Міністерство освіти і науки України

Національний технічний університет України «КПІ імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

**ЗВІТ**

про \_\_\_переддипломну\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ практику

(назва практики)

на \_\_\_\_ТОВ «Софтпозитив»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва підприємства, місто)

з «\_13\_» \_\_\_\_\_\_квітня\_\_\_\_\_\_\_\_2020 р. по «\_\_17\_» \_\_\_травня\_\_\_\_\_2020 р.

|  |  |
| --- | --- |
| Керівники практики:  від підприємства  \_\_Цапенко М.В.\_\_\_\_ \_\_\_\_\_\_\_\_\_\_  (прізвище, ініціали) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_  (дата)  від інституту  Шимкович Л.Л. \_\_\_\_\_\_\_\_\_\_  (прізвище, ініціали) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Захищено \_\_\_\_\_\_\_\_\_\_\_\_  (дата)  з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_ | Студент \_\_4\_\_ курсу, групи \_ІТ-61\_  \_\_\_Тимченко О.Ю.\_ \_\_\_\_\_\_\_\_\_\_  (прізвище, ініціали) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_  (дата) |

Київ 2020

**ЗМІСТ**

ВСТУП3

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ5

1.1 UniPing RS-4855

* 1. Ecoisme6

1.3 Smart Mac6

2. ТЕХНОЛОГІЇ І ПРИНЦИПИ ВИКОРИСТАНІ ДЛЯ РОЗРОБКИ8

2.1 Сервісно-орієнтована архітектура системи8

2.2 Принципи SOLID9

2.3 Хмарна платформа Microsoft Azure10

2.4 Платформа .NET Core10

2.5 Фреймворк ASP.NET Core10

2.6 Технологія безсерверних обчислень Azure Functions11

2.7 Azure computer vision API11

2.8 Робота з даними11

2.9 Фреймворк Angular 812

2.10 Мікроконтролер ESP32-Cam12

2.11 Живлення мікроконтролеру13

3. РЕАЛІЗАЦІЯ СИСТЕМИ14

ВИСНОВКИ16

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ17

**ВСТУП**

В сучасному світі, темп життя підвищується і людська увага стає одним із най важливіших ресурсів. Стрес як явище, негативно впливає на здатність концентруватись на дійсно важливих задачах які позитивно змінюють життя. Важливим аспектом зниження суттєвої кількості стресу це зменшення кількості рутинних справ шляхом їх автоматизації. Система що була розроблена покликана автоматизувати одну з таких задач - моніторинг засобів комунального обліку.

У процесі обслуговування певної нерухомості, невідворотно доводиться стикатися з необхідністю перевірки показань та стану встановлених засобів обліку. Зазвичай, такі засоби встановлені у не дуже зручних та доступних місцях, тому кожен акт перевірки є доволі трудозатратним та дратівлим, більш того такі операції є періодичними, тому необхідно пам’ятати про те що необхідно виконати цю операцію. Також, при необхідності проаналізувати рівень споживання того чи іншого ресурсу немає способу зробити це просто та інтуїтивно, доводиться збирати платіжні квитанції, або вести облік самостійно в електронному або паперовому виді. Такі рутинні задачі мають бути автоматизовані шляхом використання сучасних технологій, що в цілому покращить якість людського життя. Очевидно, що перекладання частини таких зобов’язань на програмний продукт має дуже велику кількість переваг, наприклад можливість перевіряти поточні показання засобів обліку у будь-якому куточку світу. В подальшому, система може бути розширена та інтегрована з сервісами для оплати комунальних послуг та інтегрована в систему розумного будинку як одна із його складових. Також одним із можливих покращень може бути розширення конфігурації мікроконтролеру за допомогою веб-сторінки або мобільного додатку.

Розробка дипломного проекту має на меті створення віддаленої, розподіленої, масштабованої та безкоштовної системи яка зможе містити у собі функціональність для моніторингу засобів обліку та подальшого збереження отриманих даних, їх відображення у текстовому та графічному вигляді. Задачею є аналіз існуючих варіацій таких систем на ринку, розробка архітектури її архітектура, розробка програмного коду для цієї системи та налагодження її безперервної та безвідмовної роботи.

