МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Освітньо-професійна програма

«Комп'ютерна інженерія

Спеціальність 123

«Комп'ютерна інженерія»

Група: 2БКС-27

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

студента денного відділення
БКС 27.15.000 КРБ

КУРУ СЕРГІЯ ІВАНОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Освітньо-професійна програма
«Комп'ютерна інженерія»
Спеціальність 123
«Комп'ютерна інженерія»
Група БКС-27

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи бакалавра на тему: ***Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ***

Проектний матеріал складається з пояснювальної записки на 88 сторінках та мультимедійної презентації з 17 слайдів

Здобувач освіти _____ (Куру С.І.)

Керівник роботи _____ (Скорнякова О.В.)

Консультанти:

з охорони праці _____ (Чорновол Н.І.)

за дотриманням вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідуючий відділенням _____ (Скорнякова О.В.)

Захист « 20 » 06 2023 р. Протокол ДКК № 1

Оцінка ДКК 5 (відм)

Секретар ДКК _____

АНОТАЦІЯ

Даний проект має на меті розробку системи обліку відвідування занять, яка допоможе установі або організації з ефективним веденням обліку присутності студентів, викладачів або співробітників на заняттях. Система буде забезпечувати автоматизовану фіксацію даних про відвідування, а адміністратор зможе аналізувати ці дані і надавати звіти та статистику для подальшого використання.

В процесі проектування системи будуть враховані такі аспекти як:

Реєстрація користувачів: Система буде мати можливість реєструвати різні дані користувачів, такі як ПІБ, групу та список занять, і забезпечувати доступ до системи через індивідуальні облікові записи.

Фіксація відвідування: Система буде використовувати механізм для фіксації відвідування на заняттях. Вона складається з такого типу ідентифікації, як використання радіочастотної мітки (RFID).

Автоматизований облік: Зібрані дані про відвідування будуть автоматично оброблятися системою для подальшого аналізу. Це допоможе отримати інформацію про відвідування конкретних занять, кількість пропущених занять, загальну відвідуваність користувача тощо.

В результаті реалізації даного проекту буде створена ефективна система обліку відвідування занять, яка спростить процес ведення обліку, покращить точність і доступність інформації про відвідування, а також допоможе у прийнятті обґрунтованих рішень на основі отриманої статистики і аналізу даних.

Ключові терміни: система обліку, СКУД, RFID, радіочастотна ідентифікація.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"_____" _____ 20____ р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачу освіти *Куру Сергію Івановичу*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи ***Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ***

затверджена наказом по коледжу від *17 жовтня 2022 р. № 235-А2-ОД*

2. Термін здачі студентом кваліфікаційної роботи _____

3. Вихідні дані до роботи *Пристрій для радіочастотної ідентифікації RFID RC522. Одноплатний комп'ютер Raspberry Pi Model B. Веб-фреймворк Meteor. СУБД MongoDB. Python програми: reader.py, reciver.py, process.py*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1. Технологічний розділ. 2. Охорона праці. Висновки. Список використаних джерел. Додатки

5. Перелік графічного матеріалу (слайдів мультимедійної презентації)

Слайд 1 – Титульний. Слайд 2 – Вступ. Слайд 3 – Скуд – як основа. Слайд 4 – RFID – Сучасна технологія ідентифікації . Слайд 5 – Головний "мозок". Слайд 6 – IoT – Інтернет речей. Слайд 7 – Meteor (веб-фреймворк). Слайд 8 – СУБД MongoDB . Слайд 9 – Мова програмування. Слайд 10 – Мова розмітки. Слайд 11 – Скриптова мова програмування. Слайд 12 – Клієнтський фреймворк. Слайд 13 – Схема роботи пристрою. Слайд 14 – Вікно авторизації і таблиця студентів. Слайд 15 – Обробник даних. Слайд 16 – Результат. Слайд 17 – Дякую за увагу.

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що стосуються їх

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний	*Скорнякова О.В.*		
Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____.

