НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського
Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Розробка клієнтської частини інформаційних систем з застосуванням мікросервісної архітектури з на мікроконтроллерах

Виконав: студент __4__ курсу, групи __ДА-31____
(шифр групи)

Штанько Денис Валерійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц., к. т. н., Харченко К. В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний доц., к. е. н., Рощина Н. В._____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент ___
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Штанько Денис Валерійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка клієнтської частини інформаційних систем з застосуванням мікросервісної архітектури на мікроконтроллерах ,

керівник роботи Харченко Костянтин Васильович, к. т. н., доцент _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10»_травня___ 2017 р. №1477__

2. Термін подання студентом роботи 13.06.2017 _____

3. Вихідні дані до роботи Мікроконтроллер Raspberry PI 3, зчитувач NFC карток RFID RC522, Docker container, тестова NFC картка, MySQL база даних, SQLite база даних.

4. Зміст роботи

1. Аналіз існуючих рішень.

2. Зібрка та налаштування клієнтського пристрою для фіксації присутності магнітного ключа.

3.Реалізація мікросервісів обліку та буферизації даних на пристрої.

4. Функціонально вартісний аналіз проекту

5.Проаналізувати результати роботи, зробити висновки.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

- DFD діаграмма системи - плакат;
- Схема з'єднання пристрою - плакат;
- Use-Case діаграмма системи - плакат;
- Презентація.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доц., к. е. н., Рощина Н. В.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з технічною літературою і підготовка теоретичної частини роботи	17.04.2017 – 21.04.2017	
2	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	22.04.2017	
3	Проектування архітектури, складання базової документації	30.04.2017	
4	Перевірка розроблених прототипів. Перевірка відповідності завданню.	05.05.2017	
5	Підготовка графічного матеріалу, оформлення пояснювальної записки, підготовка до захисту	15.05.2017	
6	Проходження нормоконтролю, отримання рецензії, подача роботи в ДЕК	10.06.2017	
7	Захист дипломної роботи	19.06.2017	

Студент _____ (підпис) _____ (ініціали, прізвище)

Керівник роботи _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АННОТАЦІЯ

бакалаврської дипломної роботи Штанька Дениса Валерійовича на тему : «Розробка клієнтської частини інформаційних систем з застосуванням мікросервісної архітектури на мікроконтроллерах»

В даній дипломній роботі розглянуто процесс практичного примінення мікросервісного стилю програмування для реалізації клієнтської частини інформаційної системи, що має стати замінником для традиційних способів обліку відвідувань студентами учбових зайнять.

Основною увага приділена розробці головному елементу клієнтської частини – пристрою зчитування магнітних карток, що дозволить швидко фіксувати присутність власника картки на зайнятті. Розглянуто основні проблеми які вирішують мікросервіси в такому контексті, та можливості які вони відкривають для розробника.

В висновках до дипломного проекту приведено спостереження що до перспектив використання мікросервісної архітектури, а також можливі переваги такого підходу.

Роботу рекомендовано до використання в якості прикладного рішення організаційних задач в структурах з численною кількістю користувачів, та необхідністю обліку їх присутності в регламентовані проміжки часу.

Загальний обсяг роботи : 55 сторінок, 2 додатки на 23 сторінках, 23 рисунків, 6 таблиць, 11 посилань.

Ключові слова : NFC, Raspberry Pi, мікроконтроллер, мікросервіси, контейнеризація, Docker.

АННОТАЦИЯ

бакалаврской дипломной работы Штанько Дениса Валерьевича на тему: «Разработка клиентской части информационных систем с применением микросервиснои архитектуры на микроконтроллерах»

В данной дипломной работе рассмотрены процесс практического приминення микросервисного стиля программирования для реализации клиентской части информационной системы, которая должна стать заменителем для традиционных способов учета посещений студентами учебных занятий.

Основное внимание уделено разработке главного элемента клиентской части - устройства считывания магнитных карт, позволяющего быстро фиксировать присутствие владельца карточки на занятии. Рассмотрены основные проблемы, которые решают микросервисы в таком контексте, и возможности которые они открывают для разработчика.

В выводах к дипломному проекту приведены наблюдения относительно перспектив использования микросервиснои архитектуры, а также возможные преимущества такого подхода.

Работа рекомендуется к использованию в качестве прикладного решения организационных задач в структурах с большим количеством пользователей, и необходимостью учета их присутствия в регламентированых промежутках времени.

Общий объем работы: 55 страниц, 2 приложения на 23 страниц, 23 рисунков, 6 таблиц, 11 ссылок.

Ключевые слова : NFC, Raspberry Pi, микроконтроллеры, микросервисы, контейнеризация, Docker.

ANNOTATION

to Denis Valeryevich Shtanko’s Bachelor's degree thesis on the topic:
"Development of client side microservices using microcontrollers"

In this thesis is described a process of practical application of the microservice programming style for the implementation of the client part of information system, which should become a substitute for traditional ways of recording student attendence to study sessions.

The main attention is paid to the development of the main element of the client part - the magnetic card reader, which allows you to quickly record the presence of the cardholder in the class. The main problems that microservices solve in this context are considered, and the opportunities that they open for the developer.

In the conclusions to the project, observations are made on the prospects of using micro-service architecture, as well as the possible advantages of such approach.

Work is recommended to be used as an applied solution of organizational tasks in structures with a large number of users, and with need to account their presence in the regulated intervals of time.

The total amount of work: 55 pages, 2 appendecies for 23 pages, 23 pictures, 6 tables, 11 references.

Keywords: NFC, Raspberry Pi, mictocontrollers, micrpservices, containertization, Docker.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP - HyperText Transfer Protocol

HTML - HyperText Markup Language

CI - Continuous Integration

XML - eXtensible Markup Language

SOA - Service Oriented Protocol

IP - Internet Protocol

URL - Uniform Resource Locator

ЗМІСТ

ВСТУП

Бакалаврська робота присвячена розробці периферійних елементів інформаційної системи, що повинна забезпечувати спрощений облік відвідування лекцій студентами. Інформаційна система скаладається з переносного пристрою, серверної частини, та хмарного хранилища реалізованого на сервісах Google.

В порядку наукової цінності інформаційна система, за виключення сервісів, що покривають функціонал хмарного хранилища, реалізується шляхом втілення мікросервісної архітектури. В порядку дослідження необхідні до розгляду аспекти мікросервісної архітектури включають : можливість налаштування проекту прямими маніпуляціями над процесами, на рівні файлової системи, спрощена імплементація та заміна функціональних блоків інформаційної системи.

В результаті розробки очікується реалізація приладу, та підтримуючої його системи, який дозволяє фіксувати присутність в полі зчитування магнітної картки стандарту MILFARE 10.1 – 13 MHZ. При наявності в полі досяжності пристрою визнаних технологією NFC приймача RC520 магнітних карток.

Розробка отримана за результатом дипломного проекту рекомендується до втілення в учбових закладах для економії кошт та мінімізації втрат часу на формування електронної звітності. За рахунок гнучкості системи, можливе розширення функціоналу інформаційної системи, щоб покрити такі категорії послуг, як голосування та захищена авторизація користувачів. За рахунок довільного зв’язку з частиною, що інтерпретує накопичену інформацію, дані зручні та універсальні для безпосереднього перевикористання.

Можливими шляхами інтеграції запропонованого пристрою можуть стати мережі та сервіси університетської самоорганізації, як CAMPUS. Відповідно до поставленої задачі, економічна вартість одного пристрою може бути оцінена в еквівалентну вартість 17 робочих днів викладача, які, згідно з апроксимованими обрахунками, викладач втрачає на організацію електронної звітності щороку.

1 СИСТЕМНИЙ АНАЛІЗ ЗАВДАНЬ БАКАЛАВРСЬКИХ ДОСЛІДЖЕНЬ

Мета дипломної роботи

Розробка периферійних функціональних елементів інформаційної системи обліку відвідувань, на базі мікросервісної архітектури. Дослідження умов повної контейнеризації мікросервісів інформаційної системи розробленої на базі сервіс-орієнтованої архітектури. Практичний аналіз економічної виправданості такої системи в порівнянні з ринковими аналогами.

Огляд літератури

Мікросервісний підхід до пороектування додатків є порівнянно новою ідеологія, бодай і бере початки свого примінення практично з 90х років. На даний момент найбільш ефективними джерелами для рзгляду примінення новітніх для массового примінення стилів програмування стають онлайн статті та інтернет видавництва, що ведуться на некомерційній освнові, адже є порівнянно ризиковими для надійно фінансованого видавництва. Такі теми потребують швидкого тестування, до доцільного обговорення спеціалістами, що мали шанс примінити концепцію в якомога більш різноманітних варіантах її використання. Тільки таким чином можна зібрати доцільну інформацію відносно, ще не затвердившої себе на ринку парадигми архітектурної побудови інформаційних систем.

Окремо від критики та огляду примінення мікросервісного стилю в роботі використані матеріали технічної документації таких продуктів та сервісів як Docker Container, Google Sheets (відповідно також Google Sheets API),

Задачі дипломної роботи

- Прототип пристрою-зчитувача магнітних карток для викладача.
- Дослідити аспекти архітектури мікроконтроллеру Raspberry Pi 3, з операційною системою Raspbian, та реалізувати з’єднання мікроконтроллера та модуля зчитування NFC магнітного сигналу.
- Розглянути альтернативи, та економічну виправданість інформаційної системи.
- Дослідити можливу декомпозицію інформаційної системи на мікросервіси, та визначити можливі та найзручніші варіанти їх комунікації.
- Створити необхідні для обліку студентів та розкладу викладачів реквізити на хмарному хранилищі Google, та створити умови захищеного до них доступу.
- Перевірити стійкість спроектованої системи до умов нестабільної мережі.
- Проаналізувати реалізацію спроектованої архітектури в умовах повної контейнеризації, за допомогою сервісу Docker Container.

1.1 Цілі дипломної роботи, ключові фактори успіху результатів роботи

Основною метою дипломної роботи є проектування і розробка приладу зчитування магнітних карток на базі мікроконтроллера, що дозволяє встановлення Unix-подібної операційної системи, а також створення веб інтерфейсу взаємодії з клієнтським контентом системи обліку відвідувань лекцій студентами.

Для реалізації обрано мікроконтроллер Raspberry Pi 3, з розширенням для зчитування NFC ключів – RC520. В інтересах бюджетності витрат, для прототипу приладу не пропонується автономного джерела живлення, проте обов’язковим є можливість бездротового під’єднання до інтернету.

Ключові фактори успіху дипломного проекту :

- спроектоване та реалізоване програмне забезпечення для зчитування приладом магнітних карток
- дослідження можливостей та переваг мікросервісного стилю проектування в рамках довгострокового проекту.
- реалізація альтернативних сегментів мікросервісної структури проекту для перевірки гнучкості системи, та можливостей її налаштування на рівні файлової системи.
- на основі результатів досліджень сформувати висновок про можливості та умови подальшого розширення та розробки проекту.
- сформувати твердження про розвиток і застосування розробки
- розглянути економічні застави широкомасштабного примінення розробки в рамках університетських структур та запропонувати можливі зміни до проекту, регулюючись можливими необхідностями та рамками примінення.

ФУНКЦІОНАЛЬНА МОДЕЛЬ

1.2.1 DFD (IDEF0) діаграма

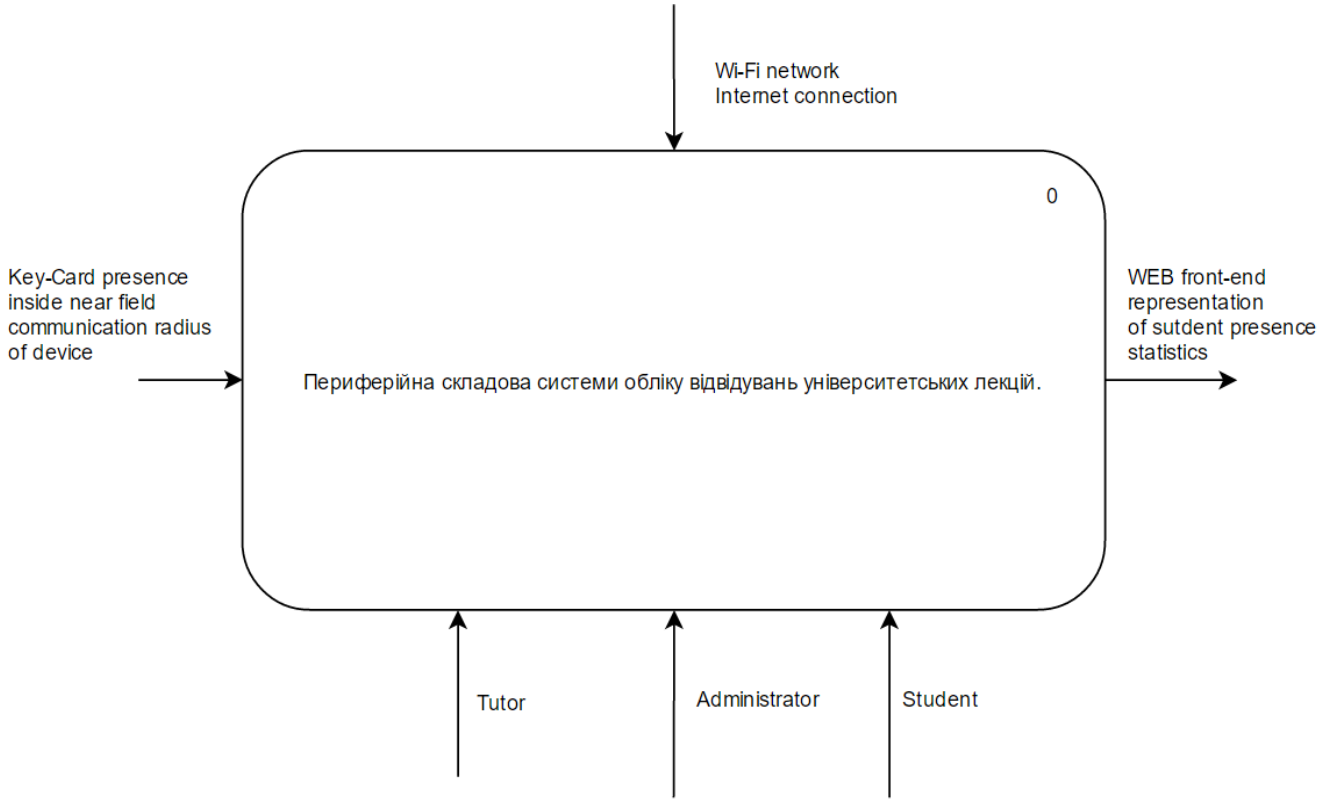

Рисунок 1.1 – Загальна DFD діаграма.

