкій та ефективній роботі бази даних у порівнянні з іншими типами баз даних [16]. Ця база даних використовує документи у форматі JSON, який легший у використанні. MongoDB поставляється з Memcached, який може зберігати дані у більшому масштабі. Простота в управлінні базою даних, створення бази даних за допомогою MongoDB полегшить роботу з керування даними. Використовуючи MongoDB, користувачам не потрібно створювати структури таблиць із збільшенням обсягу даних. MongoDB зможе автоматично додавати та створювати структури таблиць, коли користувачі вводять дані. Крім того, користувачам не потрібно вивчати мови

програмування, такі як SQL, щоб керувати базами даних, оскільки MongoDB інтегровано з Javascript. MongoDB має перевагу в тому, що вона здатна вмістити багато складних і різноманітних даних, оскільки використовує схему динамічної таблиці. За допомогою цієї схеми MongoDB може зберігати великі та різноманітні обсяги даних, від структурованих до неструктурованих. Структуровані дані — це набір даних, який містить інформацію, отриману безпосередньо через бази даних, наприклад дані про продажі та дані про співробітників. Тим часом неструктуровані дані – це дані, які потребують аналізу-оцінки, наприклад відео, звук, зображення та текстові дані на веб-сайтах. MongoDB краще керується запитами — це база даних, яка має можливість правильно керувати запитами. Використовуючи MongoDB, користувачам буде легше отримувати доступ до даних, що полегшить користувачам швидкий перегляд складних і неструктурованих результатів даних. Можливість масштабування, є наступною перевагою MongoDB. Користувачі можуть збільшити ємність бази даних, додавши кілька хмарних серверів. Це може статися тому, що MongoDB є типом бази даних NoSQL, на відміну від SQL, який вимагає масштабування IP для збільшення ємності та придбання нового обладнання з вищими специфікаціями. Перевага MongoDB, якої немає в інших базах даних, полягає в тому, що користувачі не зазнають простою під час внесення змін. Базу даних розроблено таким чином, щоб вона завжди була доступною, навіть коли в ній виконується процес оновлення структури даних. Безкоштовне використання MongoDB — це програмне забезпечення з системою баз даних, яке можна використовувати безкоштовно. Це програмне забезпечення також можна використовувати в MacOS, Linux і Windows. MongoDB - рекомендованим вибором для користувачів, які працюють над проектом, для якого не передбачено ліцензійний бюджет. Можна використовувати для різних варіацій даних. MongoDB має динамічну систему баз даних на основі схеми, яку можна використовувати для зберігання різних варіацій даних, як структурованих, так і неструктурованих. За допомогою динамічної схеми користувачі можуть зберігати дані без необхідності визначення схеми перед збереженням даних. Гнучкість MongoDB має динамічний дизайн

схеми в просторі пам’яті неструктурованих даних. MongoDB має таку гнучкість, щоб полегшити користувачам адаптацію до змін у світі даних, що швидко розвивається.

На додаток до переваг, MongoDB має ряд недоліків зокрема: Не підтримує транзакції. MongoDB не вимагає транзакцій, але використовує ACIS або Atomicity, Consistency, Isolation і Durability мехінізм, подібний до транзакцій. Якщо користувачеві потрібно оновити документ у MongoDB, можливе пошкодження даних. MongoDB вимагає багато пам'яті. База даних потребує великого обсягу пам’яті. Копіювання даних займатиме дуже великий простір пам’яті та швидко поповнюватиметься. Обмежений розмір даних MongoDB обмежує користувачів розміром даних, які можна використовувати в кожному документі, до 16 МБ. І продуктивність підтримується підтримується лише для 100 користувачів. Тому часто СУБД MongoDB дуже перевантажені. Також існують проблеми з індексуванням. Коли виникає проблема з індексуванням, її вирішення займає досить багато часу. Проблеми можуть виникнути, якщо реалізація індексу неправильна.

З іншої сторони маємо PostgreSQL. Починаючи з версії 9.4 додано деякі оптимізації та оновлення, але що найважливіше це додано HStore та Jsonb, бінарну версію для зберігання json. В цій системі реалізовано деякі функції noSQL, але не всі з них. Наприклад відсутнє горизонтальне масштабування. Та в будь якому разі ця реалізація все ще дає Postgres перевагу, оскільки тут поєднуються дві практики. Таким чином можна зберігати набір даних ключ-значення в таблицях. І це буде працювати правильно.

PostgreSQL може легко керувати складними запитами найбільших компаній та установ. Його ефективність перевірена роками. Крім того команда розробників ніколи не зупиняються на досягнутому та випускають нові і нові версії. Деякі з них також включали вдосконалення для не структурованих типів даних. Постійно пропонуючи нові інструменти та функції для забезпечення актуальності/дійсності даних. База даних забезпечує перевірку даних для будь якого поля json, введення

неправильних даних призведе до помилки. На відміну від MongoDB, працює з більшістю сучасних мов прграмування таких як: Python, Groovy, JavaScript, Java, C, C++, Perl та іншими. Postgres тепер підтримує всі формати даних: json, ключ-значення, XML. Ці формати також підтримуються MongoDB. Це робить Postgres непоганою альтернативою деяких не реляційних баз даних. Також Postgres забезпечує взаємодію з іншими джерелами даних. Є можливість отримувати дан з Oracle, MySQL, MongoDB, CouchDB, Redis, Neo4j, Twitter, LDAP, File, Hadoop та інших. Таким чином було вирішено почати зберігати дані в PostgreSQL. В мережі можна легко знайдете безліч навчальних ресурсів. Вони доступні на офіційному веб-сайті PostgreSQL і просто на інших веб-сайтах. Однак навчитися працювати в MongoDB може бути трохи складніше, оскільки його посібники та інструкції не так легко знайти.

Загалом, Postgres NoSQL має ряд переваг перед іншими базами даних NoSQL, такими як MongoDB. Причиною такого домінування є прийняття деяких функцій NoSQL. Навіщо вибирати лише один із них, SQL або NoSQL, якщо ви можете використовувати Postgres, який пропонує вам функції обох. У наступних абзацах буде порівняно технічні відмінності PostgreSQL та MySQL. Якщо говорити про продуктивність PostgreSQL є на порядок вищою за будь яку nosql. Загалом mysql також демонструє швидку роботу навіть при роботі з великими навантаженнями. Проте так було не завжди. У минулому MySQL мала репутацію швидкої бази даних за рахунок своєї паралельності. Postgres, навпаки, зазвичай демонструє збалансовані результати разом із паралельною обробкою. З останніми версіями обох баз даних усі відмінності стерто. Якщо порівнювати паралелізм, то переможцем тут є PostgreSQL. Ця база даних обробляє паралелізм набагато краще, ніж її конкурент MySQL. Це сталося завдяки Multiversion Concurrency Control, який реалізує Postgres. Postgres надзвичайно розширюваний. Він може підтримувати численні типи даних. Серед них: геометричні/ГІС, типи мережевих адрес, JSONB, власний UUID, мітки часу з урахуванням часового поясу. MySQL не пропонує таких же варіантів. Документація є ще однією сильною стороною PostgreSQL. У документації Postgres ви можете знайти відповідь майже на будь-

яке запитання. Хоча прихильники Oracle іноді критикують документацію PostgreSQL, вона все одно суттєво краща за MySQL. Порівнюючи адміністрування: У MySQL адміністрування набагато простіше. Втім, це скоріше уявна перевага — така простота є наслідком убогого функціоналу. Отже, це означає, що складні запити важко обробляти в MySQL. На відміну від цього, PostgreSQL легко впорається з ними. Щоб його правильно встановити, потрібно витратити деякий час. Однак з часом всі ваші зусилля будуть окуплені. Postgres підтримує тригери, які можуть реагувати на більшість типів команд, за винятком тих, які впливають на базу даних глобально, наприклад ролі та табличні простори. MySQL, у свою чергу, обмежений лише деякими командами.

З усього вищесказаного можна зробити висновок, що PostgreSQL є ідеальна базу даних для даного проекту. Ось дуже короткий, але інформативний підсумок:

- типи даних: PostgreSQL підтримує всі необхідні типи даних, такі як документи, примітиви, геометрія, структури тощо;

- цілісність даних: Postgres забезпечує цілісність даних, вводячи обмеження та регулюючі дані, які ви додаєте, за допомогою PostgreSQL ви можете забути про недійсність або втрату запису;

- продуктивність: розпаралелювання запитів на читання, потужні методи індексування, багатоверсійний контроль паралельності, це лише деякі з багатьох функцій, реалізованих PostgreSQL для підвищення та оптимізації продуктивності;

- продуктивність: розпаралелювання запитів на читання, потужні методи індексування, багатоверсійний контроль паралельності, це лише деякі з багатьох функцій, реалізованих PostgreSQL для підвищення та оптимізації продуктивності;

- інтернаціоналізація та текстовий пошук, Postgres підтримує міжнародні набори символів, він також забезпечує повнотекстовий пошук, щоб пришвидшити процес пошуку, і інтегрує сортування без урахування регістру та наголосу.

Підтримка не реляційних даних: це, мабуть, найважливіше оновлення бази даних. Підтримка документів JSON, XML, Hstore і Cstore фактично перетворює Postgres на базу даних NoSQL.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЕЛЕМЕНТНОЇ БАЗИ ДЛЯ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Обґрунтування вибору компонентів автоматизованої системи

В цьому розділі буде описано апаратна частина роботи, основне завдання якої зчитувати дані карток співробітників і надсилати на сервер для обробки і зберігання. Цей апарат (рисунок 2.1) повинен відповідати наступним умовам:

- пристрій повинен зчитувати тег RFID [17] і відправляти його на сервер через HTTP-запити;
- пристрій має надсилати мітку часу у форматі UNIX, а також має синхронізувати час із сервером;
- якщо сервер недоступний, теги слід зберегти в пам’яті та, коли мережа доступна, надсилати теги з пам’яті. Бажано обійтися без зовнішніх флешок;
- повинна бути можливість зберегти принаймні 300 записів у разі перерви зв’язку;
- має бути можливість встановити як статичні, так і динамічні IP-адреси для пристрою.

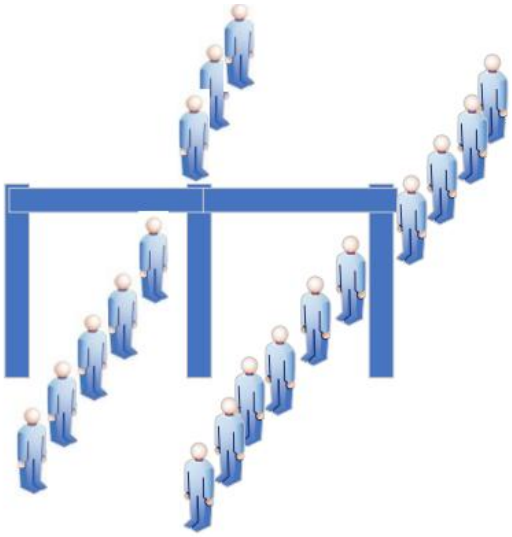

Рисунок 2.1 - Прохідна підприємства

При вході/виході потрібно встановити відповідні станції. Станція повинна бути компактною і працювати від будь-якого джерела живлення 5 вольт, PowerBank або через USB [18]. Для роботи приладу достатньо струму 300 мА. Корпус (рисунок 2.2) можна вирізати з акрилу товщиною 3 мм, фанери та інших матеріалів.

Рисунок 2.2 - Зовнішній вигляд станції

Система використовує наступні компоненти:

- mega (ATmega2560);
- ethernet Shield W5100;
- rfid зчитувач або Mifare RC522 (13.56MHz);
- at24c32 з I2C інтерфейсом;
- динамік.