Керівник роботи *Скорнякова О.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз предметної галузі*	*20.02.2023*	виконав
2.	*Аналіз технічного завдання та пошук літератури*	*01.03.2023*	виконав
1.	*Аналітичний огляд. Огляд існуючих рішень*	*20.03.2023*	виконав
4.	*Вивчення методів створення системи контролю доступу*	*10.04.2023*	виконав
5.	*Підбір апаратної та програмної бази*	*17.04.2023*	виконав
6.	*Створення системи обліку здобувачів знань*	*01.05.2023*	виконав
7.	*Внесення даних у систему*	*01.05.2023*	виконав
8.	*Розробка питань з охорони праці*	*15.05.2023*	виконав
9.	*Виконання графічної частини дипломного проекту*	*22.05.2023*	виконав
10.	*Підготовка до попереднього захисту, підготовка до*	*01.06.2023*	
.	*захисту*		виконав
11.	*Підготовка доповіді та презентації для захисту*	*10.06.2023*	виконав
12.	*Отримання рецензії, відповіді на зауваження*	*до 20.06.2023*	виконав
	рецензента		
13.	*Захист роботи*	*до 30.06.2023*	

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 27. 15 000 КРБ	Кваліфікаційна робота		
				бакалавра		
А4			БКС 27. 15 000 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 27. 12 000 КРБ			
Розробив		Куру С.І.			Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ	Літ.	Аркуш	Аркушів
Перевірив		Скорнякова О.В.				Д	4	
Н. Контр.		Петрашова В.І.				ОТФК ОНТУ, гр. БКС-27		
Затверд.		Кривченко Ю.В.						

ЗМІСТ

ВСТУП

В ВСП «Одеський технічний фаховий коледж ОНТУ», як і в усіх закладах освіти, відзначають відвідування студентами занять в закладах. Відвідування занять у вищих навчальних закладах є необхідною складовою ефективного навчання та досягнення академічних результатів. Присутність студентів на лекціях, семінарах та практичних заняттях має велике значення для їхнього особистого розвитку та успішного засвоєння навчального матеріалу. Присутність на заняттях також дозволяє студентам вчасно отримувати актуальну інформацію, оновлення в програмі навчання, а також важливі оголошення та зміни в графіку занять. Таким чином це сприяє активному долученню до навчального процессу. Але одна із таких, здавалося б, дрібниць, як відзнака відвідування занять - може сильно перешкоджати викладачам, відбираючи у них найцінніше. Час!

Це може займати дуже багато часу особливо на лекційних заняттях, оскільки лекція може проводиться одночасно для кількох груп. І, незважаючи на те, що в кабінеті може знаходитись декілька чималих груп, довжина пари лишається тією ж. Треба вирішувати цю проблему, тим паче, ми потрохи до цього рухаємось.

Тема кваліфікаційної роботи - проектування системи обліку відвідування занять здобувачами відділення комп’ютерних систем ВСП ОТФК ОНТУ.

Мета дипломного проєкту - створити систему, яка в теорії дасть змогу автоматизувати облік відвідувань і звільнить викладачів від рутини, з якої багатьом доводиться розпочинати заняття. А це насправді є чималою проблемою - незважаючи на те, що тривалість пари в післяшкільному навчальному закладі зазвичай вдвічі більша ніж у школі, кількість здобувачів знань так само збільшується і необхідно більше часу приділяти студентам, для того, щоб вони краще засвоїли лекцію.

Було поставлено завдання:

– спроектувати зчитувальний пристрій;

– створити сайт для обробки даних;

– створити програму для обробки інформації про відвідування і надсилання даних на сайт, який стежитиме за графіком відвідування й автоматично вноситиме в нього дані.

Електронна система обліку відвідувань не є новою розробкою - вона використовується на деяких підприємствах для обліку часу, в який приходять співробітники і т. д., але однак, мені не зустрічалася розробка, яка б враховувала відвідування в навчальних закладах, принаймні, розташованих на території нашої країни. Не варто забувати той факт, що якщо до такої думки прийшла одна людина - значить прийде й інша, питання лише в реалізації. Моя електронна система обліку призначена для автоматизації обліку відвідувань у навчальних закладах, зокрема, в нашому коледжі. Вона надсилає інформацію про відвідування на сайт коледжу за умови, що студент, прийшовши на заняття, відмітився за допомогою електронної картки студента на зчитувальному пристрої біля дверей кабінету.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Види систем обліку та сфера їх використання

Включення обліку відвідування занять є необхідною вимогою для будь-якого навчального закладу і має велике значення у роботі підприємств і фірм різного типу. Наприклад, в Індії було запроваджено облік робочого часу для чиновників на основі великих обсягів даних. Після шести місяців використання цієї системи було виявлено, що середньостатистичний чиновник працює на 20 хвилин більше щодня.