1.2.2 DataFlow Діаргамма другого рівня.

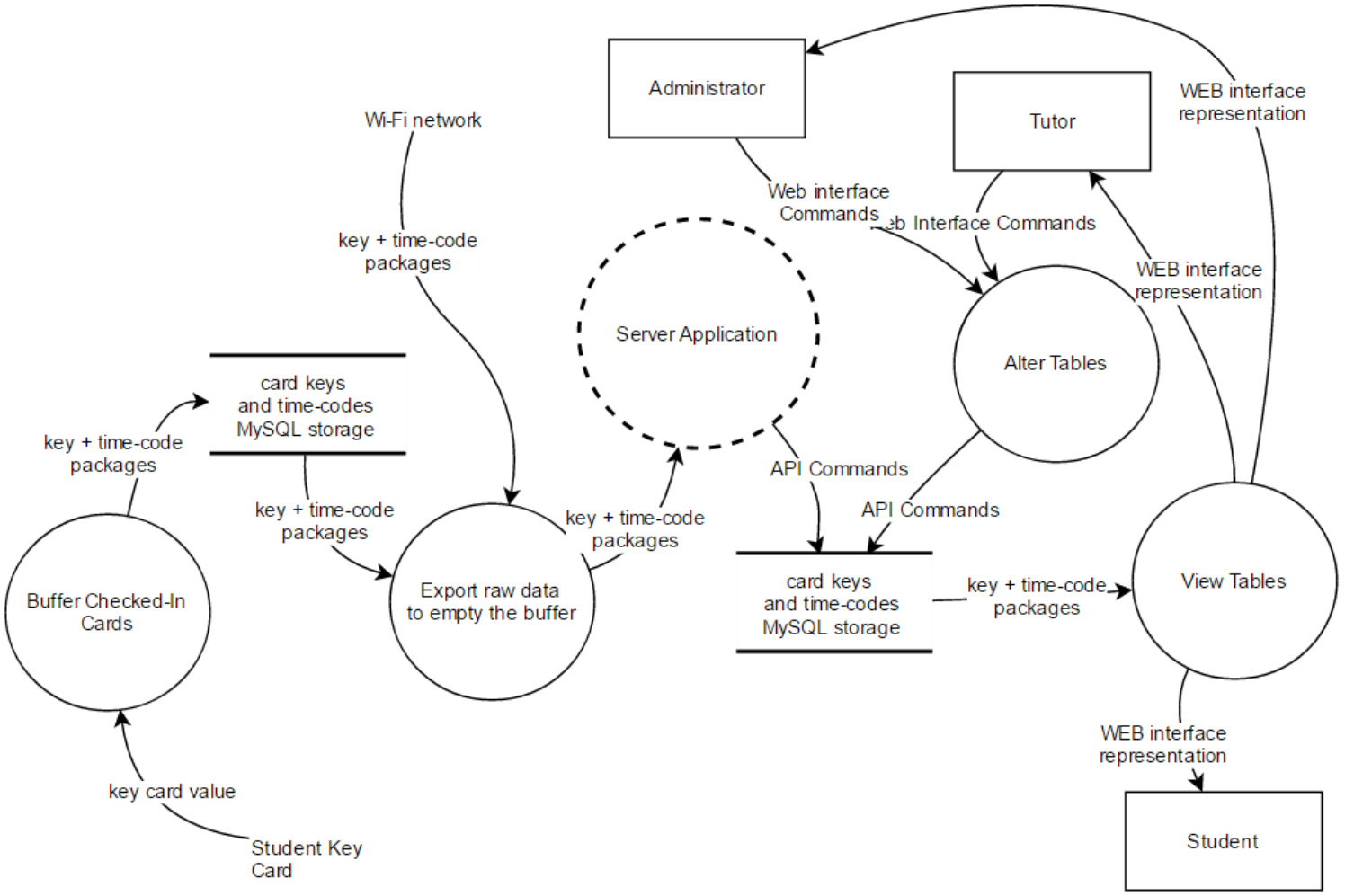

Рисунок 1.2 – Функціональна діграма другого рівня. Data Flow

1.3 Висновки до розділу 1

В даному розділі було розглянуто цілі дипломної роботи та перераховано принципи, які сприятимуть успішній реалізації цього проекту. Завдяки цим принципам ми зможемо побудувати надійну та стабільну систему та в подальшому з легкістю керувати нею.

2. ПОБУДОВА ПРОЕКТУ З ЗАСТОСУВАННЯМ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

2.1 Огляд мікросервісного стилю проектування та його особливастей

2.1.1 Мікросервіси як альтернатива.

Мікросервіси пропонуються на противагу монолітному стилю програмування (monolithic style): додаткам, побудованим, як єдине ціле. Монолітний сервер - досить очевидний спосіб побудови подібних систем. Вся логіка по обробці запитів виконується в єдиному процесі, при цьому ви можете скористатися наявними можливостями вашого мови програмування для поділу додатки на класи, функції і namespace-и. Ви можете запускати і тестувати додаток на машині розробника і використовувати стандартний процес розгортання для перевірки змін перед публікацією їх в продакшн. Ви можете масштабувати монолітне додатки горизонтально шляхом запуску декількох фізичних серверів за балансувальник навантаження.

Монолітні додатки можуть бути успішними, але все більше людей розчаровуються в них, особливо в світлі того, що все більше додатків розгортаються в хмарі. Будь-які зміни, навіть самі невеликі, вимагають перезбірки і розгортання всього моноліту. З плином часу, стає важче зберігати хорошу модульну структуру, зміни логіки одного модуля мають тенденцію впливати на код інших модулів. Масштабувати доводиться все додаток цілком, навіть якщо це потрібно тільки для одного модуля цього додатка.

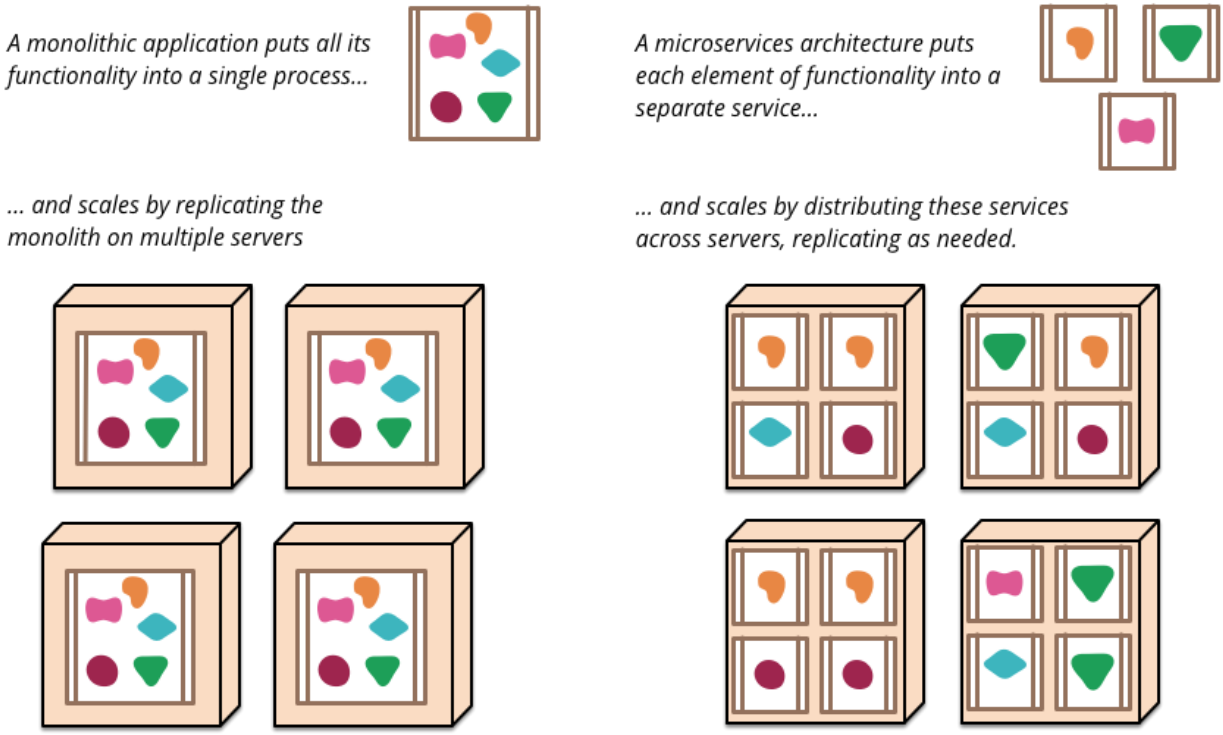

Рисунок 2.1 – Іллюстрація різниці монолтної архітектури та мікросервісної.

Ці незручності призвели до архітектурного стилю мікросервісів: побудови додатків у вигляді набору сервісів. Додатково до можливості незалежного розгортання і масштабування кожен сервіс також отримує чітку фізичну межу, яка дозволяє різним сервісам бути написаними на різних мовах програмування. Вони також можуть розроблятися різними командами.

Важливо сказати, що тут нема ствердження, що стиль мікросервісів це інновація. Його коріння сягає далеко в минуле, як мінімум до принципів проектування, використаним в Unix. Але ми все ж вважаємо, що недостатньо людей беруть до уваги цей стиль і що багато програм отримають переваги якщо почнуть застосовувати цей стиль.

2.1.2 Організація архітектури довкола потреб інформаційної системи.

Коли великі додатки розбиваються на частини, часто менеджмент процессу фокусується на технологіях, що призводить до утворення UI команди, серверної команди і БД команди. Коли команди розбиті подібним чином, навіть невеликі зміни забирають багато часу через необхідність крос-командної взаємодії. Це призводить до того, що команди розміщують будь-яку логіку на тих шарах, до яких мають доступ. Закон Конвея (Conway's Law) в дії.

«Будь-яка організація, яка проектує якусь систему (в широкому сенсі) отримає дизайн, чия структура копіює структуру команд в цій організація» - Melvyn Conway, 1967

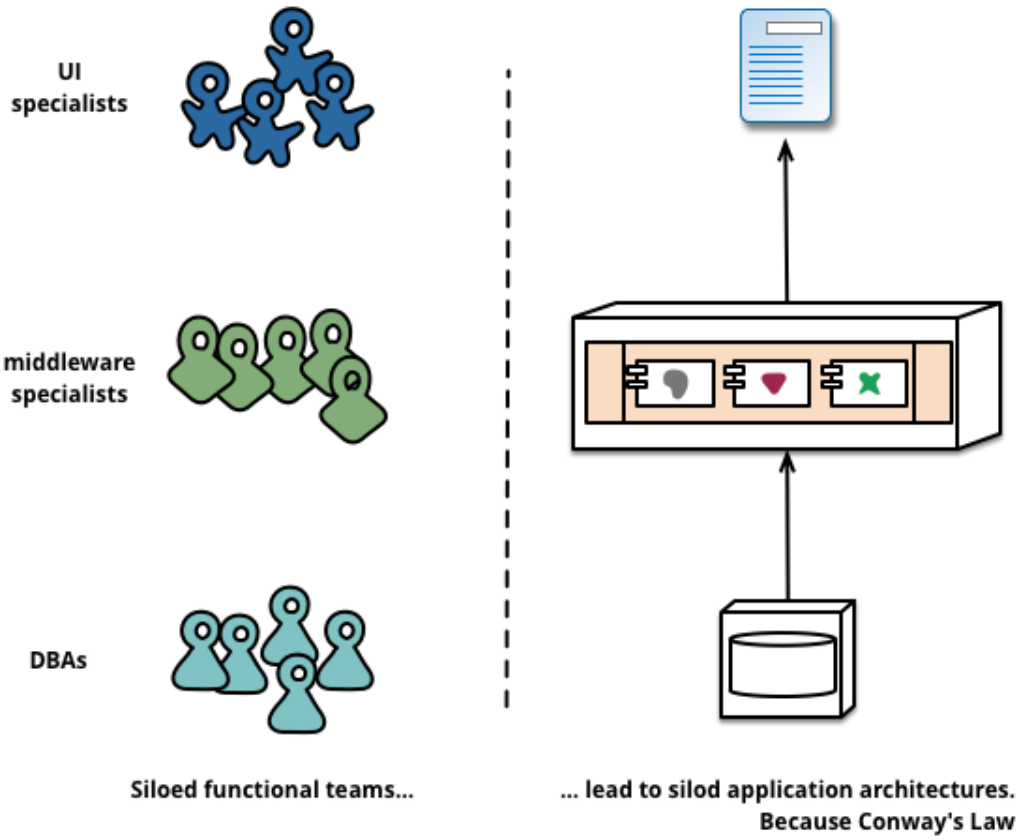

Рисунок 2.2 – закон Конуея в дії.

Ті, хто практикує мікросервісную архітектуру, зазвичай багато працювали з еволюційним дизайном і розглядають декомпозицію сервісів

як подальшу можливість дати розробникам контроль над змінами (рефакторингом) їх застосування без уповільнення самого процесу розробки. Контроль над змінами не обов'язково означає зменшення змін: з правильним підходом і набором інструментів можливо робити часті, швидкі, добре контрольовані зміни.

Кожен раз при намаганні розбити додаток на компоненти, є ризик стикнутися з необхідністю прийняти рішення, як саме ділити додаток. Чи є якісь принципи, які вказують, як найкращим способом «нарізати» додаток? Ключова властивість компонента - це незалежність його заміни або поновлення, що має на увазі наявність ситуацій коли його можна переписати з нуля без порушення взаємодіючих з ним компонентів. Багато команд розробників йдуть ще далі: вони явно планують, що безліч сервісів в довгостроковій перспективі не буде еволюціонувати, а будуть просто викинуті на смітник.

Веб-сайт Guardian - хороший приклад програми, яка була спроектована і побудована як моноліт, але потім еволюціонувало в бік мікросервісів. Ядро сайту все ще залишається монолітом, але нові елементи додаються шляхом побудови мікросервісов, які використовують API моноліту. Такий підхід особливо корисний для функціональності, яка по суті своїй є тимчасовою. Приклад такої функціональності - спеціалізовані сторінки для освітлення спортивних подій. Такі частини сайту можуть бути швидко зібрані разом з використанням швидких мов програмування і видалені як тільки подія закінчиться. Ми бачили схожий підхід в фінансових системах, де нові сервіси додавалися коли відкрилися ринкові можливості і віддалялися через кілька місяців або навіть тижнів після створення.

Такий акцент на тимчасовості - окремий випадок більш загального принципу модульного дизайну, який полягає в тому, що модульність

визначається швидкістю зміни функціоналу. Речі, які змінюються разом, повинні зберігатися в одному модулі. Частини системи, що змінюються рідко, не повинні перебувати разом з швидкоеволюціонуючими сервісами. Якщо регулярно міняти два сервісу разом, варто задуматись над тим, що можливо їх слід об'єднати.