Передбачається, що система призначена для використання кінцевими користувачами які мають на меті автоматизувати контроль і аналіз засобів обліку.

**1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ**

На ринку пропонується широка низка так званих «розумних» засобів обліку в купі з програмним забезпеченням сумісним з такими засобами. Функціонал які вони пропонують це, зазвичай, можливість віддаленого моніторингу, перегляд статистики на основі даних що були зібрані цими приладами, тощо. Для більш конкретного опису необхідно розглядати кожен продукт що є аналогом дипломного проекту окремо.

* 1. UniPing RS-485

З назви даного приладу для дистанційного моніторингу очевидно, що дані для взаємодії з передаються через інтерфейс RS-485. Дані передаються до спеціального приладу що виступає в ролі хаба, а вже сам хаб безпосередньо передає дані на веб-додаток до якого можна отримати доступ через мережу інтернет. Хаб може взаємодіяти з декількома приладами обліку.

Переваги:

* віддалений доступ до даних;
* можливість використовувати декілька приладів обліку;
* низька вартість.

Недоліки:

* незручне програмне забезпечення для доступу до даних;
* необхідна заміна штатних звичайних засобів обліку;
* автоматизація моніторингу можлива лише для електромереж;
* необхідне додаткове обладнання у вигляді хабу;
* складне налаштування системи.
  1. Ecoisme

Дана система моніторингу являє собою прилад що приєднується до входу в електромережу будинку, підключається до домашнього Wi-Fi і збирає дані в реальному часі. Зібрані та проаналізовані дані дають змогу побачити повну картину споживання електричної енергії можна за допомогою мобільного додатку.

Переваги:

* віддалений доступ до даних;
* зручний доступ до даних;
* можливість приєднання до Wi-Fi;
* не потрібно заміняти існуючі прилади обліку.

Недоліки:

* можуть бути складнощі при встановленні;
* можливість слідкувати лише за спожитою електроенергією;
* відсутність веб-версії додатку для перегляду даних;
* висока вартість.
  1. Smart Mac

Це сімейство розумних лічильників які здатні контролювати споживання як електроенергії, так і води, газу, тепла, вологість повітря, температуру, тощо. Зібрані дані зручно представлені в графічному вигляді у веб-додатку.

Переваги:

* віддалений доступ до даних;
* можливість слідкувати за будь-якими засобами обліку;
* зручний доступ до даних;
* відносно невелика ціна.

Недоліки:

* необхідність заміни існуючих приладів.

Для наочності, в таблиці 1.1 наведено всі переваги кожного із рішень що було описано вище.

Таблиця 1.1 – Порівняння систем для обліку

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Назва | Віддалений доступ до даних | Моніторинг різних комунікацій | Зручний доступ до даних | Ціна | Необхідність заміни існуючих | Бездротова мережа |
| UniPing RS-485 | + | - | - | + | + | - |
| Ecoisme | + | - | - | - | - | + |
| Smart Mac | + | + | + | - | + | + |

**2 ТЕХНОЛОГІЇ І ПРИНЦИПИ ВИКОРИСТАНІ ПРИ РОЗРОБЦІ**

2.1 Сервісно-орієнтована архітектура системи

SOA (Service oriented architecture) або сервісно-орієнтована архітектура це шаблон для проектування програмного забезпечення, ідея якого в використанні розподілених та слабко пов’язаних компонентів що можуть бути замінені, у всіх складових системи мають бути стандартизовані інтерфейси для взаємодії одним з одним за певними протоколами. Інтерфейси які використовують сервіси по своїй суті незалежні один від одного і чітко типізованими, завдяки цьому досягається достатній рівень гнучкості і модульності, адже будь який модуль системи може бути замінений якщо він реалізує заданий інтерфейс.