Згідно з дослідженням, опублікованим американським бюро статистики у 2017 році, виявлено, що шахрайство працівників щодо робочого часу (так зване "buddy punching"), яке включає ставлення позначок про прихід на роботу за іншого працівника, коштує американським роботодавцям понад 373 мільйони доларів США щороку. Це можливо в системах обліку робочого часу, де використовуються безконтактні картки, або коли хтось інший запускає комп'ютер відсутнього працівника і входить у його робочий профіль, якщо облік ґрунтується на часі роботи комп'ютера.

Існує багато способів автоматизації контролю відвідування занять студентів, робочого часу працівників або покупців у магазині. Розглянемо деякі з наявних видів автоматизації більш детально. Для цього розглянемо два способи: використання турнікету та систем контролю доступу (СКУД), встановлених на дверях.

Використання турнікету та СКУД є обов'язковим функціоналом практично всіх сучасних систем контролю доступу. Якщо розглянути різноманіття можливостей СКУД для обліку робочого часу, то можна побачити, що використання турнікету є одним з найкращих варіантів. Його унікальна особливість полягає у здатності відсікати проходження осіб поодинці. Зазвичай турнікет встановлюється біля входу до будівлі, навпроти поста охорони або чергових змін.

Рис.1.1 – Зовнішний вигляд турнікету

Використання турнікету в системі контролю доступу (СКУД) для обліку робочого часу потребує додаткового обладнання, без якого не обійтися. Це включає контролер, який керує турнікетом, комп'ютер та програмне забезпечення для управління системою, прокладання кабельних трас для передачі слабкострумових сигналів і блоки живлення.

Другий варіант використання СКУД для обліку відвідувань - це встановлення СКУД на дверях. В такому випадку двері повинні бути оснащені електронним замком, контролером СКУД та зчитувачами на вході та виході. Зчитувачі, подібно до використання турнікету, можуть бути різними - зчитувачі RFID, відбитків пальця, розпізнавання обличчя, райдужної оболонки або венозного малюнка. Якщо СКУД встановлюється не лише на вхідні двері будівлі, але й усередині, це дозволить більш точно враховувати час перебування у приміщенні, оскільки просто увійти до будівлі ще не означає досягти потрібного кабінету.

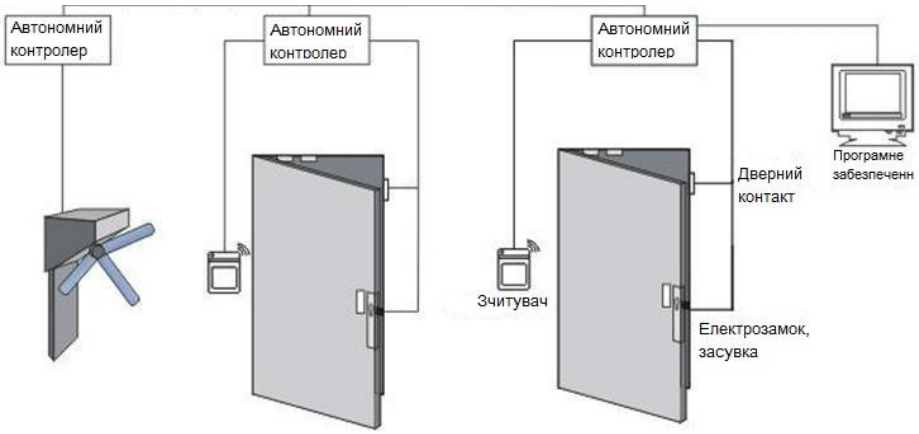


Рис. 1.2 – Схема мережевої СКУД

При використанні СКУД, встановленого на дверях, для обліку часу відвідування, існує одна особливість, про яку варто згадати з самого початку. Ця особливість називається "парним проходом" і означає, що якщо до дверей підійде група з трьох людей, система зарахує лише одну картку, хоча всі троє пройдуть. З урахуванням того, що люди зазвичай приходять і виходять з навчального закладу одночасно, така ситуація виникає досить часто.

Термінали обліку робочого часу представляють інший спосіб, який не зовсім підходить для умов студентського життя. Термінали обліку робочого часу (термінали ОРЧ) - це пристрої, що використовуються для фіксації часу приходу та відходу співробітників. Однак термінали для обліку часу не управляють жодними виконавчими пристроями, такими як турнікети або електронні замки. Їх робота полягає у простій логіці: перша ідентифікація - це прихід на роботу, а друга - покидання робочого місця.