Atmega2560 зазвичай зустрічається в Arduino Mega 2560 як основний мікроконтролер (таблиця 2.1). Це мікроконтролер на основі AVR RISC, який виконує потужні команди за один такт [19]. Це дає змогу досягти гарного балансу між енергоспоживанням і швидкістю обробки.

Таблиця 2.1 - Характеристики Atmega2560

Параметр	Значення
Розмір програмної пам'яті (Кб)	256
Швидкість ЦП (MIPS/DMIPS)	16
SRAM (КБ)	8,192
Комунікаційна периферія	4-UART, 5-SPI, 1-I2C
Діапазон температур (°C)	-40 до 85
Діапазон робочої напруги (В)	1,8 до 5,5
Таймери	2х 8-біт, 4 х 16-біт

Переваги контролера Atmega2560:

- низьке енергоспоживання при швидкому запуску;
- простий у використанні, оскільки 8-розрядний мікроконтролер менш складний, ніж 32/64-розрядні версії;
- qtouch suite дозволяє легко досліджувати, розробляти та налагоджувати власні сенсорні програми.

Недоліки контролера Atmega2560:

- обмежена кількість циклів запису флеш-пам'яті обмежує кількість разів перепрограмування, якщо їх запрограмовано на ПК;
- йому не вистачає додаткової продуктивності порівняно з мікроконтролерами з вищими розрядами.

Arduino Mega 2560 (рисунок 2.3) відомий своїми можливостями в роботі зі складними проектами, надає проектам багато простору та можливостей. Він рекомендований для 3D-принтерів і робото-технічних проектів завдяки 54 цифровим контактам введення/виведення, 16 аналоговим входам і великій пам'яті.

Рисунок 2.3 - Arduino Mega 2560

Технічні характеристики платформи Arduino Mega 2560:

- робоча напруга: 5В;
- вхідна напруга: 7-12В;
- вхідна напруга: 6-20В;
- цифрові контакти вводу/виводу: 54 (з яких 14 забезпечують вихід ШІМ);
- аналогові входи: 16;
- постійний струм введення/виведення: 40 мА;
- постійний струм для контакту 3,3 В: 50 мА;
- флеш-пам'ять: 256 КБ, 8 КБ використовується завантажувачем;
- sram: 8 Кб;
- тактова частота: 16 МГц.

Ethernet Shield W5100 (рисунок 2.4) дозволяє встановлювати дротове підключення Ethernet до мережі та сумісний з Uno, Mega [20]. Ця плата забезпечує можливість підключення Arduino до Інтернету або власної мережі, має мікросхему Wiznet W5100 Ethernet. Wiznet W5100 робить мережу доступною як за протоколом TCP так і за протоколом UDP. Цей чіп підтримує до 4 з'єднань, які можна тримати відкритими одночасно.

Рисунок 2.4 - Ethernet Shield W5100

Також є можливість швидко підключається за допомогою стандартної бібліотеки Arduino Ethernet. Цей модуль також має вбудований пристрій читання SD-карт, за допомогою якого можна легко зберігати отримані дані (рисунок 2.5).

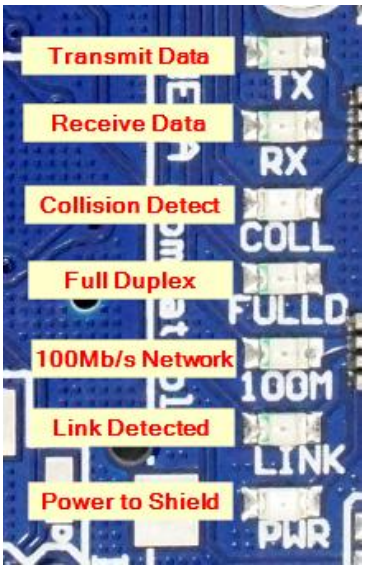

Рисунок 2.5 - Світлодіоди індикації

Модуль має наступні світлодіоди індикації:

- pwr: живлення увімкнено;
- link: успішне підключення до мережі;
- fulld: повне дуплексне підключення;
- 100m: увімкнено, коли виявлено мережу 100 Мбіт/с;

- rx: дані отримані;
- tx: надсилання даних;
- coll: виявлено мережевий конфлікт.

RFID подібний до інших технологій бездротового зв’язку, таких як радіопередавачі, Bluetooth [21]. Системи складаються з двох компонентів: тегів і зчитувачів (рисунок 2.6). Теги містять дані, а зчитувачі виявляють тег і обробляють інформацію з тегів, коли знаходяться в діапазоні.

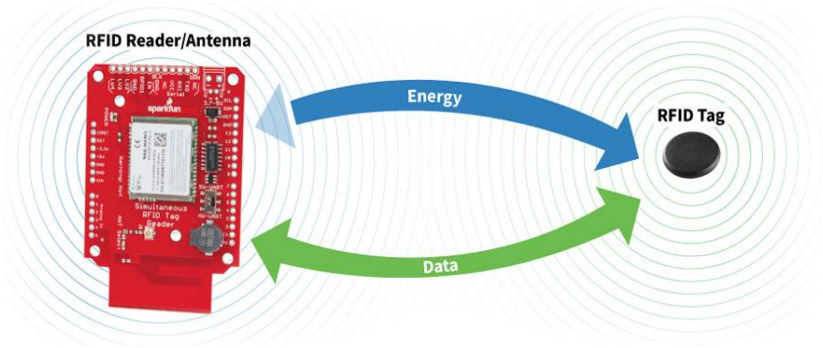

Рисунок 2.6 - RFID технологія

Теги мають невеликий об’єм пам’яті, де зберігається унікальний ідентифікатор тегу (TID), який не можна редагувати(рисунок 2.7). Невелика частина даних, що залишилася на тегах, може бути доступною лише для читання або запису.

Рисунок 2.7 – Динамік Д20

Пристрій автосигналізації, як біпер або зумер, може бути електромеханічним, п’єзоелектричним або механічним. Основною функцією цього пристрою перетворення сигналу на звук. Як правило, він живиться від напруги постійного струму та використовується в таймерах, пристроях сигналізації, принтерах, сигналізаціях, комп’ютерах тощо. В залежності від конструкцій він може генерувати різні звуки такі як будильник, музика, дзвінок і сирена.

Відповідний зумер можна підібрати за кількома основними параметрами (напруга, сила струму, режим руху, розмір, режим підключення/кріплення) і необхідним звуком (звуковий тиск і частота). Робоча напруга електромагнітного зумера може становити 1,5-24 В, а діапазон робочої напруги п'єзоелектричного зумера може становити 3В-220 В [22]. Однак за звичайних обставин діапазон робочої напруги п’єзоелектричного зумера рекомендовано вибрати більше 9 В, щоб отримати кращий звук. На рисунку 2.8 зображена фінальна компоновка станції.

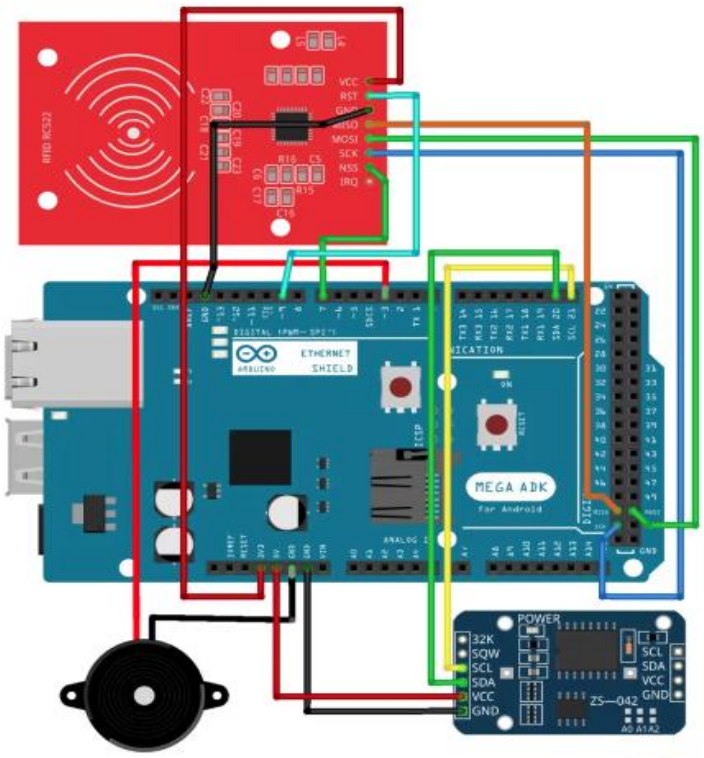

Рисунок 2.8 - Фінальна компоновка

Найкращим рішенням буде запаяти всі з'єднання, що уникнути втрати сигналу. Прошивка написана на C ++ з використанням бібліотек Arduino і скомпільовано її в Arduino IDE [23]. Знадобиться підключити наступні бібліотеки:

- ds3231 — бібліотека для роботи з модулем часу;
- spi — необхідний для роботи бібліотеки Ethernet;
- ethernet — бібліотека для роботи з Ethernet Shield;
- simpletimer — для псевдо паралельного виконання завдань на Arduino;
- mfrc522 — для роботи з модулем Mifare;
- onewire — не потрібно. Ми лише візьмемо функцію обчислення суми CRC, щоб перевірити цілісність даних;
- memoryfree — для очищення оперативної пам'яті від сміття;
- pgmStrToRAM — для зберігання та отримання тексту та FLASH-пам'яті в RAM;
- eeprom — для роботи з пам'яттю EEPROM ми будемо зберігати в ній невідправлені позначки.

Наступна діаграма (рисунок 2.9) зображує дизайн діаграму системи.

Рисунок 2.9 - Дизайн діаграма.

Система використовує клієнт серверну архітектуру. Модель клієнт-сервер, яка використовується в RESTful API, має бути відмово стійкою [24]. Очікується, що і клієнт і сервер працюватимуть незалежно. Зміни зроблені на стороні клієнта не повинні впливати на сторону сервера і навпаки. Архітектура клієнт-сервер відноситься до системи, яка розміщує і надає та керує більшістю ресурсів і послуг, які запитує клієнт. У цій моделі всі запити та послуги доставляються через мережу, її також називають моделлю мережевих обчислень або мережею клієнт-сервер. Архітектура клієнт-сервер, яку також називають моделлю клієнт-сервер — це мережева програма, яка розподіляє завдання та навантаження між клієнтами та серверами, які знаходяться в одній системі або об'єднані комп'ютерною мережею. Клієнт-серверна архітектура зазвичай складається з декількох робочих станцій користувачів, комп'ютерів або інших пристроїв, підключених до центрального сервера через підключення до Інтернету або іншу мережу. Клієнт надсилає запит з даними, а сервер приймає та обробляє запит, надсилаючи пакети даних назад користувачеві, якому вони потрібні [25]. Цю модель також називають мережею клієнт-сервер або моделлю мережевих обчислень.

Загальний алгоритм роботи:

- клієнт надсилає свій запит через мережевий пристрій;
- мережевий сервер приймає та обробляє запит користувача;
- сервер доставляє відповідь клієнту.

Технології постійно розвиваються і вдосконалюються, часто дуже швидкими темпами. Як результат сучасний бізнес все більше покладається на технології, особливо ІТ, щоб процвітати та залишатися конкурентоспроможними в середовищі. Таким чином, сучасним організаціям потрібна система, яка полегшує збір, обробку корпоративних даних, таким чином підвищуючи ефективність бізнес-процедур і забезпечуючи виживання на сучасному світовому ринку [26]. Модель мережі «клієнт-сервер» забезпечує вищий рівень обробки, що підвищує ефективність потужності робочої станції, розширення можливостей

робочої групи, дистанційне керування мережею, бізнес-орієнтований на ринок, і збереження наявних інвестицій.