Поміщення компоненту в сервіси додає можливість більш точного (granular) планування релізу. З монолітом будь-які зміни вимагають перезбирання і розгортання всієї програми. З мікросервісами вам потрібно розгорнути (redeploy) тільки ті сервіси, що змінилися. Це дозволяє спростити і прискорити процес релізу. Недолік такого підходу в тому, що вам доводиться хвилюватися щодо того, що зміни в одному сервісі зламають сервіси, які звертаються до нього. Традиційний підхід до інтеграції полягає в тому, щоб вирішувати такі проблеми шляхом версійності, але мікросервіси воліють використовувати версійність тільки в разі крайньої необхідності. Можливо уникнути версійності шляхом проектування сервісів так, щоб вони були настільки толерантні до змін сусідніх сервісів, наскільки можливо.

2.2 Визначення задачі

2.2.1 Опис вхідних даних

Окремо від адміністративних інтерфейсів системи, основна ідея має під собою одну основну структуру вхідних даних. На пристрій зчитування NFC поля має подаватись намагнічена картка типу MILFARE 10-13Mhz, що за попередніми налаштуваннями системи повинна відповідати певному користувачеві системи (студенту).

Пристрій повинен мати чітку прив'язку до світового часу, для того щоб кодувати кожен зчитанний магнітний ключ таймкодом для легітимного обрахунку відвідувань.

Тож основний цикл роботи системи визначається саме цима двома значеннями : магнітний ключ (унікальний хеш одного студента) та тайм код.

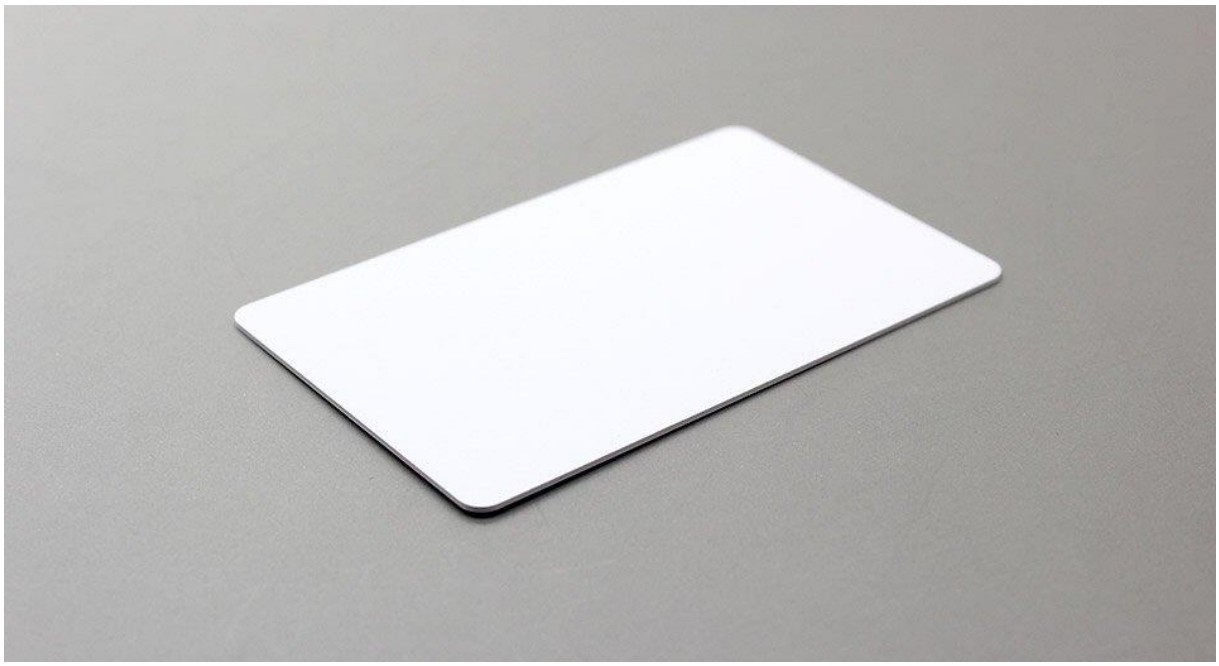

Рисунок 2.3 – Зовнішній вигляд магнітної картки стандарту MILFARE 13Mhz.

Для пере налаштування розкладів та

Загальна інформаційна система наслідує наступну структуру :

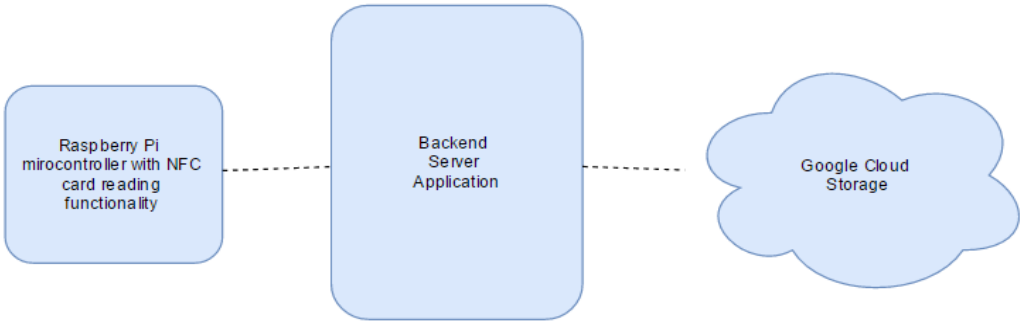

Рисунок 2.4 - Загальні сегменти системи обліку відвідувань.

Дана бакалаврська робота буде присвячена розробці периферійним елементам системи. Відповідно пристрій зчитування та механізму адміністрування Google Cloud. Тож більш детальна декомпозиція

стриктури інформаційної системи, примінимо до задач даної бакалаврської роботи виглядатиме наступним чином :

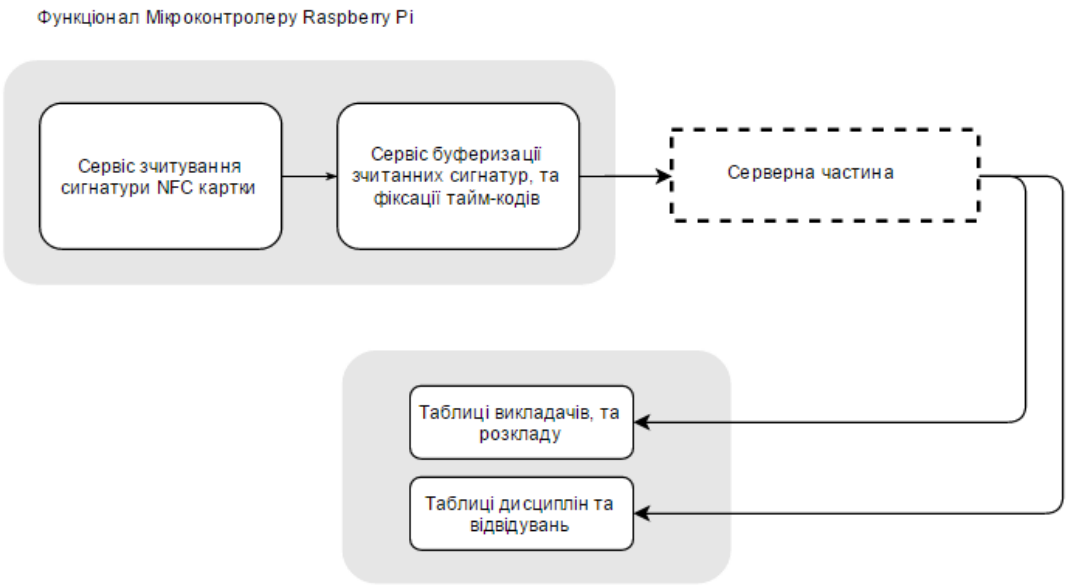

Рисунок 2.5 - Ключові функціональні сегменти периферії інформаційної системи.

2.2.2 Опис вихідних даних

За результатами розробки інформаційна система повинна пропонувати наступний функціонал. Функціонал віподвідно розрахований на три групи користувачів : адміністратори, викладачі, студенти. Система повинна також надавати можливості сегментації в інтересах її налаштування без глибокого втручання в структуру проекту.

Рисунок 2.6. Загальна Use-Case UML діаграмма для комплексного проекту.

Класс користувачів «адміністратор» так чи інакше має доступ фактично до кожного елементу функціоналу системи. Викладачі мають змогу редагувати загальну інформацію з приводу відвідувань та загалом організовувати дані часових інтервалів (для корректної інтерпретації тайм-кодів фіксації присутності студентів)

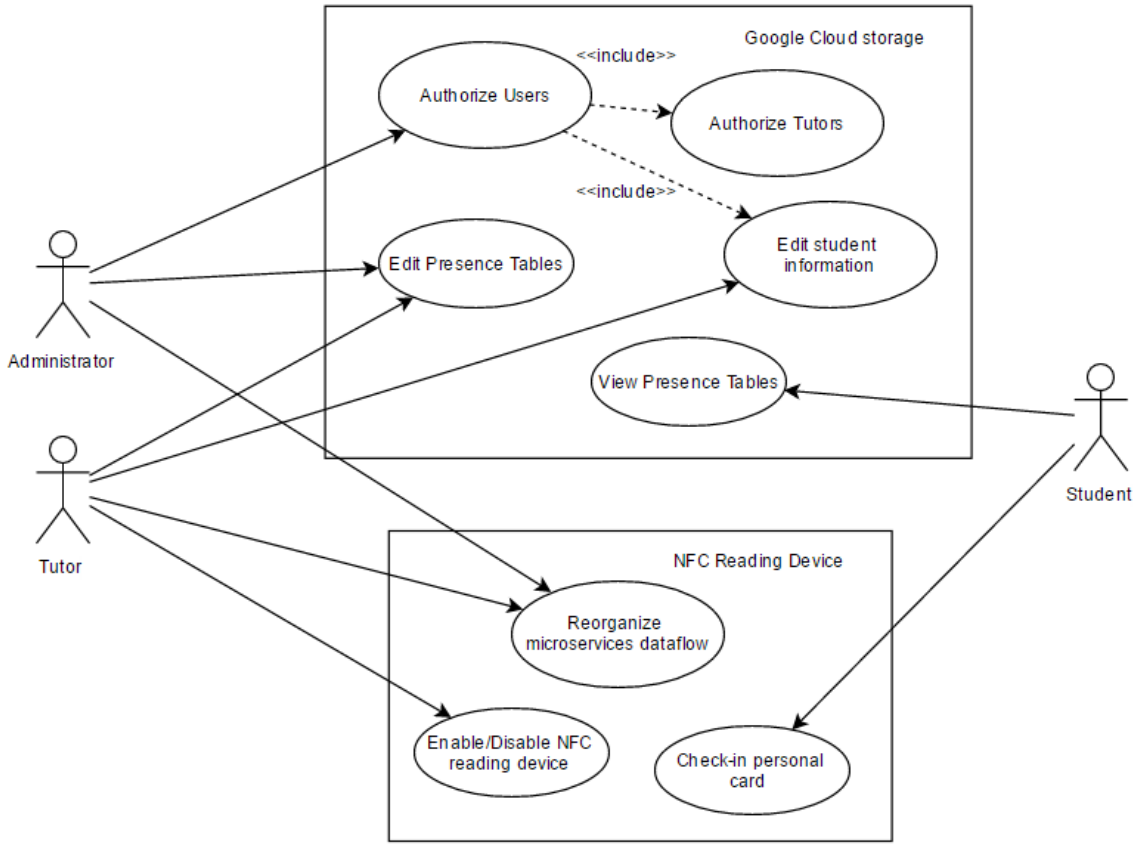

Рисунок 2.7 - Детальна Use-Case UML діаграмма.

Для роботи студента з системою необхідним забезпеченням є магнітна картка стандарту MILFARE 10-13Mhz, та реалізований вивід зафіксованих даних в часові таблиці на сервісі Google Sheets. Основними вихідними даними клієнтського пристрою зчитування є структура даних : пара поєднання унікального для кожного студента ключа намагніченої картки та тайм коду (моменту фіксації присутності студента поруч з пристроєм)

Основні механізми клієнтського пристрою мають бути доступні виключно авторизованим персонам : викладачам, та адміністраторм. Авторизація та доступ до системи регулюється адміністраторами через мета-дані на храмному хранилищі.

2.3 Висновки до розділу 2

В даному розділі було розглянуто етапи проектування інформаційної системи в мікросервісному стилі. Було розглянуто на які складові в цілому розподілена комплексна система, було визначено чіткі границі мікросервісів. Також було наведено опис вихідних даних та наведено приклад use-case діаграм.

2 РОБОТА З МІКРОКОНТРОЛЛЕРОМ

2.1 Опис задачі

Для реалізації клієнтської частини є ряд ключових, базисних задач необхідних до вирішення. На ряду з якими безпосередньо збірка пристрою та налаштування ОС для роботи з розширеннями мікроконьтроллеру. В порядку переддипломної практики було виконано саме фундаментальні критичні проблеми пов’язані з розробкою системи.

2.2 Збірка та комплектація пристрою.

Для реалізації інтерфейсу зчитування NFC міток використано модуль RC522 досупний в роздрібній торгівлі з готовими для тестування мітками в форматі картки та, магнітного ключа.

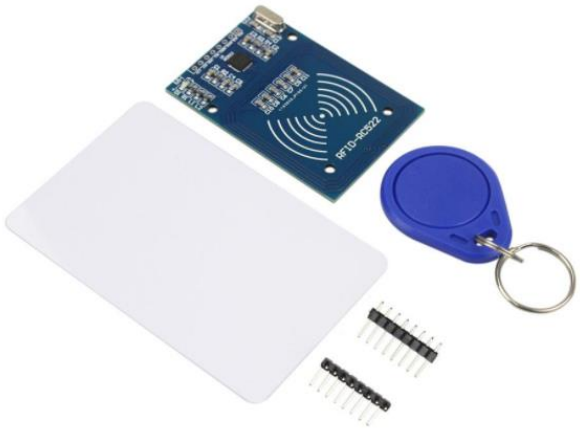

Рисунок 3.1 – Комплектація і вигляд модуля RC522.