Рис. 1.3 – Термінал ОРЧ

Більшість терміналів для обліку часу підтримують біометричну ідентифікацію, таку як розпізнавання відбитків пальців або облич, а також безконтактні карти доступу, смартфони та венозний малюнок. Деякі з них також підтримують мультифакторну автентифікацію.

Ці термінали вже мають вбудоване програмне забезпечення з готовими звітами щодо доступу співробітників. Завантажити та переглянути ці звіти можна через веб-сервер терміналу або через програмне забезпечення, встановлене на комп'ютері. Деякі пристрої також дозволяють експортувати звіти на флеш-накопичувач, який можна підключити до пристрою.

Для розпізнавання облич використовується програмне забезпечення, яке працює за такою схемою: на вхідній або вихідній точці встановлюється одна або кілька камер, в залежності від розміру прохідної. Камери розпізнають всіх, хто входить або виходить, і фіксують час приходу або виходу. За допомогою програмного забезпечення можна будувати звіти і отримувати інформацію про час, коли співробітники або студенти перебували на території закладу.

Контроль активності персоналу - це метод, спрямований на фіксацію активності персоналу під час перебування на робочому місці. Цей підхід вимагає виконання багатьох умов для ефективної роботи і, отже, не підходить для нашого випадку.

Існує також можливість використання мобільних додатків, які інтегруються з системою контролю доступу. За допомогою такого мобільного додатку на смартфоні з модулем NFC можна реєструвати час приходу та виходу. Це цікаве рішення, оскільки воно є частиною програмних комплексів відомих виробників контролерів СКУД. Для ідентифікації використовуються безконтактні карти формату MIFARE. Для зчитування інших форматів карток можна використовувати зчитувач RFID, підключений через роз'єм micro USB.

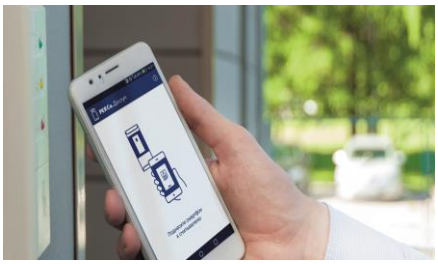


Рис. 1.4 – Мобільний термінал СКУД

Дані, які збираються мобільним додатком, передаються на сервер, на якому встановлене програмне забезпечення СКУД. Передача різної інформації між мобільним додатком та сервером СКУД може здійснюватися як через Wi-Fi, так і за допомогою мобільного інтернету. У випадку, якщо телефон знаходиться поза зоною покриття мережі, транзакції зберігаються в пам'яті додатка, і після відновлення зв'язку інформація передається на сервер.

Щодо використання простих мобільних додатків, вони є найдешевшим варіантом обліку робочого часу серед усіх розглянутих. Це пояснюється тим, що вони мають обмежений функціонал та можливості, але єдиним чинником, що обумовлює це, є їхня низька ціна або навіть безкоштовність. Зазвичай, такі мобільні додатки емулюють функції терміналів доступу на смартфоні і надають програмне забезпечення для обліку робочого часу.

Перейдемо до розробки проекту. У моєму проекті буде використовуватися модель, заснована на системі контролю доступу (СКУД). Тому розглянемо основні компоненти, які входять до складу СКУД.

1.2 Система контролю і управління доступом

Система контролю та управління доступом, СКУД (англ. Physical Access Control System, PACS) - сукупність програмно-апаратних технічних засобів контролю та засобів управління, що мають на меті обмеження та реєстрацію входу-виходу об'єктів (людей, транспорту) на заданій території через точки проходу.»: двері, ворота, КПП.

Рис. 1.5 – Види СКУД

Основна задача - управління доступом на задану територію (кого пускати, в який час і яку територію), включаючи:

– обмеження доступу на задану територію;

– ідентифікацію особи, яка має доступ на задану територію.

Додаткові задачі:

– облік робочого часу;

– розрахунок заробітної плати (при інтеграції із системами бухгалтерського обліку);

– ведення бази персоналу/відвідувачів;

– інтеграція із системою безпеки, наприклад:

– із системою відеоспостереження для поєднання архівів подій систем, передачі системі відеоспостереження сповіщень про необхідність стартувати запис, повернути камеру для запису наслідків зафіксованої підозрілої події;

– з системою охоронної сигналізації (СОС), наприклад, для обмеження доступу до приміщень, що стоять на охороні, або для автоматичного зняття та постановки приміщень на охорону.