Таким чином, клієнт-серверна архітектура забезпечує точну структуру, яка потрібна сучасним організаціям для вирішення проблем IT-світу, що швидко розвивається.

Архітектура клієнт-сервер зазвичай має такі характеристики:

- клієнтські та серверні машини зазвичай потребують різних апаратних і програмних ресурсів і надходять від інших постачальників;
- мережа має горизонтальну масштабованість, яка збільшує кількість клієнтських машин і вертикальну масштабованість, переміщує весь процес на більш потужні сервери або конфігурацію з кількома серверами;
- один комп'ютерний сервер може надавати кілька служб одночасно, хоча для кожної служби потрібна окрема серверна програма;
- і клієнтські і серверні програми взаємодіють безпосередньо з протоколом транспортного рівня, цей процес встановлює зв'язок і дозволяє об'єктам надсилати і отримувати інформацію;
- і клієнтському, і серверному комп'ютерам потрібен повний стек протоколів, транспортний протокол використовує протоколи нижчого рівня для надсилання та отримання окремих повідомлень.

Ось три приклади того, як використовується архітектуру клієнт-сервер в житті:

- сервери електронної пошти завдяки легкості та швидкості електронна пошта витіснила традиційну пошту як основну форму корпоративного спілкування, сервери електронної пошти за допомогою спеціального програмного забезпечення різних марок надсилають і отримують електронні листи між сторонами;
- файлові сервери: якщо ви зберігаєте файли в хмарних службах, таких як Google Docs або Microsoft Office, ви використовуєте файловий сервер, файлові сервери — це централізовані місця для зберігання файлів, до яких мають доступ багато клієнтів;

- веб-сервери на цих високопродуктивних серверах розміщено багато різних веб-сайтів, і клієнти отримують до них доступ через Інтернет.

Алгоритм доступу до веб ресурсу включає наступні кроки :

- клієнт/користувач використовує свій веб-браузер, щоб ввести потрібну URL-адресу;
- браузер запитує систему доменних імен (DNS) про IP-адресу;
- DNS-сервер знаходить IP-адресу потрібного сервера та надсилає її веб-браузеру;
- браузер створює запит HTTPS або HTTP;
- сервер/продюсер надсилає користувачеві правильні файли;
- клієнт/користувач отримує файли, надіслані сервером і процес повторюється за потреби.

Архітектура клієнт-сервер приносить свою частку позитивів і негативів сучасним цифровим споживачам. Почнемо зі списку переваг:

- це централізована система, яка зберігає всі дані та елементи керування в одному місці;
- це забезпечує високий рівень масштабованості, організації та ефективності;
- це дозволяє IT-персоналу окремо змінювати потужності клієнта та сервера;
- це економічно вигідно, особливо з точки зору обслуговування;
- це дозволяє відновлювати дані;
- це дозволяє балансувати навантаження, що оптимізує продуктивність;
- це дозволяє різним платформам обмінюватися ресурсами;
- користувачам не потрібно входити в інший процес, щоб отримати доступ до корпоративної інформації або настільних інструментів, таких як презентатори PowerPoint або утиліти для роботи з електронними таблицями;
- налаштування зменшує випадки реплікації даних.

Мінуси клієнт-серверної архітектури:

- якщо на сервері є вірус користувачі ймовірно підхоплять його оскільки мережа складається з пов’язаних клієнтів і серверів;
- сервер вразливий до атак типу DoS;

- пакети даних можуть бути підроблені або змінені під час передачі;
- якщо критично важливий сервер виходить з ладу;
- схильність до фішингу та атак MITM.

Трирівнева архітектура клієнт-сервер (рисунок 2.10) складається з рівня представлення, відомого як рівень інтерфейсу користувача, рівня додатків, який називається рівнем обслуговування, і рівня бази даних, що включає сервер бази даних. Трирівневу архітектуру можна розділити на три частини.

Рівень презентації (або рівень клієнта): цей рівень піклується про інтерфейс користувача. Прикладний рівень (або бізнес-рівень): цей рівень забезпечує обробку даних. Рівень бази даних (або рівень даних): цей рівень зберігає інформацію. Клієнтська система контролює рівень презентації, сервер додатків піклується про рівень додатків, а система сервера контролює рівень бази даних.

Рисунок 2.10 - Трирівнева архітектура клієнт-сервер

Однорангові мережі, також звані P2P-мережами, складаються з груп комп'ютерів, об'єднаних у мережу, де однорангові пристрої діють і як клієнт, і як сервер. Однорангові користувачі мають однакові обов'язки та права на роботу з даними. Ця установка радикально відрізняється від клієнт-серверної моделі, яка має чітко визначені групи користувачів і серверів. Це можна проілюструвати наступним чином, якщо ви зайшли ресторан швидкого харчування, підійшли до стійки та замовили гамбургер у прилавку, це були б відносини клієнт-сервер. Однак, якщо наступного дня ви зайшли в цей заклад і виявили, що там прибрали персонал, ви зробили замовлення а інший клієнт виконав ваше замовлення - це однорангова мережа.

Ось основні відмінності між двома мережевими моделями:

- мережі «клієнт-сервер» потребують центрального файлового сервера і, отже коштують більше;

- мережі клієнт-сервер розмежовують користувачів і провайдерів;
- мережі клієнт-сервер пропонують більший рівень безпеки, що робить їх безпечнішими, В іншому випадку кінцеві користувачі відповідають за безпеку однорангової мережі;
- чим більше активних вузлів у одноранговій мережі, тим більше страждає її продуктивність, мережі клієнт-сервер забезпечують кращу стабільність і масштабованість, ідеальний діапазон для мереж P2P – від двох до восьми користувачів;
- однорангові користувачі можуть обмінюватися файлами швидше та легше, ніж у мережі клієнт-сервер;
- якщо мережевий сервер клієнт-сервер виходить з ладу все зупиняється але якщо один вузол у мережі P2P виходить з ладу, решта залишається в робочому стані.

Протокол передачі даних — це набір угод логічного рівня, які визначають процес обміну даними між різними програмами, зокрема: які дані будуть передаватися, яка частина системи повинна бути задіяна, як має оброблятися інформація, що надходить, тощо. Протоколи передачі даних HTTP — це протокол який дозволяє передавати різні гіпермедійні ресурси, наприклад документи HTML. Протокол HTTP був розроблений для зв'язку між браузерами та серверами. Це основа будь-якого обміну даними в Інтернеті. HTTPS — це той самий протокол, що й HTTP. Буква S означає безпеку. Іншими словами, цей протокол має шифрування, яке запобігає несанкціонованому доступу до даних користувача. Дані надходять на сервер через протокол HTTP у тому самому вигляді, в якому їх надсилає користувач. Таким чином, зловмиснику досить легко перехопити ці дані та отримати доступ до особистої інформації користувача. У протоколі HTTPS ці дані зашифровані і доступ до них непростий. FTP є одним із основних протоколів для передачі файлів через мережу між комп’ютерами. POP3 є найпоширенішим протоколом для отримання електронної пошти. Це дозволяє завантажувати електронні листи на пристрій (наприклад, комп’ютер або смартфон) і видаляти їх із сервера [27]. Клієнт зв’язується із сервером,

надсилаючи йому запити. Сервер, у свою чергу, надає відповіді на ці запити. Щоразу, коли сервер отримує запит, він відповідає певним кодом. Деякі коди, з якими найчастіше стикаються користувачі, наведені нижче.

Від 100 до 105 – інформаційні коди:

- 100, продовжити, сервер задоволений деталями запиту, клієнт може продовжувати надсилати інформацію;
- 102, обробка, запит прийнятий, але його обробка займає багато часу. Сервер використовує цю помилку, щоб запобігти розриву з'єднання клієнтом через час очікування.

Від 200 до 226 - це коди успіху:

- 200, ok, запит виконано успішно;
- 203, неавторизована інформація, запит було виконано успішно але інформація надходить із вторинного джерела й тому була змінена трансформуючим проксі-сервером;
- 204, немає вмісту, сервер успішно обробив запит, але немає додаткового вмісту для надсилання у відповідь.

З 300 по 307 повідомляють про переадресацію:

- 301, переміщено назавжди, запитуваний документ переміщено на нову URL-адресу.

Від 400 до 499 – помилки клієнта:

- 400, неправильний запит: сталася синтаксична помилка під час отримання запиту клієнта;
- 401, не авторизовано, для доступу до запитуваного ресурсу потрібна автентифікація;
- 403, заборонено, сервер розуміє запит але відмовляється його виконувати через обмеження доступу клієнта до зазначеного ресурсу;
- 404, не знайдено, найпоширеніша помилка, яка зазвичай виникає, коли адреса веб-сторінки написана з помилкою.

Від 500 до 510 це помилки сервера:

- 500, внутрішня помилка сервера, щось пішло не так на стороні сервера веб-сайту, але це не може бути більш конкретним. Іншими словами, він охоплює всі внутрішні помилки сервера поза межами помилок інших класів;
- 502, поганий шлюз, сервер, який діє як шлюз або проксі-сервер, отримав недійсне повідомлення відповіді від вищестоящого сервера;
- 503, сервіс недоступний, сервер тимчасово не може обробити запити з технічних причин (технічне обслуговування, перевантаження);
- 504, час очікування шлюзу, час очікування сервера перевищив час, наданий для отримання відповіді.

2.2 Обґрунтування вибору хостингу

У світі веб-хостингу веб-сервіси Amazon (або скорочено AW) дійсно змінили правила гри. Багато великих і авторитетних корпорацій зараз використовують їх як рішення для веб-хостингу. Від Netflix до готелів Kempinski і навіть NASA! Ця елітна клієнтська база демонструє не лише надійність AWS, але й потужність, яку вона пропонує. Простими словами, Amazon Web Services — це віртуальне рішення для веб-хостингу для організацій [28]. Будучи офіційною дочірньою компанією бренду Amazon, заснованого в 2002 році, вона пропонує хмарний підхід до хостингу на вимогу. AWS працює за моделлю платної підписки, яка має опцію безкоштовного рівня, доступну протягом першого року. Хмарна технологія, вбудована в AWS, надає користувачам повний доступ через Інтернет до набору віртуальних комп’ютерів. Вони мають усі звичайні атрибути стандартних комп’ютерів. Наприклад, попередньо завантажене прикладне програмне забезпечення та вибір операційних систем серед іншого.

Користувачі отримують доступ до своєї системи AWS через інтернет-браузер. Це дає можливість налаштувати свій веб-сайт і контролювати свій обліковий запис AWS. Технологію AWS можна знайти на серверних кластерах по всьому світу, де дочірня компанія Amazon обслуговує та доглядає за технологією.

Відомий як найбільший у світі публічний сервіс хмарних обчислень, він дійсно займає левову частку в цьому секторі. Якщо ви керуєте IT-відділом або фінансами вашої компанії, то ймовірно, ви шукаєте способи скоротити витрати. З Amazon Web Services це реальна можливість. Без попередніх контрактів, які зв'язують вас, ви платите лише за ті послуги та ресурси, які фактично використовуєте. Через приголомшливий обсяг системи AWS коштує дуже мало, щоб додати до мережі додаткову пропускну здатність або сховище. Ця економія коштів передається клієнтам, які можуть сподіватися на використання стільки, скільки їм потрібно, у той день чи тиждень, сплачуючи лише за те, що вони використали. Само собою зрозуміло, що це може принести вам величезну економію. Однією з проблем традиційного хостингу є можливість масштабувати його вгору або вниз на основі тимчасового попиту.