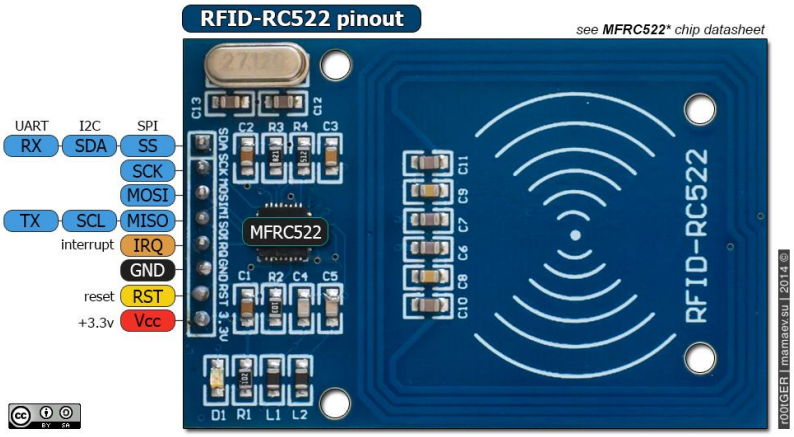

Рисунок 3.2 – Вигляд та інтерфейс плати RC522

Мікроконтроллер, що на даний момент () має ринкову ціну : …. грн.. поставляється без корпусу та кабелів в комплектації. Деталі необхідні для першого запуску та ініціалізації пристрою :

- HDMI кабель
- Usb клавіатура
- microSD карта
- екран з HDMI інтерфейсом (або DVI + перемикач з HDMI на DVI
- кабель microUSB на USB

Рис. 3.3 - Зовнішній вигляд мікроконтроллера.

Розробка пристроїв здійснюється Фондом *Raspberry Pi Foundation,* благодійною організацією зареєстрованою Комісією з благодійності. Фонд був заснований 5 травня 2009 року в Калдекот, Південний Кембриджшир, Великої Британії Його метою є «заохочення вивчення інформатики та суміжних наук, особливо на рівні школи, і знову зробити вивчення комп'ютерної науки цікавим» . Серед відомих засновників фонду Дейвид Брейбен, Джек Ланг, Піт Ломас, Роберт Маллінс, Алан Майкрофт і Ебен Аптон Подкаст з інтерв'ю з Ебен Аптон було представлено в червні 2011 року.

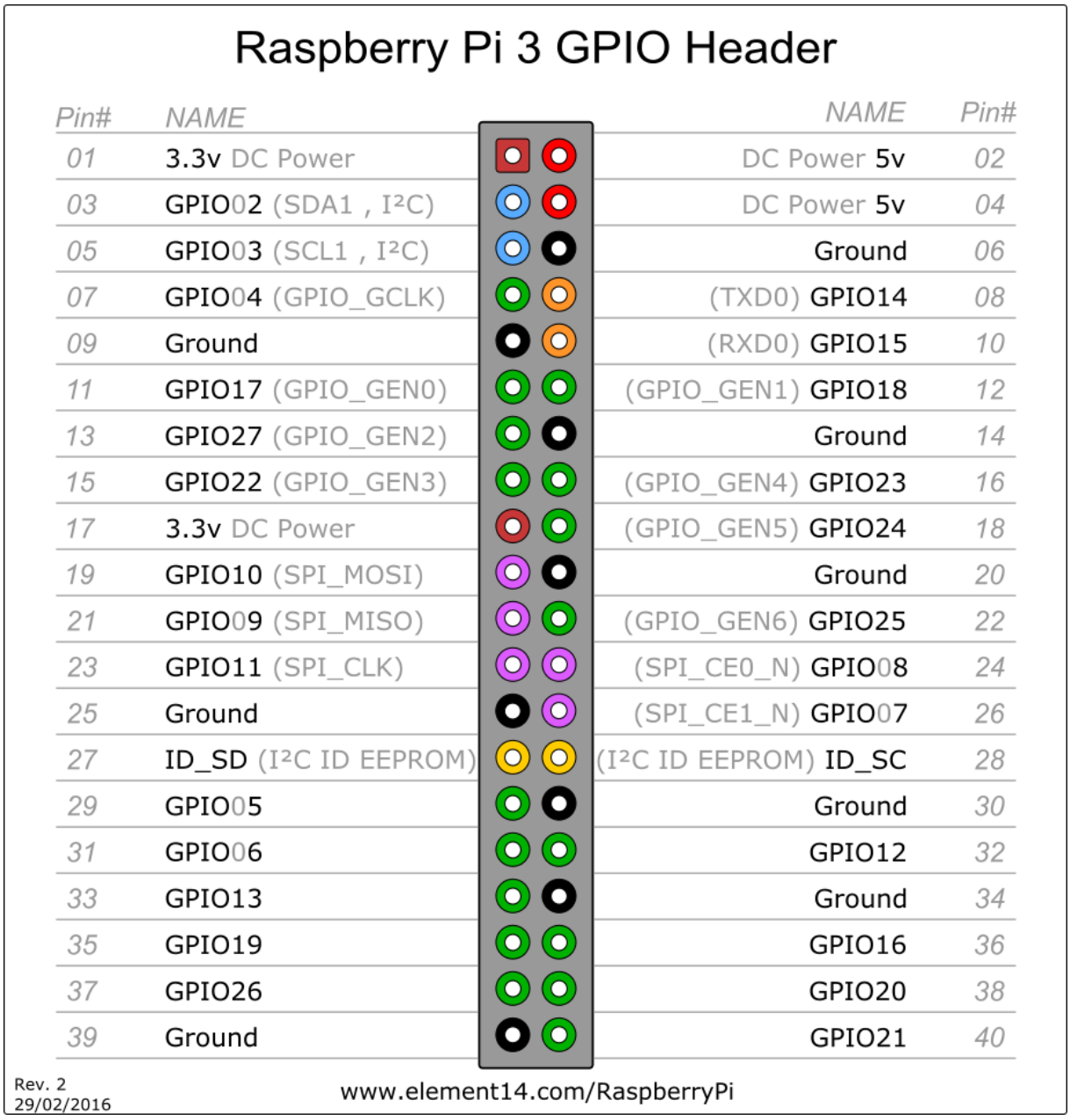

Рисунок 3.4 - Схема функціонального призначення пінів плати мікроконтроллера Raspberry P3

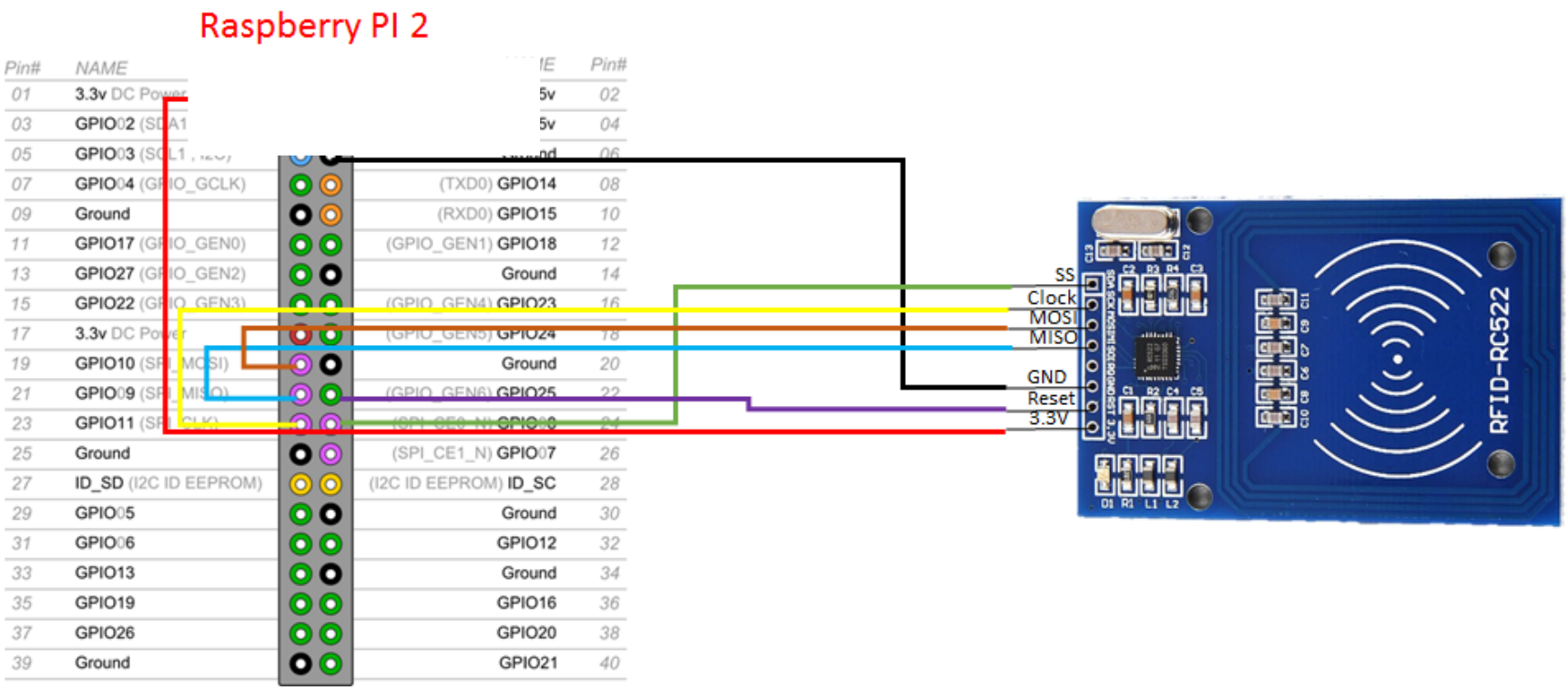

Рисунок 3.5 – Схема з'єднання модулю RC522 та плати мікроконтроллеру.

3.3 Запуск та налаштування мікроконтроллеру.

Далі наведено процедуру запуску та першого використання мікроконтролленру Raspberry Pi 3 в спрощеному мінімально вибагливому форматі.

Першим чином скачано з офіційного репозиторію останню збірку ОС Raspbian:

```
wget https://downloads.raspberrypi.org/raspbian_latest
unzip raspbian_latest
```

Після розархівування, отримано образ 2016-05-27-raspbian-jessie.img . Надалі необхідно списати образ на карту пам’яті, що буде використовуватись мікроконтроллером:

```
df -h
```

Карту необхідно від форматувати в программі GParted і від’єднати карту від файлової системи комп’ютера:

```
sudo umount /dev/sdb1
```

За допомогою утиліти “dd” образ записано на карту-носій :

```
sudo dd bs=4M if=2016-05-27-raspbian-jessie.img of=/dev/sdd
```

Необхідно мати на увазі, що параметр “” має містити вказівку на всю карту пам'яті цілком, а не її розділ. За бажанням можливо спостерігати за процессом копіювання встановиши утиліту “dcfldd”. ЇЇ можливо встановити коммандою:

```
sudo apt-get install dcfldd
```

В такому випадку команда копіювання виглядатиме наступним чином:

```
sudo dcfldd bs=4M if=2016-05-27-raspbian-jessie.img of=/dev/sdd
```

На цьому підготовка до запуску мікроконтроллера є завершенною. Отже після цього відбувається його запуск. Під'єднавши до доступного екрану мікроконтроллер через HDMI кабель, надавши на мікроконтроллер живлення, очікуваний екран запуску є наступним:

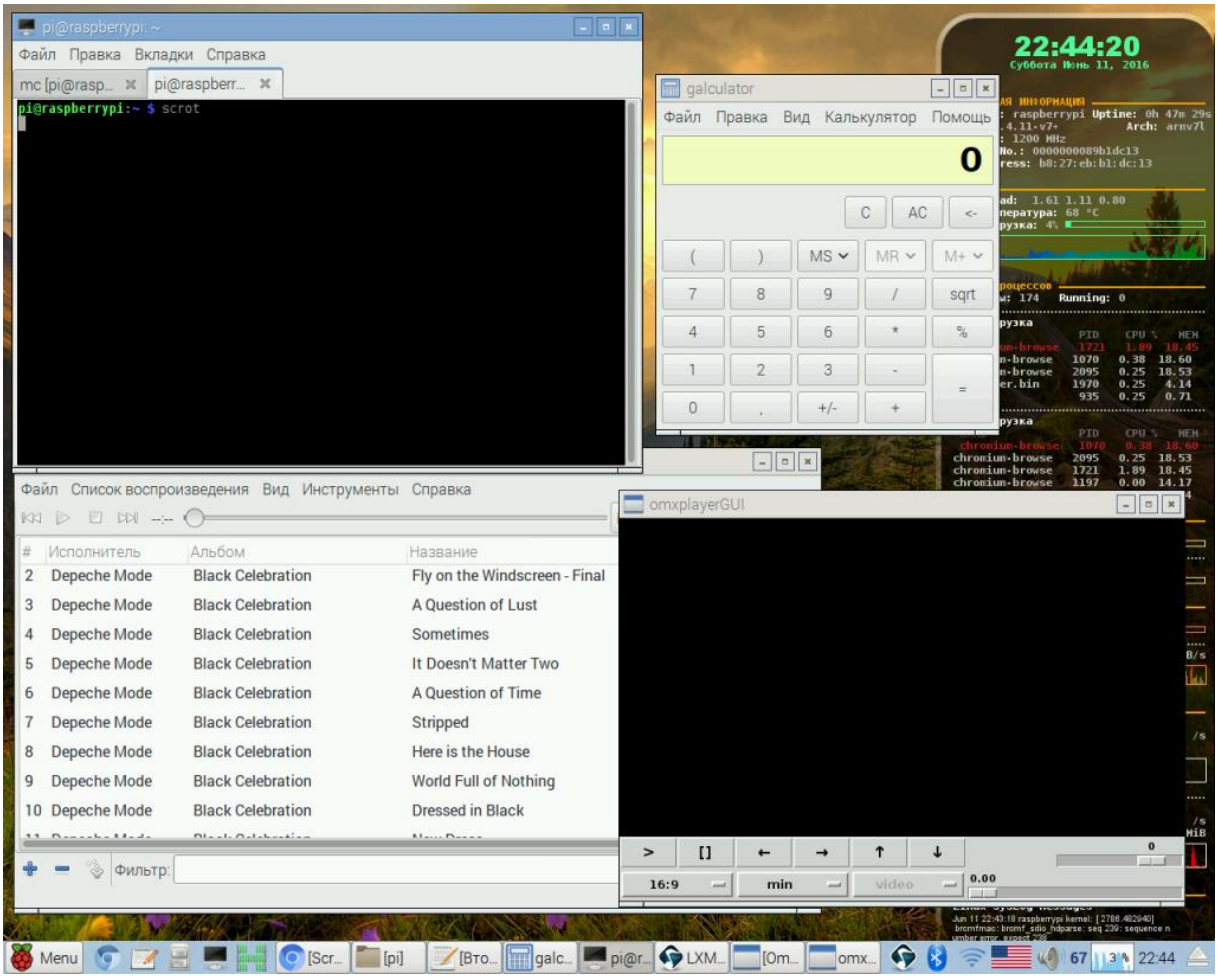

Рисунок 3.6 – Экран графічного інтерфейсу системи Raspbian.