Необхідно перерахувати базові принципи які мають зберігатись при розробці системи по даному шаблону[1]:

* відсутність стану;
* слабка зв’язність;
* повторне використання;
* стандартизація;
* відкритість;
* модульність;
* гранулярність.

Майже усі перераховані принципи інтуїтивно зрозумілі та доповнюють одне одного. Так, слабка зв’язність має на увазі невелику кількість залежностей між різними складовими системами. Такий підхід дає можливість повторного використання сервісу іншими підсистемами. Також однією з переваг при невеликій кількості залежностей між сервісами це можливість легко замінити його іншим, але це неможливо при відсутності певних стандартів якими розробники керуються під час проектування і розроблення систем. Модульність означає що кожен сервіс реалізує певні модулі бізнес-логіки, вони є частиною бізнес процесу і можуть бути замінені чи використані повторно, кожен модуль має відповідати за одну функцію. Відсутність стану означає що сервіси не мають зберігати свій стан при роботі з клієнтом, такий підхід дозволяє не орієнтуватись на те що з клієнтом буде підтримуватись зв’язок довгий період часу. Відкритість – для того щоб повторно використати сервіс, він має бути зрозумілий і легко впізнаваний. При проектуванні системи, кожен сервіс має брати на себе певну функцію в цілому бізнес процесі, це і є гранулярністю сервісів.

2.2 Принципи SOLID

Принципи SOLID це абревіатура яка означає п’ять принципів яких необхідно притримуватись для написання якісного програмного забезпечення за допомогою об’єктно-орієнтовного програмування[2].

Буква S означає single responsibility principle (принцип єдиної відповідальності) – кожен компонент має відповідати лише за одну частину функціональності. Даний принцип має на меті превентивно прибрати дублювання коду, зміну уже існуючого коду, та зменшення складності подальшої підтримки и тестування.

Буква O означає open/closed principle (принцип відкритості/закритості) – кожен клас має бути закритим для змін, але відкритим для розширення. По суті, мається на увазі що розширення функціоналу класу має здійснюватися за допомогою успадкування, проте в такому випадку кодова база росте, а значить підтримка такого коду ускладнюється.

Буква L означає Liskov substitution principle (принцип підстановки Лісков) – має на увазі що будь-який об’єкт типу можна замінити на об’єкт нащадок даного типу і поведінка програми не зміниться.

Буква I означає interface segregation principle (принцип розділення інтерфейсів) – необхідно створювати багато вузько-спеціалізованих інтерфейсів на відміну від створення одного великого. Це дозволяє розділяти відповідальність по різним частинам програми.

Буква D означає dependency inversion principle (принцип інверсії залежностей) – модулі вищих рівнів мають залежати від абстракцій так само як і модулі нижчих рівнів. Абстракції не мають залежати від деталей.

2.3 Хмарна платформа Microsoft Azure

Microsoft Azure це одна з найпопулярніших на теперішній день хмарних платформ яка поступово завойовує ринок хмарних обчислень. Як і будь-яка інша хмарна платформа вона може виступати в якості хостинга для різноманітних додатків, а також надає доступ до великої кількості різних хмарних сервісів.

В цілому, платформа надає послуги IaaS[3] (інфраструктура як сервіс) та PaaS[4] (платформа як сервіс). В даній роботі використані PaaS послуги даної платформи для розгортання додатку та для зберігання і обробки даних.

2.4 Платформа .NET Core

.NET Core це безкоштовна платформа з відкритим кодом. Головною перевагою даної платформи є його підтримка компанією Microsoft та можливість запуску програмного забезпечення на його основі на будь-якій з трьох найпопулярніших операційних систем: Windows, Linux, MacOs.

2.5 Фреймворк ASP.NET Core

ASP.NET Core є новим фреймворком для розробки сучасних веб-додатків. Він працює поверх платформи .NET Core, завдяки чому веб-застосунки побудовані на його базі є крос-платформними. Цей фреймворк також з відкритим кодом, завдяки чому він активно розробляється спільнотою і відкритий до змін.