Величезні можливості масштабування, які пропонує AWS, справді дають йому велику перевагу перед стандартними варіантами веб-хостингу. Не тільки легко збільшити пропускну здатність, якщо ви очікуєте більше трафіку в певний день, ви також можете зменшити його, якщо у вас також період спаду. Усе це легко зробити кількома натисканнями кнопки у вашій консолі AWS, а це означає, що AWS завжди буде під рукою, щоб майже миттєво реагувати на потреби вашого веб-сайту. Варто відзначити чудову гнучкість, яку він пропонує. Завдяки можливості вибору таких функцій, як операційна система, якій ви віддаєте перевагу, яку мову програмування та яку платформу веб-додатків використовувати, у вас є багато варіантів для вибору. Налаштування AWS означає, що ви можете вибрати конкретне програмне забезпечення та служби, які ви хочете використовувати [29]. Це не тільки означає, що AWS можна повністю налаштувати відповідно до ваших вимог, це також спрощує процес міграції для будь-яких існуючих програм. Ще одна чудова перевага AWS — це гнучкість створення нових рішень. Тут ключовим моментом є те, наскільки легко ним користуватися, розроблений так, щоб бути достатньо простим для будь-кого, він зробить керування та обслуговування вашого веб-сайту легким. Консоль AWS, на якій ви працюєте, добре спланована та інтуїтивна. Природно для ефективної

роботи потрібен певний рівень знань у цій сфері. Однак, порівняно з традиційними схемами розміщення, це бажаний крок вперед. Спосіб створення AWS також дозволяє постачальникам програм безпечно та швидко розміщувати програми. Це стосується незалежно від того, чи це існуюча програма, чи нова на основі SaaS [30]. Усе це в сукупності означає, що ви можете бути в безпеці, знаючи, що ваш веб-сайт завжди буде захищений, коли він онлайн, і матиме надійний хостинг, на який можна покластися.

AWS фактично здає в оренду кінцевим користувачам свої комп'ютери, розташовані в захищених центрах обробки даних по всьому світу, на всіх континентах, з'єднаних кабелями. А оскільки комп'ютери орендовані, їх можна використовувати гнучко[31]. Ці віртуальні комп'ютери мають більшість атрибутів справжнього комп'ютера, включити апаратне забезпечення RAM пам'ять, жорсткий диск/SSD; вибір операційних систем, мережа; і попередньо завантажене прикладне програмне забезпечення, таке як веб-сервери, бази даних, CRM тощо. Кожна система AWS також віртуалізує консоль із клавіатурою, дисплеєм і мишею, що дозволяє абонентам підключатися до своєї системи за допомогою браузера. Браузер діє як вікно у віртуальний комп'ютер, дозволяючи передплатникам входити, налаштовувати та використовувати свої віртуальні системи так само, як вони роблять це з фізичним комп'ютером. Залежно від того, що потрібно абоненту, він може використовувати один сервер, десять чи сто – фактично скільки завгодно, з підтримкою будь-яких існуючих [32]. Завдяки тому, як він налаштований, Amazon може впоратися з будь-якими вимогами до даних, а витрати можуть залишатися порівняно розумними, незалежно від того, наскільки масштабна інсталяція зростає з часом [33]. Одним із найбільших і найвідоміших клієнтів є Netflix, компанія онлайн-прокату фільмів, яка як повідомляється, має три-чотири тисячі серверів, що працюють у будь-який в мережі Amazon. Якщо ваш веб-додаток тільки починає працювати, оренди одного або, можливо, двох серверів буде достатньо, оскільки вам не знадобляться величезні серверні ресурси. Середній сервер може коштувати вам менше фунта на годину, а дуже великий — лише три песни на годину.

Корпорація Майкрософт має власну платформу під назвою Azure, яка дещо відрізнятися від Amazon. Amazon надає сервери кінцевим користувачам, незалежно від того, чи працюють вони з Linux чи Windows. Це робить бізнес-модель Amazon спочатку більш зручною для користувачів. AWS дає змогу використовувати будь-який код на орендованій системі під час створення вашої програми, незалежно від платформи, знаючи що клієнт поступово переросте у використання більшої кількості послуг Amazon і стане прив'язаним до них. Наразі Azure також подібно до Amazon – у нього є віртуальні сервери, які можна орендувати.

Коли ви створюєте комп'ютерну систему, наприклад, для запуску веб-програми, ви хотіли б, щоб вона постійно була онлайн. Правда полягає в тому, що абсолютна безвідмовна надійність є досить недосяжною метою. Статистика говорить нам, що будь-який веб-сервер може мати збій раз на кілька років. Навіть якщо політика хостингової компанії передбачає, що вона негайно замінить несправний сервер і скопіює на нього останній стан даних, це може зайняти кілька годин. Усунення цього ризику призведе до повного дублювання ваших серверів.

Гнучкі варіанти упаковки Spring Boot надають великий вибір, коли справа доходить до розгортання програми [34]. Ви можете розгорнути програми Spring Boot на різних хмарних платформах, на віртуальних/реальних машинах або зробити їх повністю виконуваними для систем Unix. Elastic Beanstalk забезпечує простий спосіб налаштування та розгортання програм у сервісах AWS. Існує два способи розгортання програм у середовищі Elastic Beanstalk з використання консолі AWS або з використанням EB CLI (утиліта командного рядка). Нижче наведена схема (рисунок 9) представляє робочий процес розгортання програм у Elastic beanstalk. AWS Elastic Beanstalk — це простий у використанні сервіс для розгортання та масштабування веб-додатків і служб, розроблених за допомогою Java, .NET, PHP, Node.js, Python, Ruby, Go та Docker на звичних серверах, таких як Apache, Nginx, Passenger і IIS (рисунок 2.11). Це є найшвидший і найпростіший спосіб розгорнути свою програму. За кілька хвилин програма буде готова до

використання без необхідності користувачам мати справу з базовою інфраструктурою або конфігурацією ресурсів.

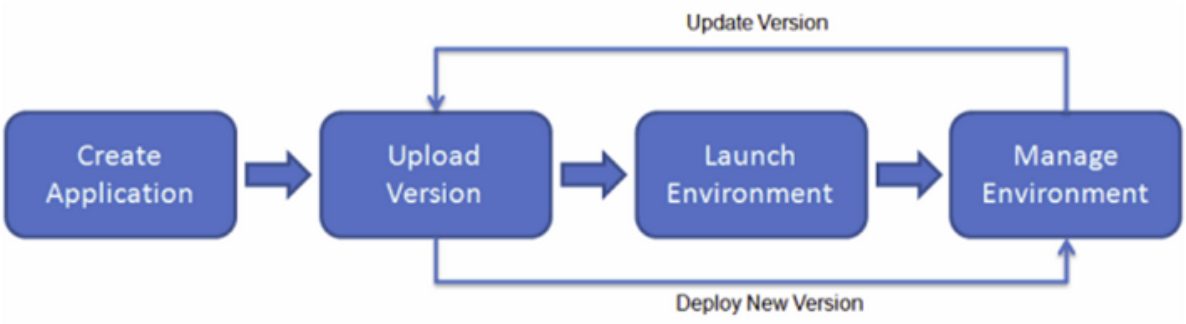

Рисунок 2.11 – Процес створення нової версії

В першу чергу, для хостингу потрібно згенерувати jar з залежностями, так званий ‘fat jar’ (рисунок 10). Після успішної збірки файл jar зявится папці. Потрібно скопіювати файл jar і зберегти його у системі в зручному місці. Файл jar — це просто тип архіву, який використовується для пакування файлів класів Java і пов’язаних ресурсів для розповсюдження. Простими словами файл JAR – це файл, який містить стислі версії файлів .class, аудіофайлів, файлів зображень або каталогів. Його можна уявити як заархівований файл (.zip), створений за допомогою програмного забезпечення WinZip (рисунок 2.12). Навіть програмне забезпечення WinZip можна використовувати для роботи із вмістом .jar. Тож можна використовувати їх для таких завдань, як стиснення даних без втрат, архівування та розпакування архіву.

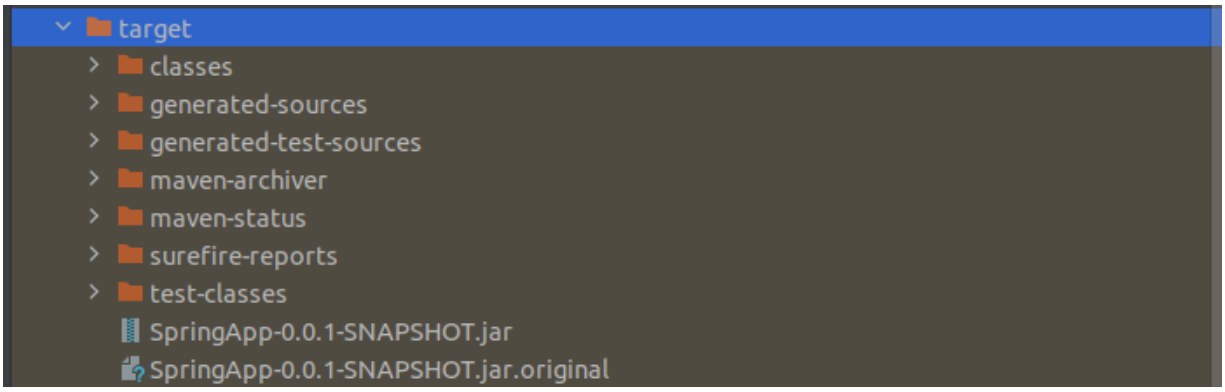

Рисунок 2.12 – Структура папки target

Наступним кроком буде розгорнути jar файл в AWS Elastic Beanstalk, потрібно ввійти в консоль керування AWS (рисунок 2.13) і відкрити службу Elastic Beanstalk.

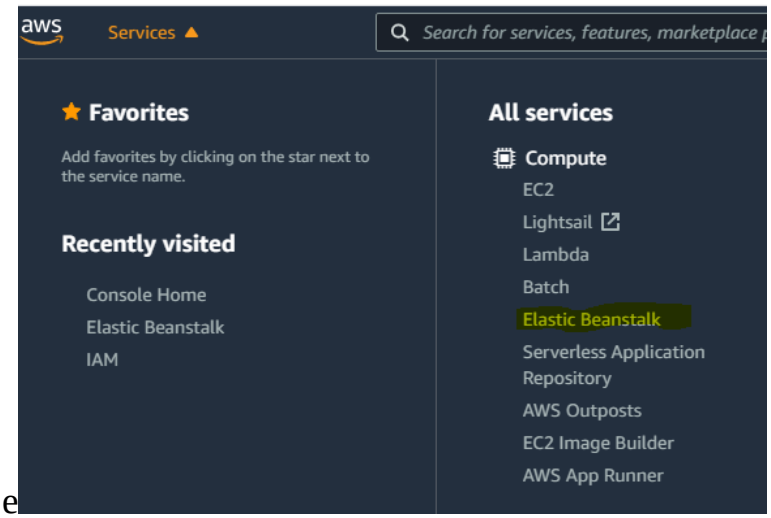

Рисунок 2.13 - Сервіси aws

Потім на домашній сторінці Amazon Elastic Beanstalk потрібно натисніть кнопку «Створити програму», щоб створити нову програму (рисунок 2.14). І ввести назву в поле назви програми. Тут ми назвемо додаток my-spring-app.