Після таких кроків стає можливим налаштування самої системи. Тож в ході роботи, для базової підготовки мікроконтроллеру було змінено розмір SWAP сектору :

```
sudo leafpad /etc/dphys-swapfile
```

Змінено CONF_SWAPSIZE=100 на CONF_SWAPSIZE=1024

```
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

3.3 Реалізація основних механізмів роботи пристрою

Робота безпосередньо з системними ресурсами мікроконтроллера, для зчитування даних при піднесенні магнітної картки.

Даний лістинг складає роботу проведену над пристроєм зчитування магнітних картор, перш за все необхідну для можливості демонстрації прототипу всієї системи на прикладі навіть мінімального об’єму вхідних даних.

3.4 Робота бази даних на пристрої.

Реалізація буфера збереження даних для передачі через інтернет. Даних буфер несе в собі збережені тайм коди та ключі зчитаних магнітних карток. На цьому етапі не відбувається ніякої інтерпретації даних. Даний етап виконує чисто технічні задачі і подається до реалізації як окремий мікросервіс поперед усе для розгляду мікросервісного стилю программмування.

Фрагмент файлу «sqlite.py»

```
from time import strftime,localtime
import datetime
# TODO: change this code for SQLite needs
def connect():
    # Mysql connection setup. Insert your values here
    return sqlite3.connect('attendance.db')
def insertReading(tagId,action):
    conn = connect()
    cur = conn.cursor()
    currentTime=strftime("%Y%m%d%H%M%S", localtime())
    print(tagId)
    print(currentTime)
    # cur.execute("INSERT INTO readings (tagId, time, action) VALUES
('%s', '%s','%s')",(tagId,currentTime,action))
    cur.execute("INSERT INTO readings (tagId, time, action) VALUES
(?, ?, ?)",(tagId,currentTime,action))
    # cur.commit()
    # cur.execute("SELECT name,surname FROM users WHERE id = (SELECT
userId FROM cards WHERE tagId='%s' LIMIT 1)",(tagId))
    # cur.execute("SELECT name,surname FROM users WHERE id = (SELECT
userId FROM cards WHERE tagId=? LIMIT 1)",(tagId))
```

Оскільки інтерпретація та формування звітності відвідувань покладено не на пристрій зчитування, то за ним залишається дві основні задачі – зчитування та накопичення даних відвідувань. За це відповідає структура даних, що складається з унікального ключа (хеш-у студента), записаного на картці, та тайм-коду в фоматі YYYY/MM/DD/hh/mm/ss. І в рамках примінення мікросервісної архітектури, задачі були відповідно розподілені на 2 мікросервіси. Мікросервіс, що зчитує NFC картки, та мікросервіс що накопичує в собі збережені структури даних, і далі відправляє їх на сервер.

Community підтримка набирає критичної ролі в підтримці системи, тому як передбачена для мікроконтроллера операційна система Raspbian постійно зазнає змін та оновлень з метою оптимізації та підтримки актуальності ядра системи. Іноді це породжує конфлікти як наприклад в механізмах доступу до системних ресурсів. Саме таку проблему було виявлено під час налаштування роботи NFC модулю, оскільки оновлення системи №00913141212 реорганізувало механізми звернення до портів, що використовуються модулем.

Фрагмент файлу «»

ч

```
def deleteLastReading(tagId):
    # '''Deletes last reading inserted max 5(+1) minutes
ago'''
    checkTime = datetime.datetime.now() -
datetime.timedelta(minutes=6)
    db = connect()
    cur = db.cursor()
    # cur.execute("DELETE FROM readings WHERE tagId='%s'
AND time>'%s' ORDER BY time DESC LIMIT
1",(tagId,checkTime.strftime("%Y%m%d%H%M%S")))
    cur.execute("DELETE FROM readings WHERE tagId=? AND
time>? ORDER BY time DESC LIMIT
1",(tagId,checkTime.strftime("%Y%m%d%H%M%S")))
    row = cur.fetchone()
    db.close()
```

Скрипт роботи з буфером даних для передачі через інтернет в реалізації мовою Python на базі MySQL . Дані, чисто технічно відмінні сегменти інформаційної системи можуть послужити демонстрацією гнучкості не монолітної структури всього сервісу.

Фрагмент файлу «mysqldb.py».

```
import
MySQLdb
          from time import strftime,localtime
          import datetime
          from unidecode import unidecode
          # TODO: change this code for SQLite needs
          def connect():
              # Mysql connection setup. Insert your values here
              return MySQLdb.connect(host="localhost", user="pi",
          passwd="raspberry", db="pi")
          def insertReading(tagId,action):
              db = connect()
              cur = db.cursor()
              currentTime=strftime("%Y%m%d%H%M%S", localtime())
              cur.execute("""INSERT INTO readings (tagId, time,
          action) VALUES (%s, %s, %s)""",(tagId,currentTime,action))
              db.commit()
```

Поміщення компоненту в сервіси додає можливість більш точного (granular) планування релізу. З монолітом будь-які зміни вимагають перезбирання і розгортання всієї програми. З мікросервісами вам потрібно розгорнути (redeploy) тільки ті сервіси, що змінилися. Це дозволяє спростити і прискорити процес релізу. Недолік такого підходу в тому, що вам доводиться хвилюватися щодо того, що зміни в одному сервісі зламають сервіси, які звертаються до нього. Традиційний підхід до інтеграції полягає в тому, щоб вирішувати такі проблеми шляхом версійності, але мікросервіси воліють використовувати версійність тільки в разі крайньої необхідності. Можливо уникнути версійності шляхом проектування сервісів так, щоб вони були настільки толерантні до змін сусідніх сервісів, наскільки можливо.

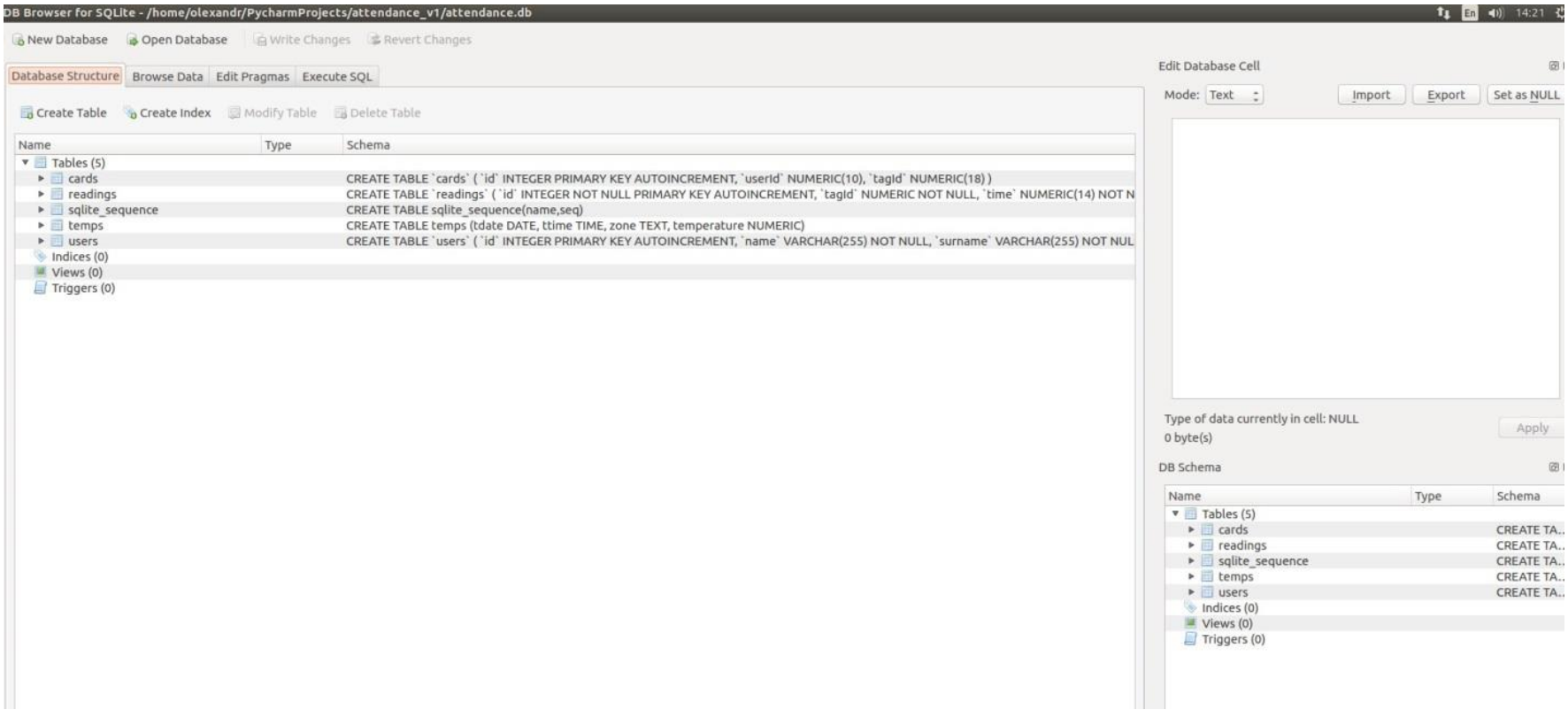

Рисунок 3.7 – Скріншот роботи СУБД программи буферизації мікроконтроллеру

3.4 Використаяння сервісу Docker Container в приміненні мікросервісної архітектури

З метою контейнеризації мікросервісів породжених в процессі приміннення мікросервісного стилю програмування, використано сервіс Docker Container. Dockerfile є файлом метаданих що керує та налаштовує створення контейнеру.

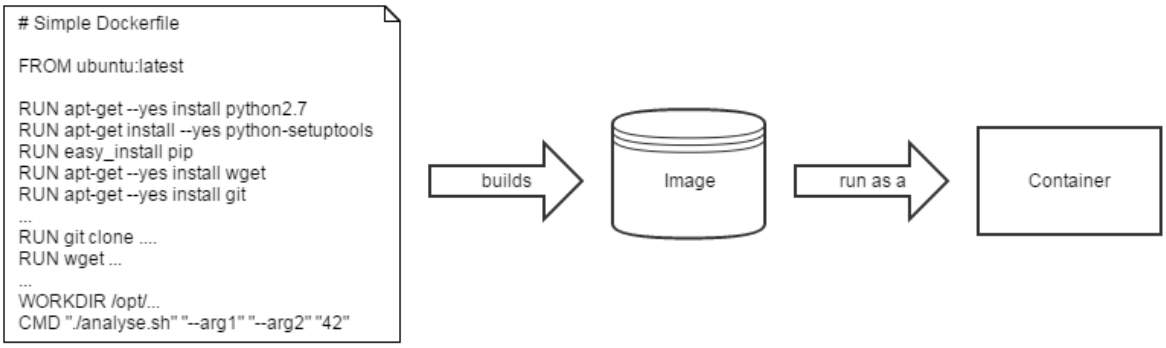

Рисунок 3.8. Загальный механізм Контейнеризації

Основною зручністю контейнеризації є те, що після такого процессу Deployment мікросервісу на сервер значно спрощується і фактично полягає в простому занесенні суммарного образу на робочий пристрій.

При наявності клієнтської сторони яка забезпечує системою даними на предмет часу переклички та присутності конкретних користувачів, задача постає в тому, щоб доповнити такі дані інформацією необхідною для формування повної звітності та реалізувати механізми експорту даних в зручний для перевикористання та перегляду формат.

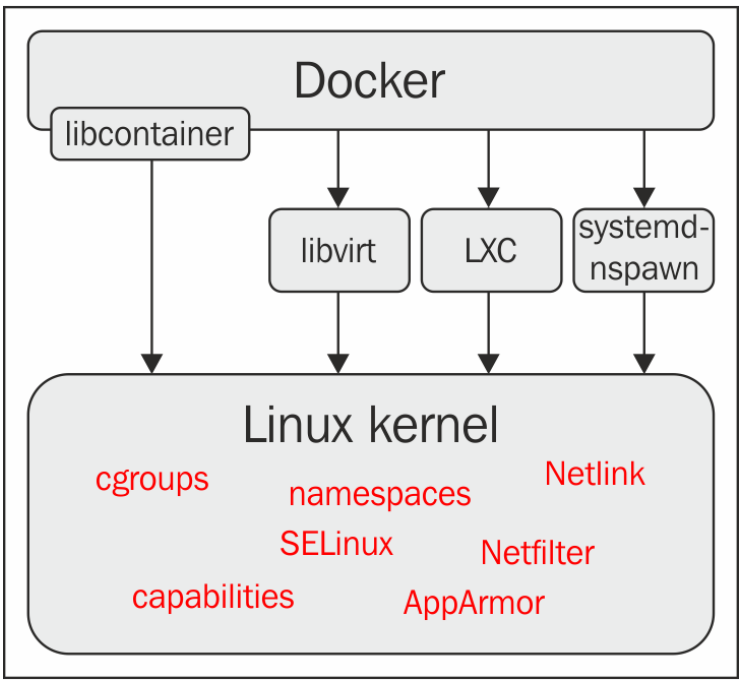

Рисунок 3.9 - Схема віртуалізації процессу в рамках Docker демону.

Саме як ідеологічний розвиток ідеї agile-у ми можемо взяти мікросервісний стиль програмування, адже така архітектура суттєво більш схильна до механізмів Continuous Integration та Continuous Deployment. Таким чином в процессі розробки, дійсно є можливість мати на кожному етапі робочий продукт. А за рахунок повної контейнеризації, система стає набагато більш платформо-незалежною і простою для портування. В контексті розробки учбових проектів її можливості практично довільної декомпозиції дозволяються долучати різні команди розробки до роботи над неймовірно масштабними задачами видаючи їм мінімальні незалежні задачі для рішення.