2.6 Технологія безсерверних обчислень Azure Functions

Azure Functions це подійно-орієнтована платформа безсерверних обчислень що надається платформою Azure. Головними особливостями є глибока інтеграція з іншими сервісами що надаються хмарною платформою, автоматичне та гнучке масштабування яке не потребує додаткового втручання розробника та модель розробки що базується на подіях, що є зручною можливостю для інтеграції с іншими сервісами. В цілому, використання цієї платформи дозволяє зняти з програміста відповідальність за налаштування інфраструктури для запуску програмного забезпечення.

2.7 Azure Computer vision API

Azure Computer Vision API – це API що надається хмарною платформою Azure та дозволяє використовувати технології комп’ютерного зору для розпізнавання зображень. Використання штучного інтелекту в сучасному світі є дуже потужним інструментом і такі сервіси значно полегшують розробку додатків. В даній роботі сервіс використовується для витягання тексту з зображень засобів обліку.

2.8 Робота з даними

Практично будь-який додаток має зберігати дані для подальшої роботи з цими даними. Загальний підхід до збереження чітких структурованих даних використовувати в якості сховища реляційну базу даних з відповідною СУБД . В даному випадку використана СУБД MSSQL та Entity Framework Core в якості ORM для роботи з базою даних в об’єктно орієнтованому стилі. Для зберігання статичних даних – зображень використане спеціальне хмарне сховище Azure Blob Storage. Його переваги це оптимізація для зберігання великих неструктурованих масивів даних, є дешевим і таким що добре масштабується.

2.9 Фреймворк Angular 8

Angular – це сучасний фреймворк та платформа для розробки ефективних та складних SPA (single page application) застосунків. Розробка ведеться мовою TypeScript що є надбудовою яка розширює можливості JavaScript, зокрема мова підтримує повноцінні класи та явне визначення типів.

2.10 Мікроконтролер ESP32-Cam

В якості мікроконтролеру було обрано відомий контролер з назвою ESP32[5], більш точно, його модифікацію ESP32-Cam що містить в собі роз’єм для камери який зображено на рисунку 1. Цей контролер є новою, покращеною версією контролеру esp8266. Він є більш потужним та має більше пам’яті. В якості камери було обрано OV2640. Головною особливістю даного мікроконтролера є встановлений Wi-Fi модуль, що дозволяє приєднатись до існуючої мережу, або створити нову.



Рисунок 1 – Мікроконтролер ESP32-Cam

2.11 Живлення мікроконтролеру

Одна із вимог до системи це автономність, тому необхідно мати автономне джерело живлення для мікроконтролеру. В якості такого будуть виступати 4 батарейки типу ААА в батарейному боксі. Така конфігурація дозволить живити контролер доволі тривалий проміжок час без заміни, в той же час, саме джерело живлення не займе багато місця.

**3 РЕАЛІЗАЦІЯ СИСТЕМИ**

На основі аналізу аналогічних існуючих систем та опису комплексної задачу було проведено декомпозиції на більш малі підзадачі, а усю функціональність було розбито між декількома підсистемами. Розглянемо схематичне зображення системи на рисунку 2.