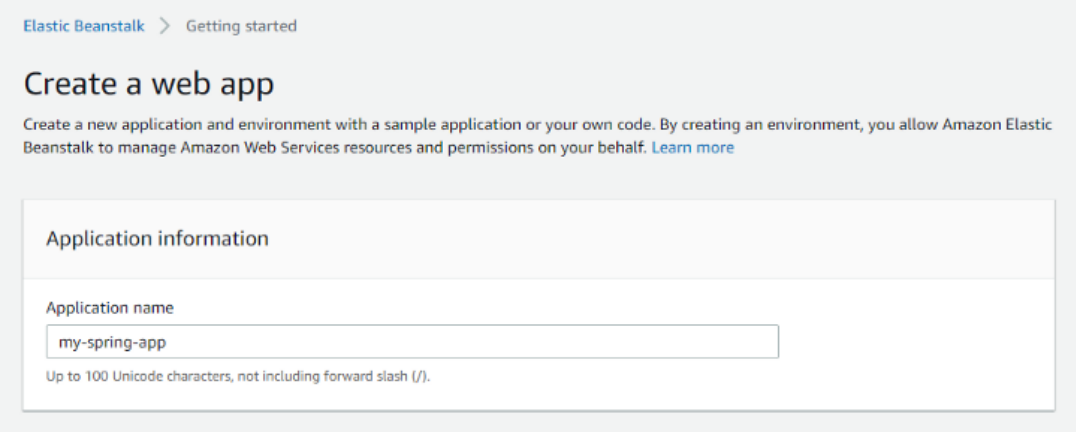

Рисунок 2.14 – Створення додатку

Потім потрібно вибрати деталі платформи в розділі платформи. Було обрано платформу для Java, оскільки збираємося розгортати програму Java (рисунок 2.15). Також у розділі Код програми виберіть опцію Завантажити свій код.

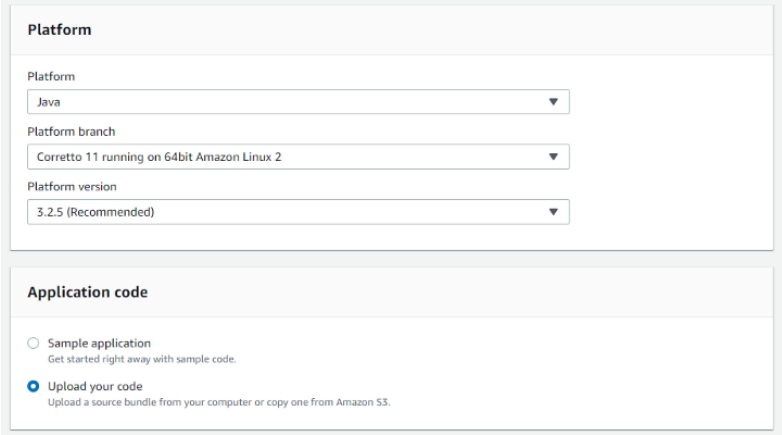

Рисунок 2.15 - Завантаження коду

Нарешті потрібно вибрати jar-файл програми для завантаження Spring, згенерований зі збірки maven (рисунок 2.16), як показано нижче.

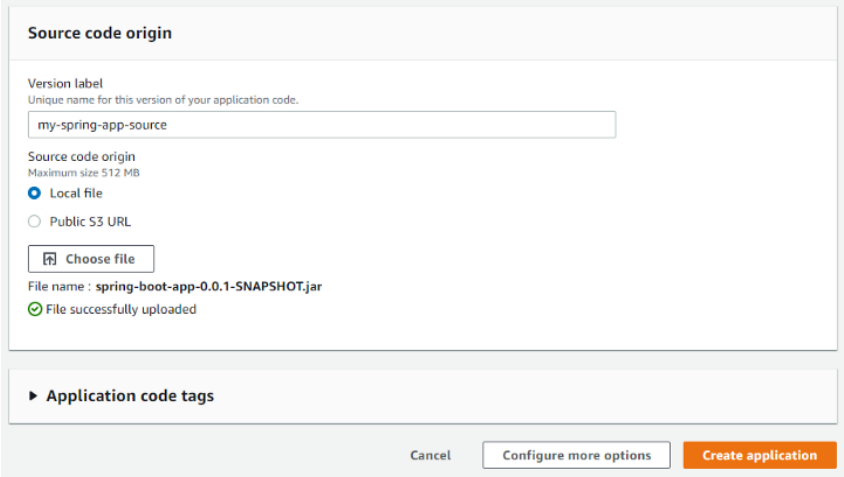

Рисунок 2.16 - Завантаження jar

Далі потрібно натиснути кнопку «Створити програму», і програма почне розгортатися в AWS Elastic Beanstalk (рисунок 2.17).

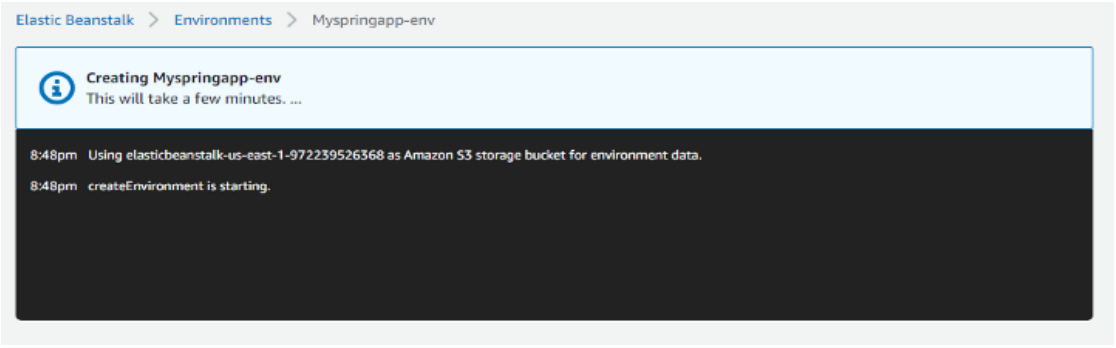

Рисунок 2.17 – Веб консоль

Після успішного розгортання програми можна побачити запис у розділі «Середовища», як показано на рисунку 2.18.

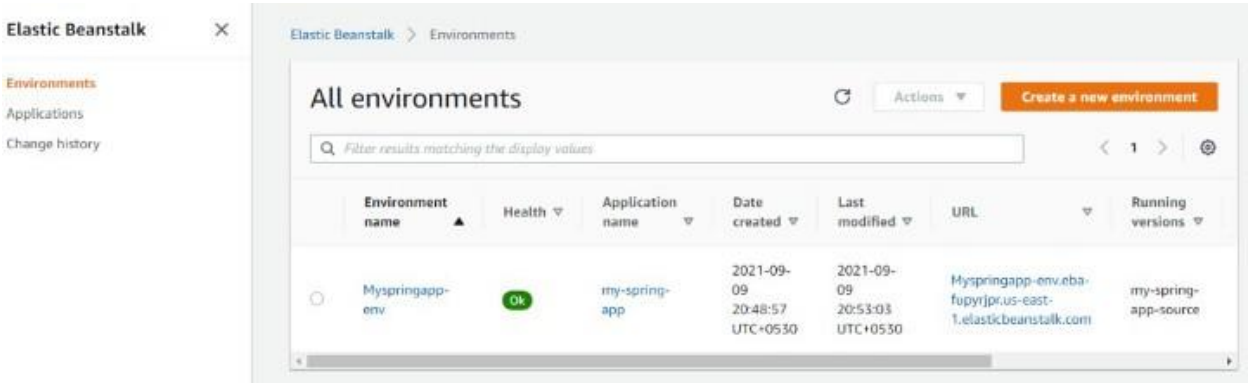

Рисунок 2.18 – Створення додатку

Також можна бачити, що стан програми в порядку, а на екрані — останні події (рисунок 2.19).

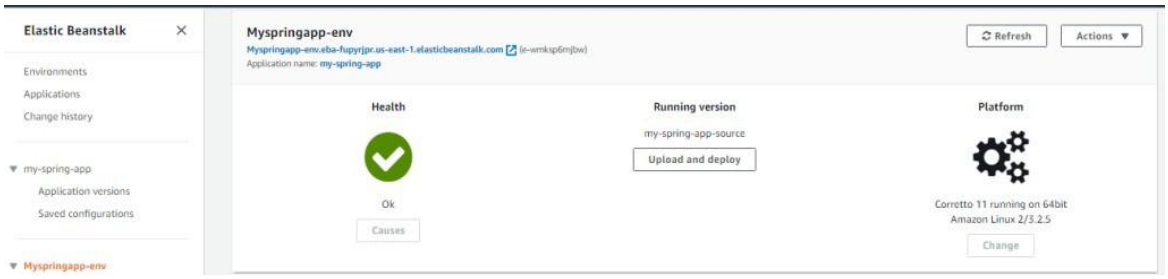

Рисунок 2.19 – Стан додакта

Можна видалити екземпляр програми, просто вибравши параметр «Видалити програму» в меню «Дії» (рисунок 2.20). Для завершення операції видалення AWS може знадобитися деякий час.

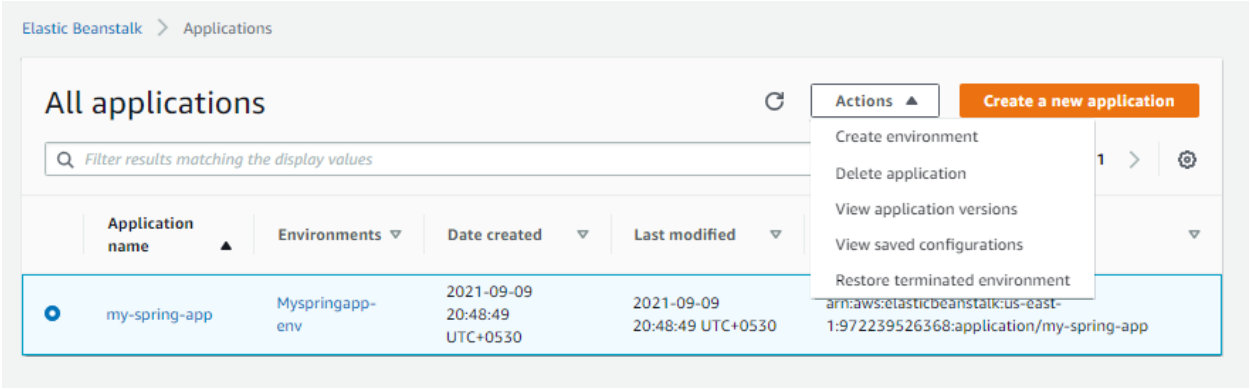

Рисунок 2.20 - Операції над додатком

Основні переваги Elastic Beanstalk включають конфігурацію сервера, що економить час. Ця автоматизація може заощадити дорогоцінний час, обробляючи всі речі, які потрібно виконати для робочої програми (налаштування ротації файлів журналу, конфігураційних файлів nginx, налаштування служби puma, встановлення пакетів Linux, встановлення Ruby, налаштування балансувальника навантаження та налаштування бази даних). Є інші сервіси, як-от Heroku, Engine Yard та інші, які також роблять це але загалом Elastic Beanstalk на рівень кращий. Деякі з найбільших недоліків Elastic Beanstalk включають ненадійне розгортання, відсутність документації щодо оновлення стека та додатків, а також загальну відсутність чіткої документації.

Удосконалення нашого процесу розгортання за допомогою контейнерів, таких як Docker, додасть ще більшої універсальності. Завдяки тонкому контролю, який пропонує Elastic Beanstalk, ми можемо вибрати технології, які найкраще підходять для нас. Зрештою найбільш корисним у Elastic Beanstalk є те, що його функції автоматизації дозволяють нам легко розгортати та оновлювати програми. Хоча це, звичайно, не ідеальний інструмент, але якщо ви прагнете скоротити системні операції та просто зосередитися на тому, що розробляєте Elastic Beanstalk — надійний вибір.

3. ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБЛІКУ РОБОЧИХ ГОДИН

3.1 Розробка методу високошвидкісної обробки сигналів у розподілених сенсорних мережах збору і обробки інформації

Радіочастотна ідентифікація (RFID) – це технологія, що швидко розвивається, яка забезпечує можливість бездротової ідентифікації та відстеження за допомогою простих пристроїв, які називаються тегами, і більш складними пристроями, що називається читачами. RFID — це нова технологія, яка є однією з найшвидше зростаючих сегментів сучасної галузі автоматичної ідентифікації та збору даних (AIDC). Зараз RFID використовується для сотень, якщо не тисяч, додатків. RFID є революція в управлінні ланцюгом поставок, заміна штрих-кодів як основного об’єкта система відстеження, і вона швидко стає економічно ефективною технологією [35]. Зниження собівартості є особливу увагу при впровадженні мікрохвильових систем RFID, оскільки вони зазвичай використовують активні теги, енергоспоживання яких має бути мінімізоване.

Пристрій RFID функціонує з тією ж метою, що і штрих-код або магнітна смужка на звороті кредитної картки чи картки банкомату. Це дозволяє зберігати унікальний ідентифікатор для цього об'єкта. І так само, як штрих-код або магнітну смужку потрібно сканувати, щоб отримати інформацію, пристрій RFID потрібно сканувати, щоб отримати ідентифікаційну інформацію.

Найважливішим параметром мітки, що вказує на продуктивність системи RFID є діапазон читання. Тобто максимальна відстань, на якій читач може прочитати інформацію з тегу. Оскільки чутливість читача зазвичай вища порівняно з чутливістю тега, діапазон читання визначається порогом відповіді тегу. Діапазон читання також чутливі до орієнтації тегу, матеріалу, до якого прикріплено тег, і середовища. Діапазон читання r можна розрахувати за допомогою формули вільного простору Фріїса:

$$r = \frac{\lambda}{4\pi}\sqrt{\frac{EIRP \cdot G_r \cdot \tau}{P_{chip}}},$$

де λ — довжина хвилі, а EIRP — еквівалентна ізотропно випромінювана потужність, визначається місцевими правилами країни (EIRP = 3,3 Вт в Європі), P — мінімальна порогова потужність необхідна для активації RFID-чіпа, G — коефіцієнт посилення приймальної антени-мітки, τ – коефіцієнт передачі потужності між чіпом і антеною.

RFID складається із трьох основних компонентів (рисунок 3.1): зчитувач, тег і антена. Зв'язок між антенами та зчитувачем називається Downlink, а зв'язок між мітками та зчитувачем називається Uplink.

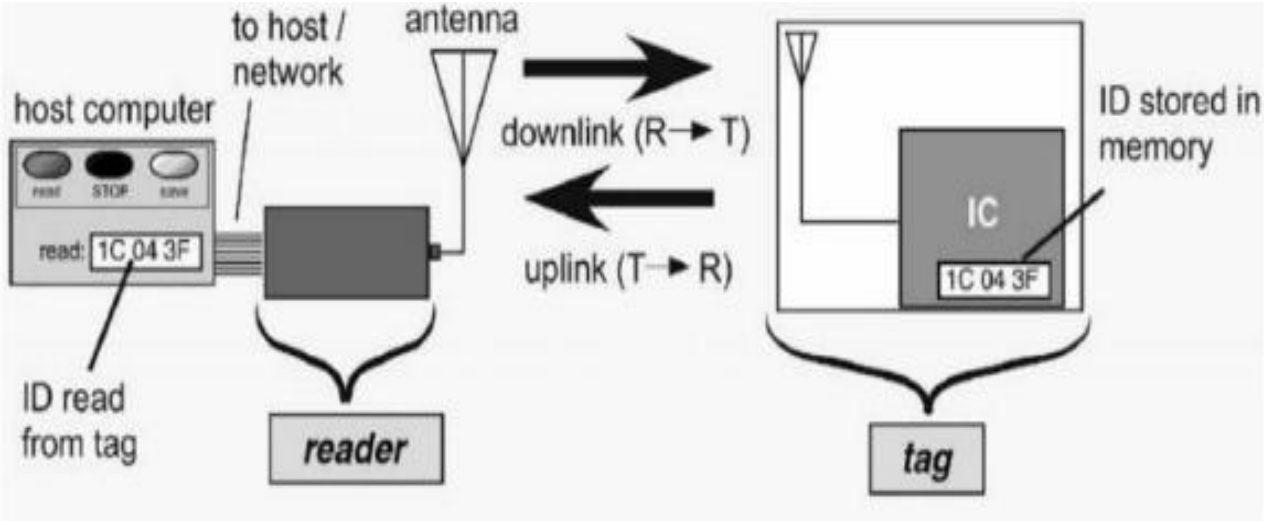

Рисунок 3.1 – Базовий RFID пристрій

Коли працівник прикладає RFID тег до терміналу, зчитувач зчитує tag_id з памяті тегу. Зчитувач виявляє тег і генерує вибірку даних в настпному форматі (ідентифікатор тегу, знаходження, час) та надсилає дані на сервер. Кількість подібної інформації буде досить велика. Після проведення аналізу було виявло, що велика кількість інформації дублюється. Так як працівник проходить кожній контрольно пропускний пункт двічі, перший раз коли заходить і другий раз коли виходить. Якщо не враховувати підприємства де можлово декілька входів/виходів,

$(\text{ТЕГ1}, A, 3), (\text{ТЕГ2}, A, 2), (\text{ТЕГ3}, A, 4),$

$(\text{ТЕГ1}, A, 4), (\text{ТЕГ2}, A, 4), (\text{ТЕГ3}, A, 5),$

$(\text{ТЕГ1}, A, 5), (\text{ТЕГ2}, B, 5), (\text{ТЕГ3}, A, 7),$

$(\text{ТЕГ1}, B, 6), (\text{ТЕГ2}, B, 7), (\text{ТЕГ3}, D, 13),$

$(\text{ТЕГ1}, B, 9), (\text{ТЕГ2}, B, 8), (\text{ТЕГ3}, D, 16),$

$(\text{ТЕГ1}, C, 13), (\text{ТЕГ2}, D, 14), (\text{ТЕГ3}, E, 4),$

$(\text{ТЕГ1}, C, 9), (\text{ТЕГ2}, D, 8), (\text{ТЕГ3}, E, 19),$

$(\text{ТЕГ1}, C, 19), (\text{ТЕГ2}, D, 14), (\text{ТЕГ3}, E, 21).$

Для оптимізації памяті було вирішено зберігати теги у наступному форматі (ідентифікатор тегу, знаходження, час входу, час виходу) де час входу - це час коли працівник увійшов на підприємство а час виходу відповідно вийшов. Наступні записи не мають дублікатів,

$(\text{ТЕГ1}, A, 3, 5), (\text{ТЕГ2}, A, 2, 4), (\text{ТЕГ3}, A, 4, 7),$

$(\text{ТЕГ1}, B, 6, 9), (\text{ТЕГ2}, B, 5, 8), (\text{ТЕГ3}, D, 13, 16),$

$(\text{ТЕГ1}, C, 13, 19), (\text{ТЕГ2}, D, 14, 18), (\text{ТЕГ3}, E, 17, 19),$

$(\text{ТЕГ1}, K, 21, 32), (\text{ТЕГ2}, K, 19, 21), (\text{ТЕГ3}, K, 23, 25).$

Для зручності і більшої читабильності можна трансформувати попередні коди в наступний формат, що відображає переміщення працівка по підприємстві,

$\text{ТЕГ1}: A[4,5] \rightarrow B[6, 11] \rightarrow C[14, 15],$

$\text{ТЕГ2}: A[7,11] \rightarrow B[12, 13] \rightarrow C[16, 18],$

$\text{ТЕГ3}: A[22,5] \rightarrow B[6, 11] \rightarrow C[17, 19],$

$\text{ТЕГ4}: A[3,5] \rightarrow B[7, 11] \rightarrow C[12, 15].$

Використання трасуючих кодів дозволяє отримати інформацію використовуючи два типи запитів, тобто запити відстеження та запити, орієнтовані на шлях. Запити відстеження використовуються для отримання історії тегу. Запити, орієнтовані на шлях, використовуються для вибору набору тегів і задоволення певної умови шляху. Записи трасування зберігають інформацію про шлях за допомогою двох чисел: номера кодування елемента та номера кодування

порядку. Записи трасування зберігають інформацію про шлях за допомогою двох чисел: номера кодування елемента (НКЕ) та порядковий номер кодування (НКП).

Припустимо що локації Q, W, E, R відповідають наступним простим числам 3, 5, 7, 11 відповідно. Також припустимо що для тега згенерувався настуаний трасуючий код TAG1: Q[3, 5] → W[6, 9] → E[13, 19]. Відповідно до функції декодування,

$$НКЕ = 3 \cdot 5 \cdot 7 = 105,$$
$$НКР \bmod 3 = 1,$$
$$НКР \bmod 5 = 2,$$
$$НКР \bmod 7 = 3,$$
$$НКР < 105,$$
$$НКР = 17.$$

Тепер щоб отримати порядковий номер локації, яку відвідував працівник потрібно використати наступну формулу:

$$N = \text{НКП} \bmod L,$$

де N порядковий номер, НКП – номер кодування порядку, L – код локації.

Якщо проаналізувати наступний вираз 17 mod 7 = 3, можна побачити, що локацію із кодом 7 (E) працівник відвідав третьою. І якщо обчислити відповідно 17mod 3, 5, 7 отримаємо коди локації Q, W, E відповідно. Всі представлені розрахунки базуються на *Китайській* теоремі про залишки. Яка говорить, що існує масив взаємно простих чисел і їхній добуток більший від числа X. Ця теорія допомагає вирішити наступну систему рівнянь:

$$X \bmod P_1 = A_1,$$
$$X \bmod P_2 = A_2,$$
$$X \bmod P_3 = A_3,$$
$$X \bmod P_4 = A_4,$$
$$\ldots\ldots\ldots$$
$$X \bmod P_n = A_n.$$

Сигнали, що представляють інтерес для RFID, як правило, мають цифрову модуляцію. Цифрово модульований сигнал (рисунок 3.2) — це потік різних

символів. Загальним рішенням проблеми живлення є кодування двійкових даних перед модуляцією. Один із підходів кодування RFID відомий як імпульсно-інтервальне кодування. Двійковий «1» кодується як короткий імпульс вимкнення після тривалого інтервалу повної потужності, а двійковий «0» кодується як коротший інтервал повної потужності з тим самим імпульсом вимкнення живлення.

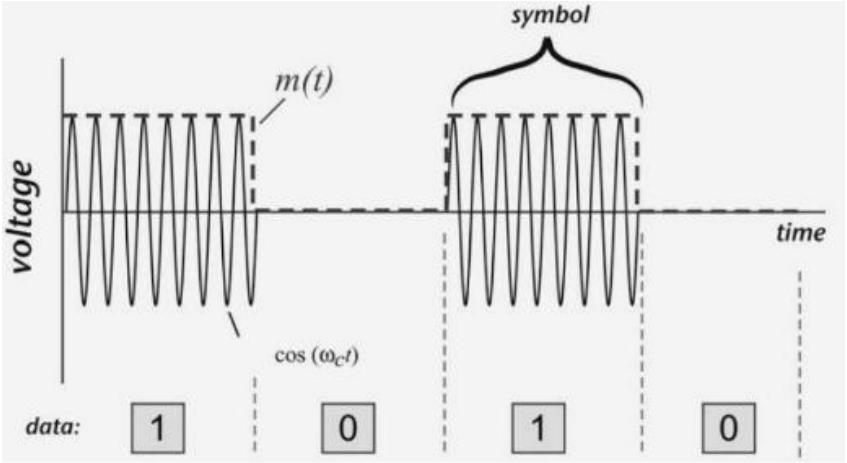

Рисунок 3.2 – Модуляція силналу

У пасивній системі RFID передавач не вимикається, а замість цього передає сигнал протягом часу, коли приймач прослуховує сигнал мітки. Радіо RFID використовує спеціалізовані компоненти, відомі як циркулятори або з'єднувачі, щоб пропускати до приймача лише відбиті сигнали, які можуть бути насиченими потужним переданим сигналом.