```
pi@raspberrypi:~/attendance $ sqlite3 attendance.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> SELECT * FROM readings;
sqlite> ^Z
[6]+  Stopped                 sqlite3 attendance.db
pi@raspberrypi:~/attendance $ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:AlexanderKutoviy/attendance
   44b91f5..bb857ef  master     -> origin/master
Updating 44b91f5..bb857ef
Fast-forward
 sqlite.py | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
pi@raspberrypi:~/attendance $ sqlite3 attendance.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> ^Z
[7]+  Stopped                 sqlite3 attendance.db
pi@raspberrypi:~/attendance $ python attendance.py
/home/pi/attendance/MFRC522.py:110: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(22, GPIO.OUT)
202311771577
20170508111204
INSERTED
pi@raspberrypi:~/attendance $ sqlite3 attendance.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> SELECT * FROM readings;
1|202311771577|20170508111204|1
sqlite> ^Z
[8]+  Stopped                 sqlite3 attendance.db
pi@raspberrypi:~/attendance $
```

Рисунок 3.10– Скріншот трафіку запитів до бази даних по факту зчитування.

3.5 Висновки до розділу 3

В розділі 3 дипломного проекту розглянуто основні умови роботи з мікроконтроллером Raspberry Pi та надано документацію до виконання повтороного налаштування мікроконтроллеру.

Community підтримка набирає критичної ролі в підтримці системи, тому як передбачена для мікроконтроллера операційна система Raspbian постійно зазнає змін та оновлень з метою оптимізації та підтримки актуальності ядра системи. Іноді це породжує конфлікти як наприклад в механізмах доступу до системних ресурсів. Саме таку проблему було виявлено під час налаштування роботи NFC модулю, оскільки оновлення системи №00913141212 реорганізувало механізми звернення до портів, що використовуються модулем.

За результатами запропонованих інструкцій, користувач має отримати необхідний до використання на встановлення мікрорсервісів клієнтський пристрій, що дозволить йому під'єднатись до інформаційної системи обліку відівдувань, за умови, що номер пристрою є введеним в таблиці мета-даних організації інформаційної системи.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для аналізу нелінійних нестаціонарних процесів. Інтерфейс користувача був розроблений за допомогою мов програмування Java, Python та C у середовищі розробки PyCharm, Intellij Idea та CLion. Інтерфейс користувача передбачений лише веб інтерфейсом і використовує готову базу Google Sheets.

Програмний продукт призначено для використання на персональних комерційних пристроях Raspberry Pi під управлінням операційної системи Raspbian.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На

цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на забезпечених портативних пристроях з попередньо налаштованим середовищем;

– забезпечувати високу швидкість обробки чітко передбачених об’ємів даних у реальному часі, з метою уникнення черг для зчитування карток;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка системи, яка реагуючи на чітки визначений формат взаємодії з клієнтом категоризує таку взаємодію відносно розкладу та регламенту учбового процесу. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – розпізнавання вхідних даних;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування C++;

Функція F_2:

а) занесення даних за допомогою візуальних способів розпізнання;

б) зчитування NFC сигнатури.

Функція F_3:

а) інтерфейс користувача, що керується скриптами Google Sheets;

б) Написання окремого веб-інтерфейсу, для маніпуляцій з системою.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

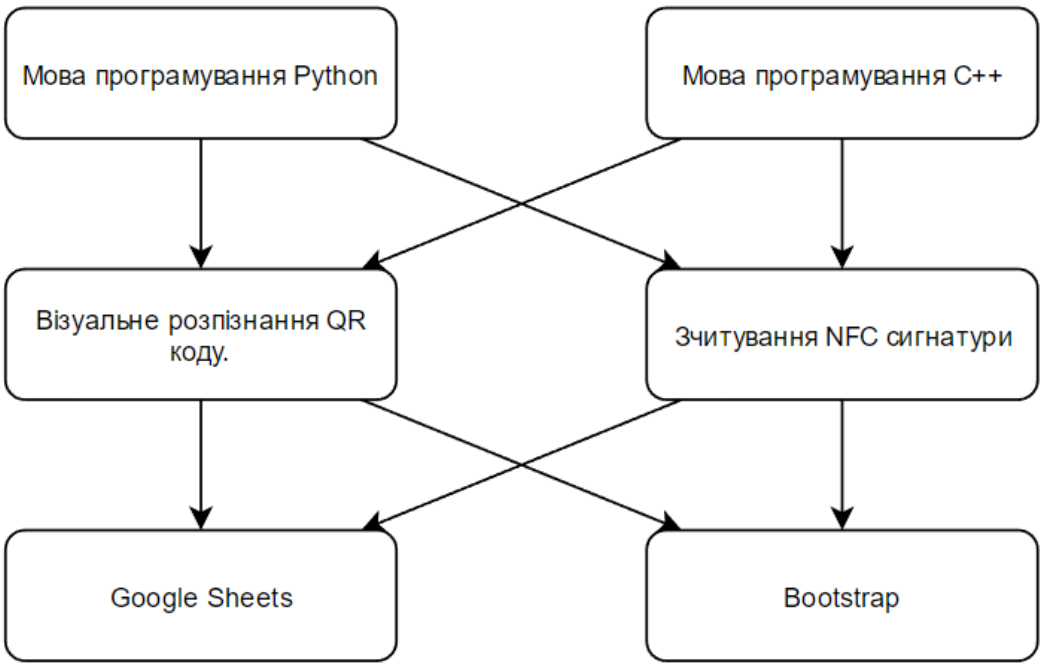

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Доступний і гнучкий. Легше інтегрується для взаємодії з системними ресурсами.	Повільніший, не багато поточний.
	Б	Код швидко виконується, кросплатформений	Складний в підтримці та рефакторингу. Більш вибагливий при запуску проекта.
F2	А	Простий у використанні	Менша точність підрахунків
	Б	Найоптимальніший для використання тільки у власних програмних продуктах	Затрачений час
F3	*А*	Легкий у створенні	Відсутність кросплатформеності
	Б	Стабільний у використанні	Необхідна додаткова Інсталяція

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант б) має бути відкинутий.

Функція *F2:*

Оскільки планується розглядати широкий спектр різноманітних варіантів даних, та перший є більш зручним в даному випадку, то варіант б) має бути відкинутий

Функція *F3:*

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1а – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	350	320	300
Час обробки даних алгоритмом	X3	мс	5000	600	80
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

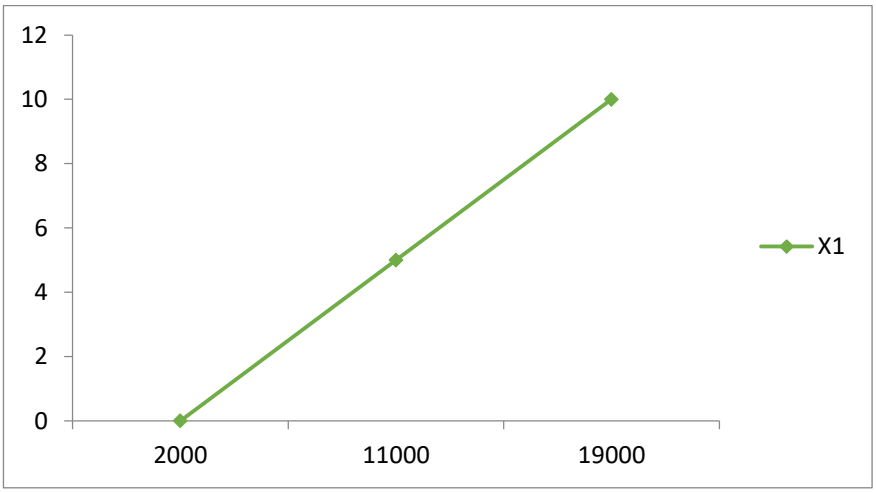

Рисунок 4.2 – X1, швидкодія мови програмування

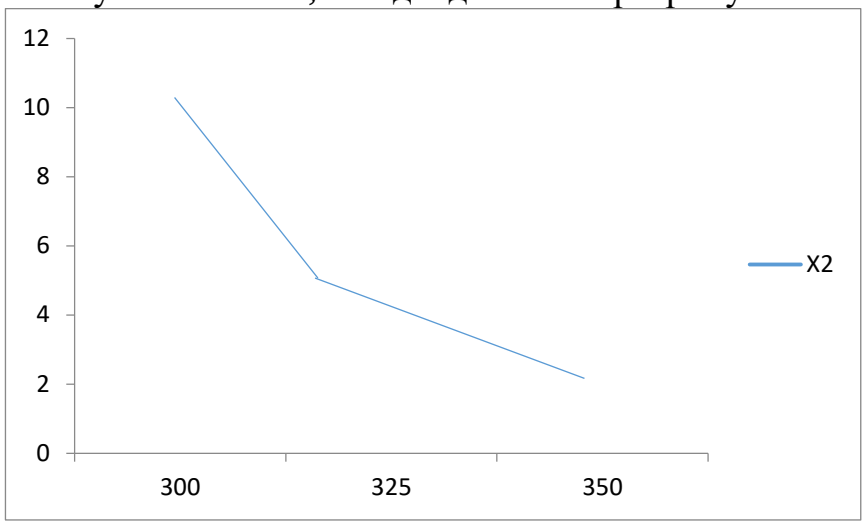

Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

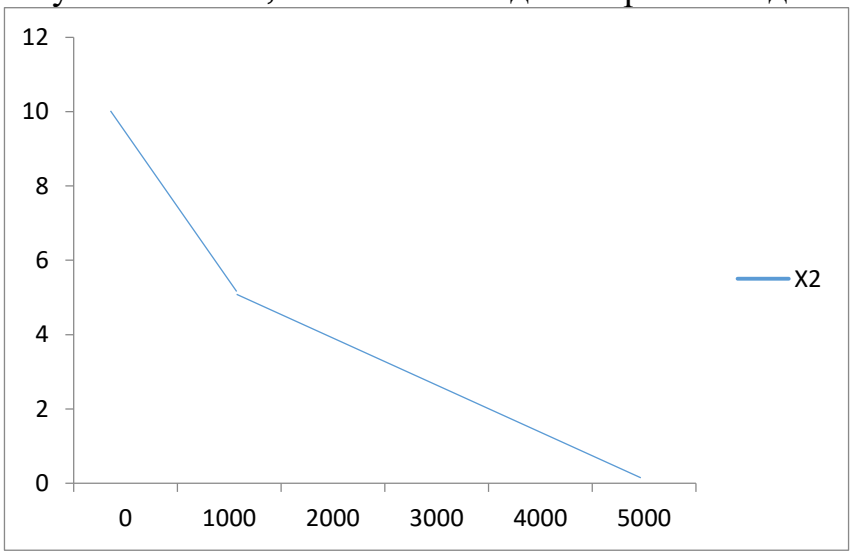

Рисунок 4.4 – X3, час обробки даних алгоритмом

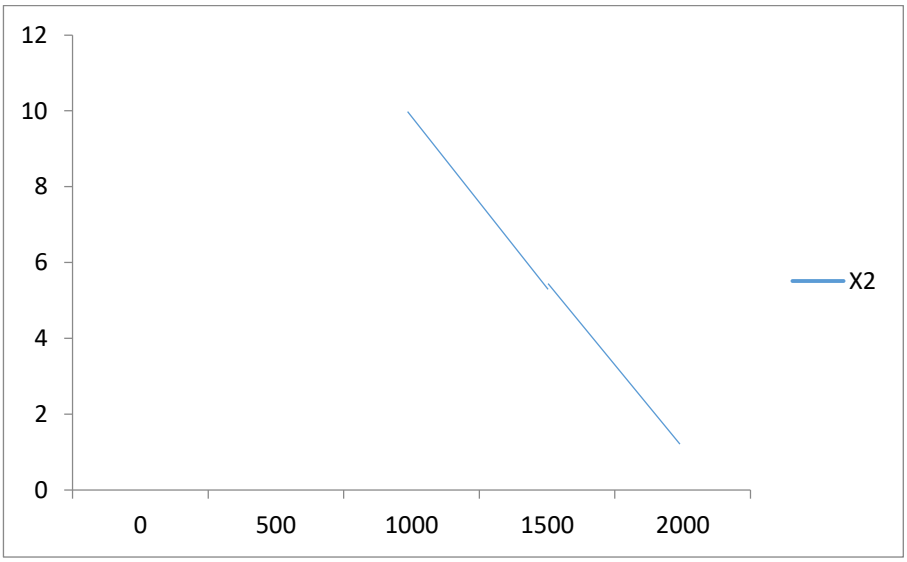

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю4.4.

Таблиця4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}$,де $b_i = \sum_{i=1}^{N} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$,де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	80	1	0,154	0,154

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,835 = 2,57

K_{K2} = 0,774 + 0,962 + 0,154 = 1,89

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =70 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.5. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1$ = 70·1.7·0.8 = 105.8 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =32 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2$ = 32 · 0.9 · 0.8 = 32.04 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (105.8 + 32.04 + 4.8 + 32.04) · 8 = 1397,44 людино-годин;

$Т_{II}$ = (105.8 + 32.04 + 6.91 + 32.04) · 8 = 1414.32 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 10000 грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.},$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_Ч = \frac{20000}{3 \cdot 21 \cdot 8} = 39{,}68 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д,$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_і$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 39,68 \cdot 1397.44 \cdot 1.2 = 66540,5$ грн.

II. $С_{ЗП} = 39,68 \cdot 1414.32 \cdot 1.2 = 67344,26$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (ФОП II групи) становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 66540.5 \cdot 0.22 = 14639$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 67344,26 \cdot 0.22 = 14816$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 10000 \cdot 0,2 = 24000$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 24000 \cdot (1 + 0.2) = 28800$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.3677 = 24000 \cdot 0,22 = 5280$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 10000 \cdot 0.05 = 500$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,2436 \cdot 2 = 129,695$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 17280 + 6353,86 + 2300 + 460 + 129,69 + 5360 = 31883,55$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 31883,55/1706,4 = 18,68$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 18,68 \cdot 1328,64 = 24819$ грн.;

II. $С_М = 18,68 \cdot 1345.52 = 25134,52$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 66437,31 \cdot 0,67 = 44513$ грн.;

II. $С_Н = 67281,38 \cdot 0,67 = 45078,52$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 66437,31 + 24429 + 24819 + 44513 = 160198,31$ грн.;

II. $С_{ПП} = 67281,38 + 24739 + 25134,52 + 45078,52 = 162233,42$ грн.;

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1} = 5,214 / 160198,31 = 0,33 \cdot 10^{-4}$;

$К_{ТЕР2} = 3,569 / 162233,42 = 0,22 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,33 \cdot 10^{-4}$.

4.6 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-

економічного рівня якості

$К_{ТЕР} = 0,33 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Python;

– зчитування NFC сигнатури магнітних карток;

– інтерфейс користувача, на базі Google Forms.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В ході роботи досліджено аспекти примінення мікросервісного стилю програмування, як альтернатива монолітній архітектурі, у випадку створення клієнтської частини для складної інформаційної системи, що використовує власну платформу для донесення розробленого функціоналу. Таку платформу реалізовано на базі мікроконтроллера Raspberry Pi 3 з передвстановленою операційною системою Raspbian.

Реалізовано компонентний зв’язок та роботу модуля зчитування NFC поля RFID RC522 для роботи з магнітними ключами. В якості магнітних карток ідентифікації користувача обрано картки типу MILFARE 13Mhz, за їх досутпність на роздрібному ринку та розповсюдженність технології на інших магнітних носіях. За бажанням необхідна магнітна сигнатура може бути згенерована за допомогою телефону, що має підтримку технології NFC, що значно полегшує задачу матеріальної підтримки, адміністрування, та відмовостійкість всієї системи реєстрації пристуності.

Було виконане налаштування мікроконтроллеру від заводського стану до аспектів контейнеризації та використання системних ресурсів зчитування NFC поля. В ході таких маніпуляцій мікросервісна архітектура проявила себе якнайкращим чином, оскільку дозволяла надгнучку організацію та редагування проекту, і відповідне підлаштування під аппаратні потреби, які були так само зформовані в ході виконання проекту. В результаті отримано один базовий прототип клієнтського пристрою, з доступним тестовим зразком картки ідентифікації студента. На пристрої реалізовано механізму доступу до процессів буферизації та пересилання накопичених даних з приводу зафіксованих відвідувань. Основна сукупність сервісів та функціоналу пристрою напрямлена на відмовостійкіст системи та захист від втрати даних за технічних причин.

За рахунок мікросервісного стилю програмування, зв’язок пристрою з серверною частиною може бути легко переналаштований. Досліджена можливість окремої контейнеризації мікросервісів пристрою також даже підстави пере використання та значного розширення роботи

мікроконтроллера, без суттєвих проблем інтеграції нового функціоналу з попереднім.

В окремих розділах роботи описані основні теоретичні та визнані індустрією застави та аспекти мікросервісної архітектури.

ПЕРЕЛІК ПОСИЛАНЬ

1. Raspberry Pi. Manufacturer’s Documentation. – Режим доступа: http://www.raspberrypi.org/archives/2180 /. – Дата доступа: 17.11.2016
2. Microservices. – Режим доступу: https://martinfowler.com/. – Дата доступу: 14.05.2017
3. *Balalaie A.,* The Philosophy of Microservice Architecture / Microservices Book. *Balalaie A., 14.04.2015 Birmingham. Режим доступа:* microservicesbook.io. / Дата доступу: 04.05.2017
4. Microservices: Five Architectural Constraints. – Режим доступу: http://www.nirmata.com/2015/02/02/microservices-five-architectural-constraints/. – Дата доступу: 20.05.2017
5. Ньюмен С. Створення мікросервісів / Сем Ньюмен. – СПБ. : Питер., 2014-543 с.
6. Continuous Deployment: Strategies. – Режим доступу: https://www.javacodegeeks.com/2014/12/continuous-deployment-strategies.html /– Дата доступу: 18.04.2017
7. Офіційний сайт компанії W3Techs. – Режим доступу: https://w3techs.com/. Дата доступу: 27.05.2017
8. Oliver Wolf. Introduction into Microservices. Режим доступа: https://specify.io/concepts/microservices Дата доступа: 14.04.2017
9. Experiences from Failing with Microservices. – Режим доступу: https://www.infoq.com/news/2014/08/failing-microservices/. Дата доступу: 14.03.2017
10. Фаулер М. Архітектура корпоративних програмних додатків/ Мартін Фаулер. – СПБ. : Питер., 2014-543 с.
11. HTTPS on Stack Overflow: The End of a Long Road. – Режим доступу: https://nickcraver.com/blog/2017/05/22/https-on-stack-overflow/?utm_source=telegram.me&utm_medium=social&utm_campaign=vchera-stack-overflow-vklyuchili-https-po-d#performance-http2. Дата доступу: 18.04.2017

12. Бьюли. А. Изучаем SQL/ Алана Бьюли. – СПБ. : Питер., 2016-303 с.
13. Docker documentation. – Режим доступа: https://docs.docker.com/. – Дата доступа: 20.05. 17
14. Пол М. Continuous Integration: Improving Software Quality and Reducing Risk / Пол М. Дюваль. – Massachusetts: Addison-Wesley Professional., 2007-336 с.

Додаток А

Лістинг файлу «MFRC522.py»