– з системою пожежної сигналізації (СПС) для отримання інформації про стан пожежних сповіщувачів, автоматичне розблокування евакуаційних виходів та закриття протипожежних дверей у разі пожежної тривоги.

На особливо відповідальних об'єктах мережа пристроїв СКУД виконується фізично не пов'язаною з іншими інформаційними мережами.

Переваги СКУД.

Безпека. Система дає змогу налаштувати час доступу для кожного відвідувача чи співробітника в індивідуальному порядку. Стороння людина, яка не занесена до списку допущених, не зможе потрапити на територію об'єкта.

Контроль робочого дня. Система управління доступом дозволяє задавати та контролювати час присутності та відсутності співробітника на робочому місці. Крім цього, керівник зможе враховувати весь перероблений годинник, кількість часу, витраченого на перерви та обід, індивідуально для кожного працівника.

Економічність. Незважаючи на кількість функцій система не вимагає великих витрат електроенергії, при цьому має тривалий термін служби, до того ж допомагає заощадити на додатковій охороні.

Традиційно прийнято виділяти три групи систем:

Автономні системи. Такий пристрій є альтернативою дверним замкам, як правило, у комерційних приміщеннях. У систему вносяться коди карток доступу,

яким дозволено вхід. При збігу карти та запрограмованого коду в системі замок відмикається. Ці системи не вимагають підключення до комп'ютера і мають мінімальний набір функцій.

Автономні системи, порівняно з іншими варіантами, є більш економічними та простими у використанні, оскільки вони не вимагають прокладання великої кількості кабелів або використання спеціальних пристроїв для підключення до комп'ютера чи навіть самого комп'ютера. Однак, недоліком таких систем є обмежені можливості створення звітів, ведення обліку робочого часу, передачі та агрегування інформації про події, а також відсутність можливості дистанційного керування ними.

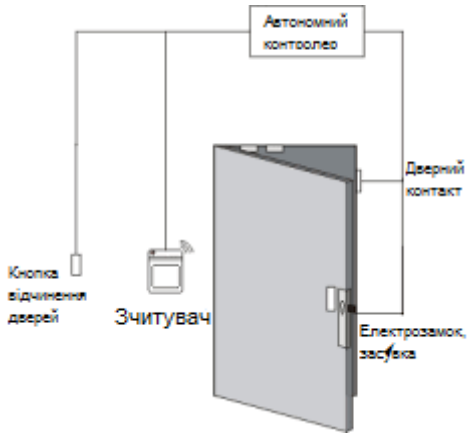


Рис. 1.6 – Схема автономної СКУД

При виборі автономної системи з високими вимогами безпеки рекомендується звернути увагу на наступне:

– Зчитувач повинен бути відокремлений від контролера, щоб дроти, якими можливе відкривання замка, були недоступні зовні.

– Контролер повинен мати резервне джерело живлення у разі відключення електроживлення.

– Переважно використовувати зчитувач у вандалозахищеному корпусі.

У складі автономної системи контролю доступу використовуються також електронні замки, що передають інформацію бездротовими каналами зв'язку: у двері встановлюється механічний замок з електронним керуванням і вбудованим зчитувачем. Замок по радіоканалу пов'язаний з хабом, який вже по дротах

обмінюється інформацією з робочою станцією, де встановлено програмне забезпечення.

Мережеві. Такі СКУД мають більше можливостей: вони можуть налаштовувати доступ до приміщення за розкладом, контролюють графік роботи та інтегруються з відеокамерами, охоронною та протипожежною системами. Мережеві системи з'єднані з комп'ютером та керуються дистанційно.

За допомогою мережевої системи контролю доступу можливо одночасно контролювати та керувати безліччю точок проходу відвідувачів та проїзду транспорту. Можливості нарощування мережевих СКУД залежать тільки від параметрів, закладених при проектуванні та виробництві таких систем.