3.2 Моделювання системи радіочастотної ідентифікації

В цьому розідлі змодельвовано ланцюг бездротового зв'язку (рисунок 3.3) системи радіочастотної ідентифікації (RFID) із фазовою маніпуляцією у Matlab. Для виконання моделювання будуть використані наступні інструменти - Matlab R2015a.

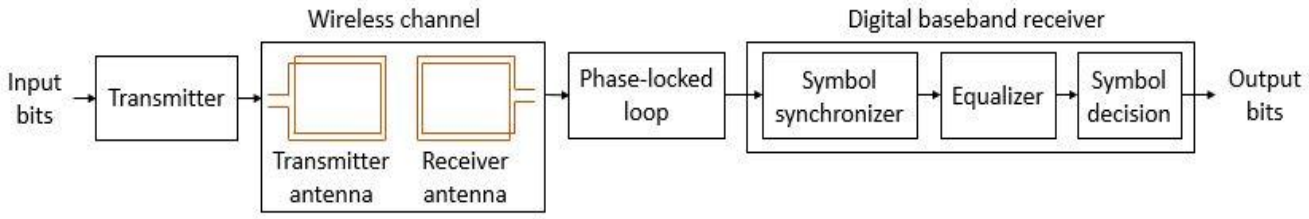

Рисунок 3.3 – Ланцюг бездротового зв’язку

Ланцюг бездротового зв’язку, який моделюється в цьому проекті, включає:

- передавач;
- бездротовий канал і модель антени;
- фазовий автопідстроювання частоти;
- цифровий приймач базової смуги, що складається з блоків синхронізації символів, еквалайзера та блоків визначення символів.

Передавач моделює `M` сигнал PSK на несучій частоті `fc` та із швидкостю передачі даних `fc/4 [36]`. Файл `constellation.m` визначає карту сузір'я PSK на комплексній площині та бітове кодування символу (рисунок 3.4). Диференціальне кодування використовується для протидії ковзанням циклу.

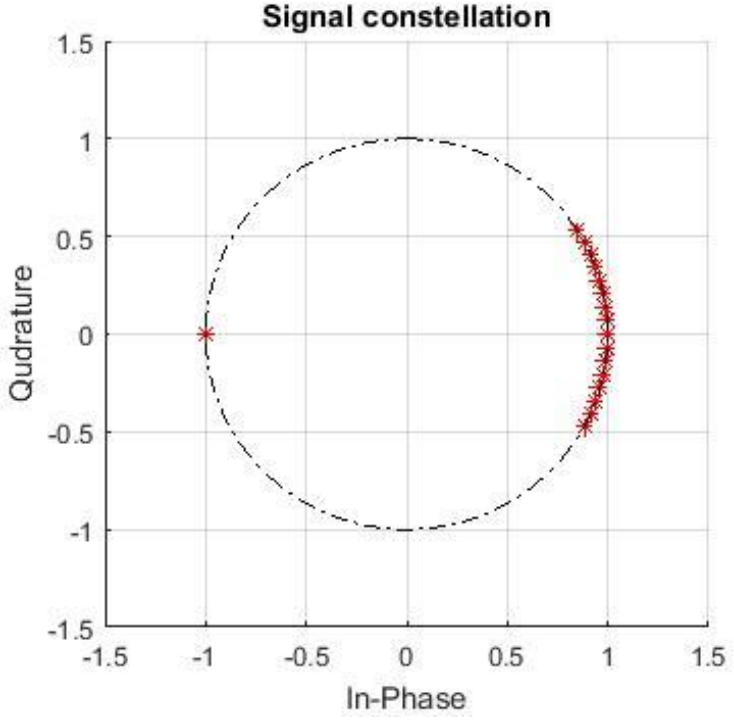

Рисунок 3.4 – Сигнал передавача

Передавач формує сигнал виду:

- початковий період мовчання (не модульований синусоїдальний сигнал);
- преамбула або заголовок (позначає початок пакета даних);
- N диференціально закодовані символи (корисні дані);
- послідовність символів кінця пакета (що означає кінець пакета даних);
- остаточний період тиші (не модульований синусоїдальний сигнал).

Параметри моделювання за замовчуванням для передавача представлені на рисунку 3.5. Потрібно зауважити , що симуляція включає швидкісну вибірку. Передавач і бездротовий канал імітуються в 16 разів швидше `TsSim` порівняно з цифровим приймачем основної смуги частот, який імітується при `Ts`.

```
N = 800;
M = 16;
fc = 13.56e6;
dataRate = fc/4;
nSampSim = 32;
nSamp = 2;
TsSim = 1/(nSampSim*dataRate);
Ts = 1/(nSamp*dataRate);
```

Рисунок 3.5 – Параметри передавача

Бездротовий канал складається з антени передавача, середовища бездротової передачі повітря та антени приймача. Антенна схема передавача та приймача моделюється як послідовна та паралельна схема RLC відповідно.

Для всіх практичних цілей, враховуючи, що відстань між передавачем і приймачем зазвичай становить менше 10 см у застосуванні RFID, ми можемо змоделювати бездротовий канал як взаємно пов’язану пару котушок антени передавача та приймача. Це призводить до еквівалентної схеми, де котушки індуктивності з’єднані взаємною індуктивністю,

$$M = K\sqrt{Л_t \cdot Л_к}$$

де k це коефіцієнт зв’язку, який визначає близькість первинної та вторинної котушок із значеннями, як правило, між 0,03 і 0,3. Значення k є точною одиницею лише тоді, коли дві котушки об’єднані в одну котушку (рисунок 3.6).

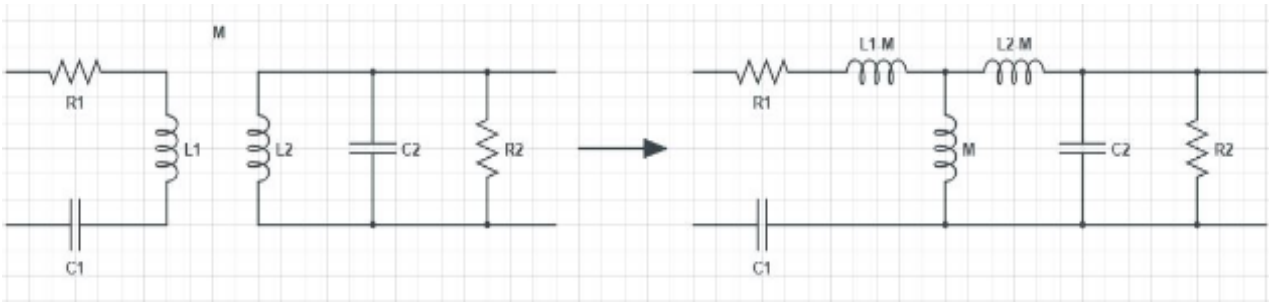

Рисунок 3.6 – Модель антени

Функція передачі аналогового смугового каналу потім відображається на цифровий еквівалент низьких частот (рисунок 3.7).

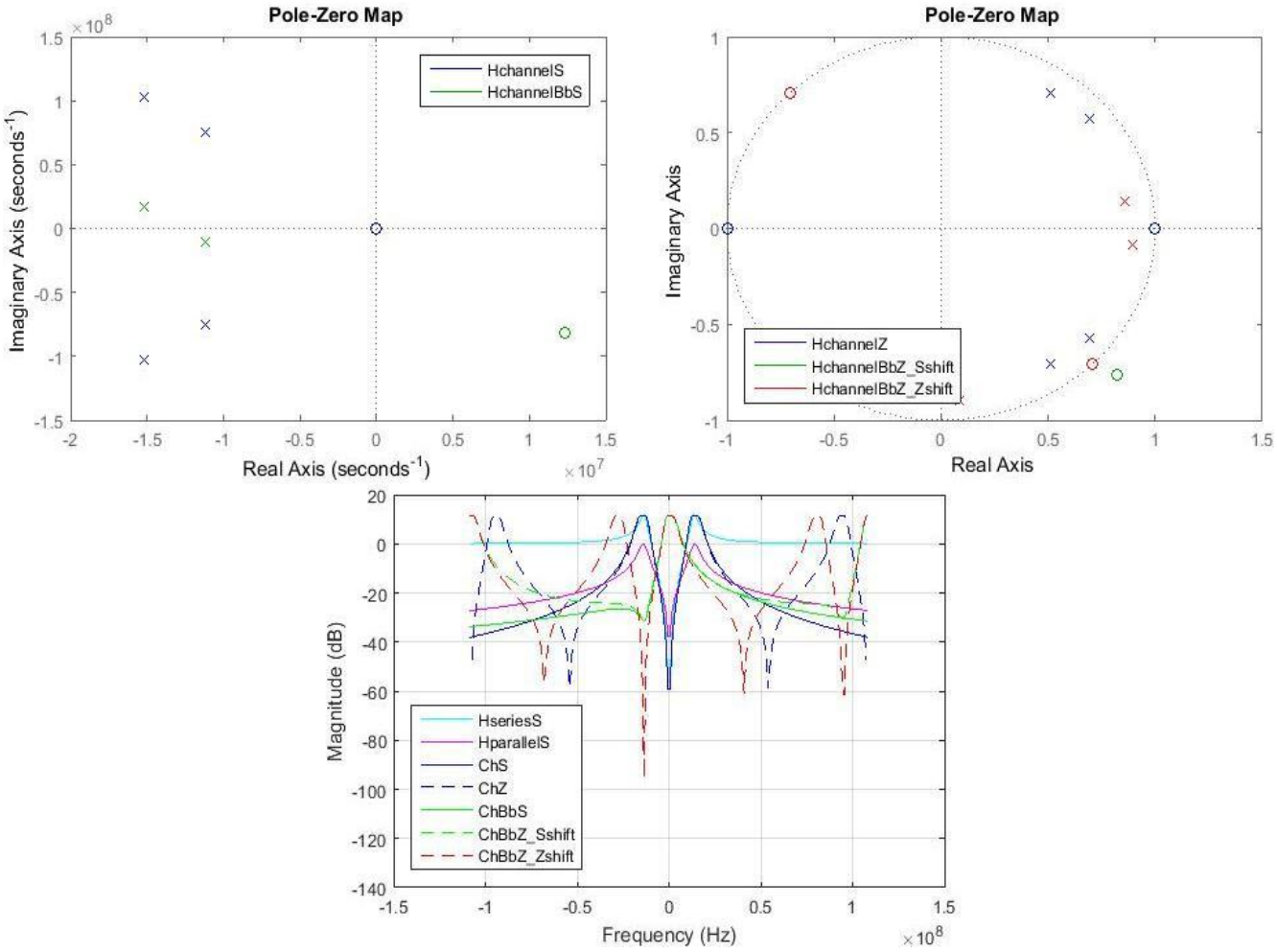

Рисунок 3.7 – Відображення передаточної функції

Загальна передаточна функція комбінованого резонансного каналу задана як

$$\frac{C_1 M P_2 c^2}{\left[\begin{array}{c} P_2 + Л_2 s \; - C_1 M^2 c^3 + C_1 Л_1 Л_2 c^3 + C_1 Л_1 P_2 c^2 + C_1 Л_2 P_1 c^2 + C_2 Л_2 P_2 c^2 + \\ C_1 P_1 P_2 s \; - C_1 C_2 M^2 P_2 c^4 + C_1 C_2 Л_1 Л_2 P_2 c^4 + C_1 C_2 Л_2 P_1 P_2 c^3 \end{array}\right]}.$$

Форма хвилі отриманої модульованої смуги пропускання на приймачі після поширення через передавач і бездротовий канал зображена на рисунку 3.8.