```
import
RPi.GPIO
as GPIO
          import spi
          import signal
          class MFRC522:
            NRSTPD = 22
            MAX_LEN = 16
            PCD_IDLE       = 0x00
            PCD_AUTHENT    = 0x0E
            PCD_RECEIVE    = 0x08
            PCD_TRANSMIT   = 0x04
            PCD_TRANSCEIVE = 0x0C
            PCD_RESETPHASE = 0x0F
            PCD_CALCCRC    = 0x03
            PICC_REQIDL    = 0x26
            PICC_REQALL    = 0x52
            PICC_ANTICOLL  = 0x93
            PICC_SElECTTAG = 0x93
            PICC_AUTHENT1A = 0x60
            PICC_AUTHENT1B = 0x61
            PICC_READ      = 0x30
            PICC_WRITE     = 0xA0
            PICC_DECREMENT = 0xC0
            PICC_INCREMENT = 0xC1
            PICC_RESTORE   = 0xC2
            PICC_TRANSFER  = 0xB0
            PICC_HALT      = 0x50
            MI_OK       = 0
            MI_NOTAGERR = 1
            MI_ERR      = 2
            Reserved00     = 0x00
```

```
CommandReg     = 0x01
CommIEnReg     = 0x02
DivlEnReg      = 0x03
CommIrqReg     = 0x04
DivIrqReg      = 0x05
ErrorReg       = 0x06
Status1Reg     = 0x07
Status2Reg     = 0x08
FIFODataReg    = 0x09
FIFOLevelReg   = 0x0A
WaterLevelReg  = 0x0B
ControlReg     = 0x0C
BitFramingReg  = 0x0D
CollReg        = 0x0E
Reserved01     = 0x0F
Reserved10     = 0x10
ModeReg        = 0x11
TxModeReg      = 0x12
RxModeReg      = 0x13
TxControlReg   = 0x14
TxAutoReg      = 0x15
TxSelReg       = 0x16
RxSelReg       = 0x17
RxThresholdReg = 0x18
DemodReg       = 0x19
Reserved11     = 0x1A
Reserved12     = 0x1B
MifareReg      = 0x1C
Reserved13     = 0x1D
Reserved14     = 0x1E
SerialSpeedReg = 0x1F
Reserved20        = 0x20
CRCResultRegM     = 0x21
CRCResultRegL     = 0x22
Reserved21        = 0x23
ModWidthReg       = 0x24
```

```
    Reserved22        = 0x25
    RFCfgReg          = 0x26
    GsNReg            = 0x27
    CWGsPReg          = 0x28
    ModGsPReg         = 0x29
    TModeReg          = 0x2A
    TPrescalerReg     = 0x2B
    TReloadRegH       = 0x2C
    TReloadRegL       = 0x2D
    TCounterValueRegH = 0x2E
    TCounterValueRegL = 0x2F
    Reserved30      = 0x30
    TestSel1Reg     = 0x31
    TestSel2Reg     = 0x32
    TestPinEnReg    = 0x33
    TestPinValueReg = 0x34
    TestBusReg      = 0x35
    AutoTestReg     = 0x36
    VersionReg      = 0x37
    AnalogTestReg   = 0x38
    TestDAC1Reg     = 0x39
    TestDAC2Reg     = 0x3A
    TestADCReg      = 0x3B
    Reserved31      = 0x3C
    Reserved32      = 0x3D
    Reserved33      = 0x3E
    Reserved34      = 0x3F
    serNum = []
    def __init__(self,spd=1000000):
      spi.openSPI(speed=spd)
      GPIO.setmode(GPIO.BOARD)
      GPIO.setup(22, GPIO.OUT)
      GPIO.output(self.NRSTPD, 1)
      self.MFRC522_Init()
    def MFRC522_Reset(self):
      self.Write_MFRC522(self.CommandReg,
```

```
self.PCD_RESETPHASE)
  def Write_MFRC522(self,addr,val):
    spi.transfer(((addr<<1)&0x7E,val))
  def Read_MFRC522(self,addr):
    val = spi.transfer((((addr<<1)&0x7E) | 0x80,0))
    return val[1]
  def SetBitMask(self, reg, mask):
    tmp = self.Read_MFRC522(reg)
    self.Write_MFRC522(reg, tmp | mask)
  def ClearBitMask(self, reg, mask):
    tmp = self.Read_MFRC522(reg);
    self.Write_MFRC522(reg, tmp & (~mask))
  def AntennaOn(self):
    temp = self.Read_MFRC522(self.TxControlReg)
    if(~(temp & 0x03)):
      self.SetBitMask(self.TxControlReg, 0x03)
  def AntennaOff(self):
    self.ClearBitMask(self.TxControlReg, 0x03)
  def MFRC522_ToCard(self,command,sendData):
    backData = []
    backLen = 0
    status = self.MI_ERR
    irqEn = 0x00
    waitIRq = 0x00
    lastBits = None
    n = 0
    i = 0
    if command == self.PCD_AUTHENT:
      irqEn = 0x12
      waitIRq = 0x10
    if command == self.PCD_TRANSCEIVE:
      irqEn = 0x77
      waitIRq = 0x30
    self.Write_MFRC522(self.CommIEnReg, irqEn|0x80)
    self.ClearBitMask(self.CommIrqReg, 0x80)
    self.SetBitMask(self.FIFOLevelReg, 0x80)
```

```
        self.Write_MFRC522(self.CommandReg, self.PCD_IDLE);
        while(i<len(sendData)):
          self.Write_MFRC522(self.FIFODataReg, sendData[i])
          i = i+1
        self.Write_MFRC522(self.CommandReg, command)
        if command == self.PCD_TRANSCEIVE:
          self.SetBitMask(self.BitFramingReg, 0x80)
        i = 2000
        while True:
          n = self.Read_MFRC522(self.CommIrqReg)
          i = i - 1
          if ~((i!=0) and ~(n&0x01) and ~(n&waitIRq)):
            break
        self.ClearBitMask(self.BitFramingReg, 0x80)
        if i != 0:
          if (self.Read_MFRC522(self.ErrorReg) & 0x1B)==0x00:
            status = self.MI_OK
            if n & irqEn & 0x01:
              status = self.MI_NOTAGERR
            if command == self.PCD_TRANSCEIVE:
              n = self.Read_MFRC522(self.FIFOLevelReg)
              lastBits = self.Read_MFRC522(self.ControlReg) &
0x07
              if lastBits != 0:
                backLen = (n-1)*8 + lastBits
              else:
                backLen = n*8
              if n == 0:
                n = 1
              if n > self.MAX_LEN:
                n = self.MAX_LEN
              i = 0
              while i<n:

backData.append(self.Read_MFRC522(self.FIFODataReg))
                i = i + 1;
```

```
        else:
          status = self.MI_ERR
      return (status,backData,backLen)
    def MFRC522_Request(self, reqMode):
      status = None
      backBits = None
      TagType = []
      self.Write_MFRC522(self.BitFramingReg, 0x07)
      TagType.append(reqMode);
      (status,backData,backBits) =
self.MFRC522_ToCard(self.PCD_TRANSCEIVE, TagType)
      if ((status != self.MI_OK) | (backBits != 0x10)):
        status = self.MI_ERR
      return (status,backBits)
    def MFRC522_Anticoll(self):
      backData = []
      serNumCheck = 0
      serNum = []
      self.Write_MFRC522(self.BitFramingReg, 0x00)
      serNum.append(self.PICC_ANTICOLL)
      serNum.append(0x20)
      (status,backData,backBits) =
self.MFRC522_ToCard(self.PCD_TRANSCEIVE,serNum)
      if(status == self.MI_OK):
        i = 0
        if len(backData)==5:
          while i<4:
            serNumCheck = serNumCheck ^ backData[i]
            i = i + 1
          if serNumCheck != backData[i]:
            status = self.MI_ERR
        else:
          status = self.MI_ERR
      return (status,backData)
    def MFRC522_Init(self):
      GPIO.output(self.NRSTPD, 1)
```

```
        self.MFRC522_Reset();
        self.Write_MFRC522(self.TModeReg, 0x8D)
        self.Write_MFRC522(self.TPrescalerReg, 0x3E)
        self.Write_MFRC522(self.TReloadRegL, 30)
        self.Write_MFRC522(self.TReloadRegH, 0)
        self.Write_MFRC522(self.TxAutoReg, 0x40)
        self.Write_MFRC522(self.ModeReg, 0x3D)
        self.AntennaOn()
```

Лістинг файлу «DaemonBroker.java»

```
package
attendance.daemon.logpar
ser;
                        import
                        com.mchange.v2.c3p0.ComboPooledDataSource
                        ;
                        import
                        org.apache.commons.io.input.Tailer;
                        import
                        org.apache.commons.io.input.TailerListene
                        r;
                        import
                        org.apache.logging.log4j.LogManager;
                        import org.apache.logging.log4j.Logger;
                        import rx.Scheduler;
                        import rx.schedulers.Schedulers;
                        import rx.subjects.PublishSubject;
                        import rx.subjects.SerializedSubject;
                        import java.io.File;
                        import java.io.IOException;
                        import java.sql.Connection;
                        import java.text.SimpleDateFormat;
                        import java.util.Date;
                        import java.util.List;
                        import java.util.concurrent.Executors;
```

```
import
java.util.concurrent.ThreadPoolExecutor;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class EximLogParser implements
Daemon {
    @NameProperty
    public String filePath;
    @NameProperty
    public String namespace;
    public final ThreadPoolExecutor
executor = (ThreadPoolExecutor)
Executors.newFixedThreadPool(1, r -> new
Thread(r, "ExecutorThread"));
    private SimpleDateFormat dFormat =
new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
    private Logger l =
LogManager.getLogger(getClass());
    SerializedSubject<String, String>
subj =
PublishSubject.<String>create().toSeriali
zed();
    ComboPooledDataSource cpds;
    String lastTimestamp;
    Long prevTimestamp;
    Scheduler scheduler =
Schedulers.from(executor);
    Thread thread;
    Tailer tailer;
    @Override
    public void run() throws Exception {
        final Pattern entryRegex =
Pattern.compile("(?i)^(?<time>\\d{4}-
\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2})
(?<id>[a-z0-9]{6}-[a-z0-9]{6}-[a-z0-
```

```
9]{2}) (?<msg>.*)$");
        try (Connection conn =
cpds.getConnection()) {
            lastTimestamp =
SqlQuery.create(conn, "SELECT
TIMESTAMP(MAX(timestamp)) FROM
ProviderMailEvent WHERE
namespace=:serverId")
                    .set("serverId",
namespace)
                    .queryString();
            if (lastTimestamp != null)
                lastTimestamp =
lastTimestamp.substring(0, "2015-00-00
00:00:00".length());
            l.trace("Last timestamp: " +
lastTimestamp);
        }
        subj.map((String t) ->
toLogEntry(t, entryRegex))
                .filter((LogEntry t) -> t
!= null)
                .filter((LogEntry t) ->
lastTimestamp == null ||
lastTimestamp.compareTo(t.timestamp) <=
0)
                .lift(new
OperatorOnBackpressureGroup<>(scheduler))

.subscribe((List<LogEntry> t) -> {
                    boolean retry = true;
                    while (retry)
                        try {
                            retry =
false;
                            try
```

```
(Connection conn = cpds.getConnection())
{
                                    try {

conn.setAutoCommit(false);
                                          for
(LogEntry e : t) {

long time =
dFormat.parse(e.timestamp).getTime();

if (prevTimestamp != null && time <
prevTimestamp) {

l.warn("Log timestamp {} is less than
previous timestamp {}. Doublechek it.",

e.timestamp,

dFormat.format(new Date(prevTimestamp))