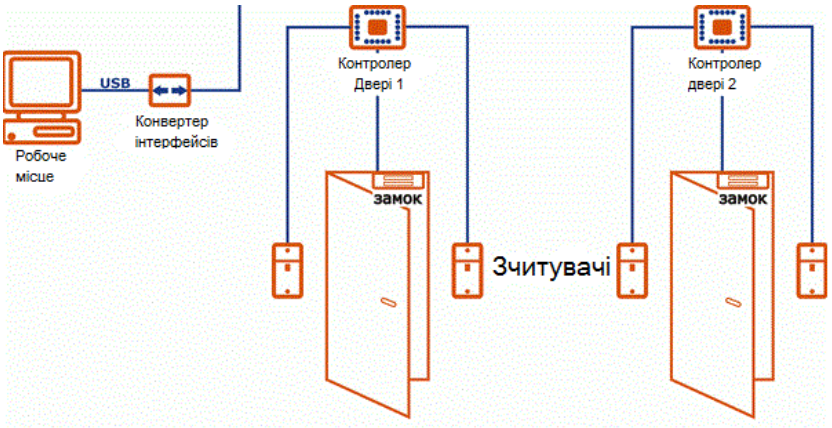


Рис. 1.7 – Схема мережевої СКУД

Підключення мережевих контролерів до єдиної мережі системи контролю та управління доступом (СКУД) здійснюється за допомогою інтерфейсів, які вбудовані в конкретну систему доступу під час її виробництва. Зазвичай обладнання СКУД підключається за допомогою спеціальних ліній зв'язку, які підтримують протоколи, власні виробниками мережевих систем. Також широко використовується включення елементів СКУД до існуючої мережі Ethernet підприємства, якщо обладнання системи доступу підтримує таку можливість.

У процесі проектування мережевої системи контролю та управління доступом враховуються всі деталі, щоб вибрати відповідне обладнання та створити основу майбутньої системи, яка має необхідні характеристики та можливості для конкретного користувача мережевого СКУД.

Біометричні системи. Ці системи контролю та управління доступом мають індивідуальну спрямованість: унікальний код кожного працівника. У програму заносяться відбитки пальців або малюнок райдужної оболонки ока. Такий підхід забезпечує більш високий рівень безпеки та надає більш повну інформацію щодо кожного співробітника кожного відділу. За допомогою біометричної системи є можливість вести журнал відвідувань, контролювати час, витрачений працівником на перерви та відрядження.

Рис. 1.8 – Схема біометричної СКУД

Можливості налаштування біометричної системи контролю та управління доступом (СКУД) є індивідуальними і різноманітними, але можуть впливати на рівень захищеності системи. Наприклад, налаштування максимальної швидкості обробки даних може скоротити час очікування проходу і збільшити пропускну здатність, але при цьому може знизити рівень захисту від несанкціонованого проникнення (збільшити ризик помилки або фальсифікації даних). В ситуаціях, коли потрібна висока пропускна здатність, але необхідно зберігати високий рівень захисту системи СКУД, застосовуються додаткові методи перевірки, такі як використання карт, паролів і т.д. Поєднання цих методів з біометричним скануванням практично усуває можливість помилки.

Для реалізації моєї системи контролю та управління доступом (СКУД), необхідно вибрати компоненти, на основі яких буде створена конструкція RFID-зчитувача. RFID - це технологія взаємодії з радіомітками, які можуть бути вбудовані в брелоки, пластикові картки та інші пристрої.

1.3 Вибір інструментів та технічних рішень для проектування

Існує багато різних видів і технологій ідентифікації, наприклад, смарт-картки, магнітні картки тощо. Ми ж вирішили зупинитися на технології Технологія радіочастотної ідентифікації Radio Frequency Identification - RFID.

RFID - це сучасна технологія ідентифікації, що надає істотно більше можливостей порівняно з традиційними системами маркування. Мною було обрано пристрій моделі RFID RC522 13.56MHz (рис. 1.9)

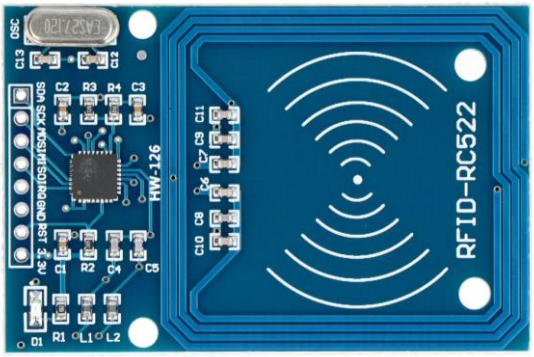


Рис. 1.9 – Зобряження RFID RC522

Технічні характеристики модуля.