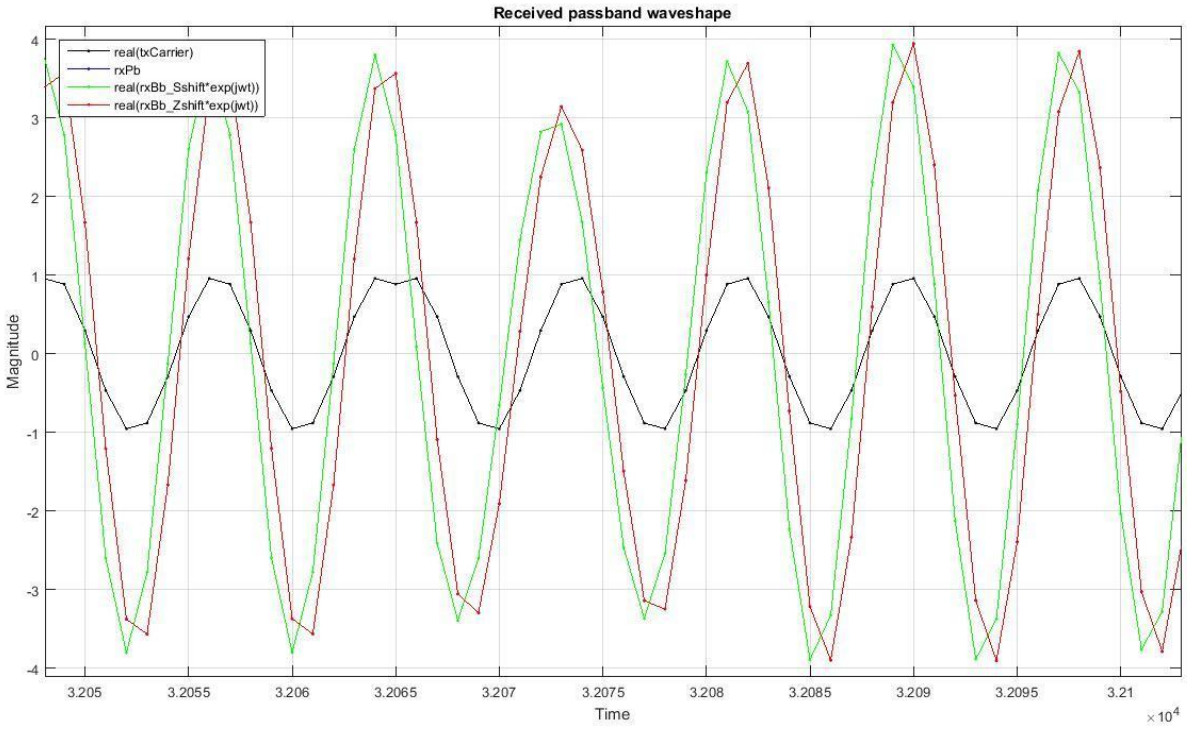

Рисунок 3.8 – Форма сигналу в передавачі

Параметри моделювання за замовчуванням для моделі бездротового каналу та антени (рисунок 3.9).

```
k = 0.3;
Qr = 4;
Qc = 3;
fresr = 13.56e6;
fresc = 14e6;
```

Рисунок 3.9 – Параметри для антени

Вихідний сигнал подається в цифровий приймач основної смуги частот. По-перше, для виявлення початку пакета даних використовується синхронізатор символів у формі псевдо узгодженого фільтра. Як тільки пакет виявлено, отриманий сигнал поширюється через еквалайзер, за яким слідує блок рішення про символ. Рішення приймаються з використанням метрики мінімальної евклідової відстані. Еквалайзер налаштований у конфігурації з дробовим інтервалом із зворотним зв'язком рішення, оскільки частково розподілений еквалайзер забезпечує оптимальну узгоджену фільтрацію та синхронізацію часу символа. Еквалайзер із дробовим інтервалом видає єдине рішення для кожного символу, відібраного зі швидкістю, вищою за швидкість символу.

Параметри моделювання за замовчуванням для цифрової моделі приймача основної смуги частот (рисунок 3.10).

```
kFwd_pos = 0:-1:-2;
kBwd = 2;
mu = 0.25;
L = 2;
gamma = 1;
bias = 0;
delay = 1;
fracEn = 1;
```

Рисунок 3.10 – Параметри приймача

Алгоритм проекції використовується як оптимізатор для оновлення вагових векторів. Алгоритм повторного використання даних, обраний через його переваги:

- він має складність і швидкість збіжності між алгоритмом найменшого середнього квадрата і рекурсивного алгоритму найменших квадратів;
- він надає можливість контролювати швидкість адаптації, величину та неправильне налаштування коефіцієнтів.

ВИСНОВКИ

У представлені кваліфікаційній роботі була розроблена комп'ютерно-інтегрована система обліку робочих годин працівників. У ході роботи була розроблена серверна частина та функціональна схема яка досить чітко описує темінал. Досліджено та проаналізовано існуючі системи обліку робочих годин та проведено їхню класифікацію, що в свою чергу дає можливість вибрати оптимальну систему для проекту. Обруновано вибір компонентів серверної частини проекту а саме: мови програмування, веб-фреймворку, бази даних. Коротко обґрунтовано вибір апаратної частини терміналу та детально описано їхні характеристики. Детально описано процес хостингу за допомогою AWS. При виборі обґрунтовано використання технічних засобів, безпосередньо для даного об’єкту управління, з урахуванням контрольованих параметрів відповідність габаритів та потужності, умов навколишнього середовища, шум, вібрацію та споживання енергії засобів контролю швидкості. Розроблено метод обробки та зберігання даних, що згенерові RFID тегом. Дагий метод дозволяний знизити кількість необхідної памяті, оскільи необхідно зберігати тільки два числа.

Проведений аналіз технологічного процесу, як об’єкта керування та виявлення недоліків існуючих інформаційних алгоритмів та технічних засобів, що використовуються при функціонуванні обладнання в подібних системах обліку.

Змодельовано за допомогою математичного комплексу програмного забезпечення метод обробки сигналів для RFID, що дало можливість більш надійно та ефективніше відобразити результати високошвидкісної обробки сигналів.

Дана система забезпечує надійну, безперебійну та якісну роботу в процесі обліку, також оперативне реагування та само балансування на нестандартних ситуаціях роботи. Це в свою чергу дозволяє мінімізувати витрати фізичної праці, що значно підвищує надійність роботи системи. Також ця система дозволяє мінімілізувати навантаження на мереж, що значно підвищує надійність роботи подібної систем та мережі в цілому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М. Г. Попович, О. В. Ковальчук. Теорія автоматичного керування: Підручник. — 2-ге вид. — К.: Либідь, 2007. — 656 с.
2. Іванов А. О. Теорія автоматичного керування: Підручник. — Дніпропетровськ: Національний гірничий університет. — 2003. — 250 с.
3. Louis C. Westphal. Handbook of Control Systems Engineering. — 2nd edition; The Springer International Series in Engineering and Computer Science. —Springer, 2001. — Т. 635. — 1063 с. — ISBN 978-0792374947. англ.)
4. Бесекерский В. А., Попов Е. П. Теория Систем Автоматического Управления. 2003.
5. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
6. Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки, 2014. - 280 с.
7. Блинов И.Н., Романчик В. С. Java. Методы программирования : уч.-мет. пособие / И. Н. Блинов, В. С. Романчик. Минск : издательство "Четыре четверти", 2013. 896 с.
8. Блинов, И.Н. Java 2: практ. рук. / И.Н. Блинов, В.С. Романчик. - Мн.: УниверсалПресс, 2005. - 400 с.
9. Блинов, И.Н. Java. Промышленное программирование : практ. пособие / И.Н. Блинов, В.С. Романчик. - Минск : Универсал Пресс, 2007. - 704 с.
10. Васильев А. Н. Java. Объектно-ориентированное программирование: Учебное пособие. СПб.: Питер, 2011. 400 с.
11. Гамма, Э., Хелм, Р., Джонсон, Р., Влиссидес, Дж. Приемы объектноориентированного проектирования. Паттерны проектирования. СПб. : Питер, 2007. 366 с.
12. Глоба Л. С. Розробка інформаційних ресурсів та систем [Електронний ресурс] : конспект лекцій / Л. С. Глоба, Т. М. Кот. - Київ : НТУУ "КПІ", 2014. - 318 с.
13. Грязнова В. О., Єфіменко С. В. Основи методології програмування. - К.: ВПЦ "Київський університет", 2010.
14. Давыдов В.Г. Программирование и основы алгоритмизации: Учеб. пособие. / В.Г. Давыдов. М.: Высш. шк., 2003. 447 с.
15. Інженерія якості програмного забезпечення: навч. посібник / Г.В Табунщик, Р.К. Кудерметов, Т.І. Брагіна. - Запоріжжя: ЗНТУ, 2013. - 180 с.
16. Кингсли-Хьюджес Э., Кингсли-Хьюджес К. Справочник программиста. - М.: ООО "ИД Вильямс", 2007.
17. Ларман, К. Применение UML 2.0 и шаблонов пректирования. Введение в объектно-ориентированный анализ и проектирование. 3-е изд. СПб. : Вильямс, 2012. 736 с.
18. Перри, Б. У. Java сервлеты и JSP. Сборник рецептов. М. : Кудиц-пресс, 2009. 768 с.

19. Рудаков А. В. Технология разработки программных продуктов : учебник для студ. сред. проф. образования / А. В. Рудаков. 7-е изд., стер. М. : Издательский центр "Академия", 2012. - 208 с.
20. Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. - Харків : ХНУМГ ім. О. М. Бекетова, 2017. - 93 с.
21. Хорстманн, К. С., Корнелл, Г. Библиотека профессионала. Java 2 : Том 1. Основы. 8-е изд. М. : Вильямс, 2013. 816 с.
22. Шевчук І. Б. Інформаційні технології в регіональній економіці: теорія і практика впровадження та використання : монографія. Львів : Видавництво ННВК "АТБ", 2018. 448 с.
23. Шилдт Г. Полный справочник по Java SE 6. М.: Вильямс, 2010. 1040 с.
24. Головний сайт платформи Arduino, електронний ресурс. Режим доступу: http://arduino.ua
25. Головний сайт модуля розширення Cnc shield v3, електронний ресурс. Режим доступу: http://blog.protoneer.co.nz/arduino-cnc-shield/
26. Стаття, присвячена мові програмування g-code, електронний ресурс. Режим доступу: https://ru.wikipedia.org/wiki/G-code
27. Головна сторінка платформи GRBL, електронний ресурс. Режим доступу: https://github.com/grbl/grbl
28. “Системи керування електроприводів” - В.М. Тєрехова та О.І. Осіпова. Друге видання, видавництво Academia, Москва 2006.
29. Міжнародний науковий журнал “Молодой ученый”, випуск 15 за 2016 рік. Видавництво «Издательство Молодой ученый».
30. “Центральный металлический портал РФ”, електронний ресурс. Режим доступу: http://metallicheckiyportal=frezernix_stankov
31. Головна сторінка компанії ТехноАрсенал, електронний ресурс. Режим доступу: ttp://www.technoarsenal.ua/gravirovalnaya_mashina_gm.html
32. Довідник за типами файлів, електронний ресурс. Режим доступу: http:// open-file.ru/types/nc
33. Довідникова сторінка онлайн-програми MakerCam, електронний ресурс. Режим доступу: http://www.makercam.com/help.html
34. Головна сторінка платформи UGS, електронний ресурс. Режим доступу: https://winder.github.io/ugs_website/
35. Стаття, присвячена серії крокових двигунів Nema, електронний ресурс. Режим доступу: http://reprap.org/wiki/NEMA_17_Stepper_motor
36. Стаття про фрезерні верстати з ЧПК, електронний ресурс. Режим доступу: https://en.wikipedia.org/wiki/CNC_router