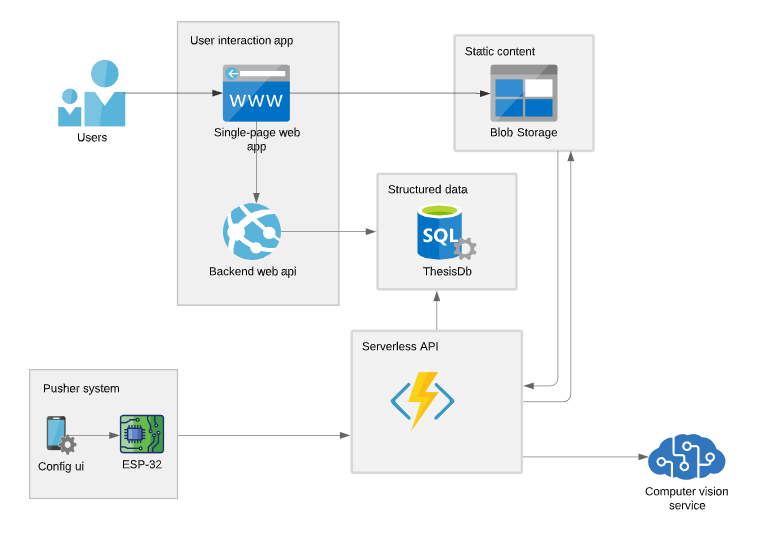


Рисунок 2 – Схематичне зображення системи

З рисунку стає очевидно що загальна система для моніторингу засобів обліку складається з трьох менших підсистем, а саме: мікроконтролер який безпосередньо встановлюється на засіб обліку – генератор фото, безсерверний backend побудований за допомогою Azure Functions – обробник фото, додаток для взаємодії з користувачами.

Всі взаємодії між сервісами відбувається за стандартизованими контрактами, по найпоширенішому протоколу HTTP. Також для більш зручної роботи зі сховищем статичного контенту також використовуються події яке воно генерує при взаємодії з ним.

Підсистема pusher system, яка містить в собі мікроконтролер відповідальна за взаємодію з засобом обліку на який встановлюється перший. Обраний мікроконтролер доволі потужний, а тому здатен запускати в собі повноцінний веб-сервер і загалом взаємодіяти через мережу. Веб-сервер піднятий на борту має слугувати приймачем конфігураційних запитів для налаштування поточної Wi-Fi мережі, інтервалу для відправки даних, тощо. За допомогою камери яка встановлена на мікроконтролері, він робить фотографію раз у визначений проміжок часу та відправляє її для подальшої обробки на безсерверний backend.

Обробник фотографій побудований на основі Azure Functions і відповідальний за отримання фотографій від генератора фото, їх збереження в сховище статичного контенту, реагування на події від сховища статичного контенту, розпізнавання тексту на фотографіях за допомогою стороннього сервісу комп’ютерного зору та збереження отриманих результатів в реляційну базу даних.  
 Додаток для взаємодії з користувачами складається з двох менших підсистем, а саме frontend частини та backend частини. Frontend відповідає за безпосереднє візуальне представлення даних користувачеві за допомогою графіків і таблиць, а backend обслуговує frontend і виконує функції авторизації користувача, взаємодії з базою даних та агрегації даних.

В рамках дипломного проекту виконано декілька графічних матеріалів:

* загальна функціональна схема системи;
* структурна схема підсистеми з мікрокотролером;
* ERD діаграма структури бази даних;
* діаграма розгортання системи.

**ВИСНОВКИ**

Проходження практики це кінцевий етап навчання, під час неї було узагальнені і вдосконалені навички отримані в університеті. Найбільш цінним є отриманий досвід по застосуванню набутих під час навчання в університеті теоретичних знань в реальних практичних завданнях.

Протягом переддипломної практики, мною були обрані та випробувані на практиці технології для розробки програмного продукту в рамках дипломного проекту, зокрема поглиблено знання про фреймворк asp.net core та хмарну платформу Microsoft Azure. Також були зібрані матеріали для написання пояснювальної записки, спроектовані та розроблені деякі частини дипломного проекту, усі окрім частини системи з мікроконтролером що відповідальна за взаємодію з засобами обліку.

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Порівняння типів архітектури систем сервісів / Петренко А. А. / Системні дослідження та інформаційні технології, 2015, № 4. – ст. 51
2. SOLID Principles made easy / Medium [Електронний ресурс]. – режим доступу: <https://medium.com/@dhkelmendi/solid-principles-made-easy-67b1246bcdf>
3. What is IaaS? [Електронний ресурс]. – режим доступу: <https://azure.microsoft.com/en-us/overview/what-is-iaas/>
4. What is PaaS? [Електронний ресурс]. – режим доступу: <https://azure.microsoft.com/en-us/overview/what-is-paas/>
5. ESP32 Overview [Електронний ресурс]. – режим доступу: <https://www.espressif.com/en/products/socs/esp32/overview>