);

lastTimestamp = e.timestamp;
                                                }

prevTimestamp = time;

if (lastTimestamp != null &&
lastTimestamp.equals(e.timestamp)) {

Integer has = SqlQuery.create(conn,
"SELECT 1 FROM ProviderMailEvent " +

"WHERE namespace=:serverId " +
```

```
                "AND mailId=:mailId " +
                "AND timestamp=:timestamp " +
                "AND message=:message")
                .set("serverId", namespace)
                .set("mailId", e.mailId)
                .set("timestamp", e.timestamp)
                .set("message", e.message)
                .queryInt();
                if (has != null)
                continue;
                                                    }

                new SqlSimpleInsert(conn,
                "ProviderMailEvent")
                .set("namespace", namespace)
                .set("mailId", e.mailId)
                .set("timestamp", e.timestamp)
                .set("message", e.message)
                .execute();
                                                }

                conn.commit();
```

```
                                    } catch
(Throwable e) {

conn.rollback();
                                        throw
e;
                                    } finally
{

conn.setAutoCommit(true);
                                    }
                                }
                            } catch
(Exception ex) {

l.error("Error. Sleeping 30 sec before
retry", ex);
                                retry = true;
                                try {

Thread.sleep(30000);
                                } catch
(InterruptedException ex1) {

l.error("", ex);
                                }
                            }
                    });
            File file =
getAbsoluteFile(File.listRoots()[0],
filePath);
            tailer = new Tailer(file, new
TailerListener() {
                @Override
                public void init(Tailer
tailer) {
```

```
                    l.trace("Init tailer");
                    try {
                        l.trace("File: " +
tailer.getFile().getCanonicalPath());
                    } catch (IOException ex)
{
                        l.error("", ex);
                    }
                }
                @Override
                public void fileNotFound() {
                    l.trace("File not
found");
                }
                @Override
                public void fileRotated() {
                    l.trace("File rotated");
                }
                @Override
                public void handle(String
line) {
                    subj.onNext(line);
                }
                @Override
                public void handle(Exception
ex) {
                    l.error("Tailer error",
ex);
                }
            }, 500, false, false, 10000);
            thread = new Thread(tailer);
            thread.start();
        }
        @Override
        public void stop() throws Exception {
            thread.interrupt();
```

```
            tailer.stop();
        }
        @Override
        public void
setCpds(ComboPooledDataSource cpds) {
            this.cpds = cpds;
        }
        @Override
        public boolean isStopped() {
            return thread.isInterrupted();
        }
        public static class LogEntry {
            public String timestamp;
            public String mailId;
            public String message;
            @Override
            public String toString() {
                return "LogEntry{" +
"timestamp=" + timestamp + ", mailId=" +
mailId + ", message=" + message + '}';
            }
        }
        private LogEntry toLogEntry(String t,
Pattern entryRegex) {
            Matcher m =
entryRegex.matcher(t);
            if (m.matches()) {
                LogEntry e = new LogEntry();
                e.timestamp =
m.group("time");
                e.mailId = m.group("id");
                e.message = m.group("msg");
                return e;
            }
            return null;
        }
```

```
                private File getAbsoluteFile(File
            root, String path) {
                    File file = new File(path);
                    if (file.isAbsolute())
                        return file;
                    if (root == null)
                        return null;
                    return new File(root, path);
                }
            }
```

```
impo
t
MFRC
522
        # from attendance import onScreen
        def readNfc():
            reading = True
            while reading:
                MIFAREReader = MFRC522.MFRC522()
                #while continue_reading:
                (status,TagType) =
        MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
                #if status == MIFAREReader.MI_OK:
                #    print("Card detected")
                (status,backData) = MIFAREReader.MFRC522_Anticoll()
                if status == MIFAREReader.MI_OK:
                    #print ("Card Number:
        "+str(backData[0])+","+str(backData[1])+","+str(backData[2])+","+st
        r(backData[3])+","+str(backData[4]))
                    MIFAREReader.AntennaOff()
                    reading=False
                    return
        str(backData[0])+str(backData[1])+str(backData[2])+str(backData[3])
        +str(backData[4])
```

Додаток 2. Лістинг файлу «attendance.py»

```
import
logging
        import sys
        import termios
        import time
        import tty
        # import RPi.GPIO as GPIO
        import thread
        # import display
        import sqlite
        import nfc
        # Enable debug logging into log
        DEBUG = True
        # Enable printing informations to std. output
        VERBOSE = True
        class Actions:
            incomming = 1
            outcomming = 2
            breakstart = 3
            breakend = 4
        if (DEBUG):
            logging.basicConfig(format='%(asctime)s %(message)s',
        filename='attendance.log', level=logging.DEBUG)
        def debug(message):
            logging.debug(message)
        def onScreen(message):
            if (VERBOSE):
                print(message)
        def read():
            cardId = nfc.readNfc()
            return cardId
        def readNfc(action):
            if (action == 55):  # 7 - Incomming
                # onScreen("Logging In...")
                # display.lcdWriteFirstLine("Prichod...")
```

```
            # display.lcdWriteSecondLine("Swipe your Card")
            cardId = read()
            logging.info("Incomming - %s", cardId)
            name = sqlite.insertReading(cardId, Actions.incomming)
            # display.lcdWriteSecondLine(name)
        if (action == 57):  # 9 - outcomming
            # onScreen("...")
            # display.lcdWriteFirstLine("Logging out...")
            # display.lcdWriteSecondLine("Swipe your Card")
            cardId = read()
            logging.info("Outcomming - %s", cardId)
            name = sqlite.insertReading(cardId, Actions.outcomming)
            # display.lcdWriteSecondLine(name)
        if (action == 49):  # 1 - break start
            # onScreen("Zacatek pauzy...")
            # display.lcdWriteFirstLine("Pauza zacatek...")
            # display.lcdWriteSecondLine("Swipe your Card")
            cardId = read()
            logging.info("Break start - %s", cardId)
            name = sqlite.insertReading(cardId, Actions.breakstart)
            # display.lcdWriteSecondLine(name)
        if (action == 51):  # 3 - break end
            # onScreen("Konec pauzy...")
            # display.lcdWriteFirstLine("Pauza konec...")
            # display.lcdWriteSecondLine("Swipe your Card")
            cardId = read()
            logging.info("Break end - %s", cardId)
            name = sqlite.insertReading(cardId, Actions.breakend)
            # display.lcdWriteSecondLine(name)
        if (action == 53):  # 5 - Deletion of last inserted action
            # onScreen("Delete the last entry...")
            # display.lcdWriteFirstLine("Deleting...")
            # display.lcdWriteSecondLine("")
            cardId = read()
            logging.info("Deleting last action - %s", cardId)
            (lastTime, lastAction) = sqlite.getLastReading(cardId) or
(None, None)
            if (lastTime == None or lastAction == None):
                # display.lcdWriteSecondLine("Unknown Event")
```

```
                logging.info("Action not found")
                time.sleep(1)
            # else:
            #     display.lcdWriteFirstLine("Delete Event?")
            #     if (lastAction == Actions.incomming):
            #         display.lcdWriteSecondLine("Check In")
            #     elif (lastAction == Actions.outcomming):
            #         display.lcdWriteSecondLine("Check Out")
            #     elif (lastAction == Actions.breakstart):
            #         display.lcdWriteSecondLine("Pauza zacatek")
            #     elif (lastAction == Actions.breakend):
            #         display.lcdWriteSecondLine("End of Pause?")
            #     a = getOneKey()
            #     if (a == 49):  # 1
            #         onScreen("Mazu")
            #         logging.info(" - Deleting action %s (cas: %s)",
lastAction, lastTime)
            #         sqlite.deleteLastReading(cardId)
            #         # display.lcdWriteSecondLine("Deleted!")
            #     else:
            #         onScreen("Not Deleted")
            #         logging.info(" - Deleting canceled")
                      # display.lcdWriteSecondLine("Not deleted!")
    # Sleep a little, so the information about last action on
display is readable by humans
    time.sleep(1)
# Backing up the input attributes, so we can change it for
reading single
# character without hitting enter  each time
fd = sys.stdin.fileno()
old_settings = termios.tcgetattr(fd)
def getOneKey():
    try:
        tty.setcbreak(sys.stdin.fileno())
        ch = sys.stdin.read(1)
        return ord(ch)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
displayTime = True
```

```
def printDateToDisplay():
    while True:
        # Display current time on display, until global variable
is set
        if displayTime != True:
            thread.exit()
        # display.lcdWriteFirstLine(time.strftime("%d.%m.
%H:%M:%S", time.localtime()))
        # onScreen(time.strftime("%d.%m.%Y %H:%M:%S",
time.localtime()))
        time.sleep(1)
# def main():
#     GPIO.cleanup()
#     try:
#         initGpio()
#         # display.init()
#         while True:
#             # display.lcdWriteSecondLine("Choose an action...")
#             global displayTime
#             displayTime=true
#             #Start new thread to show curent datetime on
display
#             # and wait for user input on keyboard
#             # thr = thread.start_new_thread(printDateToDisplay,
())
#             a = getOneKey()
#             # displayTime=False
#             if 47 < a < 58:
#                 readNfc(a)
#     except KeyboardInterrupt:
#         GPIO.cleanup()
#         pass
#     GPIO.cleanup()
def main():
    readNfc(55)
if __name__ == '__main__':
    debug("---------========== Starting session! ==========-----
-----")
    main()
```

Додаток 3. Скрипт для роботи з таблицями Google Sheets.

```
from
__future__
import
print_functio
n
                import httplib2
                import os
                from apiclient import discovery
                from oauth2client import client
                from oauth2client import tools
                from oauth2client.file import Storage
                try:
                    import argparse
                    flags =
                argparse.ArgumentParser(parents=[tools.argparser]).parse_ar
                gs()
                except ImportError:
                    flags = None
                # If modifying these scopes, delete your previously saved
                credentials
                # at ~/.credentials/sheets.googleapis.com-python-
                quickstart.json
                SCOPES =
                'https://www.googleapis.com/auth/spreadsheets.readonly'
                CLIENT_SECRET_FILE = 'client_secret.json'
                APPLICATION_NAME = 'Google Sheets API Python Quickstart'
                def get_credentials():
                    """Gets valid user credentials from storage.
                    If nothing has been stored, or if the stored
                credentials are invalid,
                    the OAuth2 flow is completed to obtain the new
                credentials.
                    Returns:
                        Credentials, the obtained credential.
```

```
    """
    home_dir = os.path.expanduser('~')
    credential_dir = os.path.join(home_dir, '.credentials')
    if not os.path.exists(credential_dir):
        os.makedirs(credential_dir)
    credential_path = os.path.join(credential_dir,
                                   'sheets.googleapis.com-
python-quickstart.json')
    store = Storage(credential_path)
    credentials = store.get()
    if not credentials or credentials.invalid:
        flow =
client.flow_from_clientsecrets(CLIENT_SECRET_FILE, SCOPES)
        flow.user_agent = APPLICATION_NAME
        if flags:
            credentials = tools.run_flow(flow, store,
flags)
        else: # Needed only for compatibility with Python
2.6
            credentials = tools.run(flow, store)
        print('Storing credentials to ' +
    credential_path)
    return credentials
def main():
    """Shows basic usage of the Sheets API.
    Creates a Sheets API service object and prints the
  names and majors of
    students in a sample spreadsheet:

https://docs.google.com/spreadsheets/d/1BxiMVs0XRA5nFMdKvBdB
ZjgmUUqptlbs74OgvE2upms/edit
    """
    credentials = get_credentials()
    http = credentials.authorize(httplib2.Http())
   discoveryUrl =
('https://sheets.googleapis.com/$discovery/rest?'
                    'version=v4')
  service = discovery.build('sheets', 'v4', http=http,
```

```
            discoveryServiceUrl=discoveryUrl)
    spreadsheetId =
'1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs74OgvE2upms'
    rangeName = 'Class Data!A2:E'
    result = service.spreadsheets().values().get(
        spreadsheetId=spreadsheetId, range=rangeName).execute()
    values = result.get('values', [])
    if not values:
        print('No data found.')
    else:
        print('Name, Major:')
        for row in values:
            # Print columns A and E, which correspond to
indices 0 and 4.
            print('%s, %s' % (row[0], row[4]))
if __name__ == '__main__':
    main()
```

Додаток 6. Dockerfile для виконаного проекту.

```
FROM  phusion/baseimage:0.9.20

RUN locale-gen en_US.UTF-8
ENV LANG="en_US.UTF-8" \
    LANGUAGE="en_US:en" \
    LC_ALL="en_US.UTF-8"

ENV JAVA_VER=8\
    JAVA_HOME=/usr/lib/jvm/java-8-oracle \
    ANT_VERSION=1.9.7 \
    MAVEN_VERSION=3.3.9

RUN echo 'deb http://ppa.launchpad.net/webupd8team/java/ubuntu
xenial main' >> /etc/apt/sources.list && \
    echo 'deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu
xenial main' >> /etc/apt/sources.list && \
    apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
C2518248EEA14886 && \
    add-apt-repository universe && \
    apt-get update

RUN echo oracle-java${JAVA_VER}-installer shared/accepted-oracle-
license-v1-1 select true | /usr/bin/debconf-set-selections && \
    apt-get install -y --force-yes --no-install-recommends \
        oracle-java${JAVA_VER}-installer oracle-java${JAVA_VER}-set-
default oracle-java8-unlimited-jce-policy && \
    rm -rf /var/cache/oracle-jdk-installer && \
    update-java-alternatives -s java-${JAVA_VER}-oracle && \
    echo "export JAVA_HOME=/usr/lib/jvm/java-${JAVA_VER}-oracle" >>
~/.bashrc && \
```

```
    echo "export JAVA_TOOL_OPTIONS=\"-Dfile.encoding=UTF-8\"" >>
~/.bashrc

RUN cd && \
    wget -q http://archive.apache.org/dist/ant/binaries/apache-ant-
${ANT_VERSION}-bin.tar.gz && \
    tar -xzf apache-ant-${ANT_VERSION}-bin.tar.gz && \
    mv apache-ant-${ANT_VERSION} /opt/ant && \
    rm apache-ant-${ANT_VERSION}-bin.tar.gz
ENV ANT_HOME=/opt/ant \
    PATH="${PATH}:/opt/ant/bin"

#RUN apt-get install -y maven
RUN cd && \
    wget -q http://apache.volia.net/maven/maven-
3/${MAVEN_VERSION}/binaries/apache-maven-${MAVEN_VERSION}-bin.tar.gz
&& \
    tar -xzf apache-maven-${MAVEN_VERSION}-bin.tar.gz && \
    mv apache-maven-${MAVEN_VERSION} /opt/maven && \
    rm apache-maven-${MAVEN_VERSION}-bin.tar.gz

ENV M2_HOME=/opt/maven
ENV M2="${M2_HOME}/bin"
ENV MAVEN_OPTS="-Xmx512m -XX:MaxPermSize=256m" \
    PATH="${PATH}:$M2"

RUN apt-get install -y python3-pip python3-click libxml2-dev
libxslt1-dev && \
    pip3 install lxml

RUN apt-get install -y openssh-client

RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```