- Напруга живлення: 3.3V;
- Споживаний струм: 13-26mA;
- Робоча частота: 13.56MHz;
- Дальність зчитування: 0 - 60 мм;
- Інтерфейс: SPI;
- Швидкість передачі: максимальна 10Мбіт/с;
- Розмір: 40мм х 60мм.

Інтерфейси і призначення виводів.

Мікросхема MFRC522 підтримує інтерфейси SPI, UART та I2C (див. рисунок 1.10). Вибір інтерфейсу здійснюється встановленням логічних рівнів на певних виводах мікросхеми. На цьому модулі обрано інтерфейс SPI.

Рис. 1.10 – Призначення виводів RC522

Призначення виводів інтерфейсу SPI:

- SDA - вибір веденого;
- SCK - сигнал синхронізації;
- MOSI - передача від master до slave;
- MISO - передача від slave до master;
- RST - вивід для скидання;
- IRQ - вивід переривання;
- GND - земля;
- Vcc - живлення 3.3 В.

Сигнал скидання RST - це сигнал, що надходить від цифрового виходу контролера. При надходженні сигналу LOW відбувається перезавантаження зчитувача. Також рідер установкою на RST низького рівня повідомляє, що перебуває в режимі сну, для виведення модуля з режиму сну необхідно подати на цей вивід сигнал HIGH.

RFID по суті - спосіб автоматичної ідентифікації об'єктів, у якому за допомогою радіосигналів зчитуються або записуються дані, що зберігаються в так званих транспондерах, або RFID-мітках.

Будь-яка RFID-система складається зі зчитувального пристрою (зчитувач, рідер або інтеррогатор) і транспондера (він же RFID-мітка, іноді також застосовується термін RFID-тег).

За дальністю зчитування RFID-системи можна поділити на системи:

- ближньої ідентифікації (зчитування проводиться на відстані до 20 см);
- ідентифікації середньої дальності (від 20 см до 5 м);
- дальньої ідентифікації (від 5 м до 300 м)

Більшість RFID-міток складаються з двох основних компонентів. Перша - інтегральна схема (IC), яка відповідає за зберігання та обробку інформації, модулювання та демодулювання радіочастотного (RF) сигналу та виконання інших функцій. Друга - антена, яка використовується для прийому і передачі сигналу.

Зі зростанням використання RFID-міток у повсякденній житті виникають певні проблеми. Наприклад, споживачі, які не мають зчитувачів, не завжди можуть виявити мітки, які були прикріплені до товарів на етапі виробництва та упаковки, та позбутися їх. Хоча на етапі продажу такі мітки зазвичай знищуються, сам факт їх наявності може викликати побоювання у правозахисних та релігійних організацій.

Розроблені застосування RFID, такі як безконтактні картки для систем контролю та управління доступом, системи дальньої ідентифікації та платіжні системи, набувають все більшої популярності разом із розвитком інтернет-послуг.

Класифікація RFID-міток.

Існує кілька способів систематизації RFID-міток і систем:

- за робочою частотою
- за джерелом живлення
- за типом пам'яті
- за виконанням

Розглянемо розподіл RFID міток залежно від джерела живлення.

Активні RFID мітки мають власний джерело живлення і не залежать від енергії зчитувача. Це дозволяє їм зчитуватися на великому відстані, а також мати більші розміри та додаткову електроніку. Однак такі мітки є дорожчими і мають обмежену годину роботи батареї. Активні мітки зазвичай є надійнішими та забезпечують найвищу точність зчитування на максимальному відстані. Такі мітки можуть генерувати сильніший вихідний сигнал, що дозволяє їх застосовувати у більш агресивних для радіочастотного сигналу средах. Багато активних міток можуть передавати сигнал на відстань у сотнях метрів протягом 10 років життя батареї. Деякі з них можуть мати вбудовані сенсори для моніторингу температури швидкопсувних товарів.

Активні мітки зазвичай мають значно більший радіус зчитування (до 300 м) та більший обсяг пам'яті порівняно з пасивними мітками, що дозволяє зберігати більше інформації для передачі зчитувачу.

Пасивні RFID мітки не мають вбудованого джерела живлення. Електричний струм, індукований в антені від електромагнітного сигналу зчитувача, забезпечує достатню енергію для роботи кремнієвого КМОП-чіпа, що знаходиться в мітці, та передачі відповідного сигналу.

Пасивні мітки низької частоти (НЧ) та високої частоти (ВЧ) передають сигнал шляхом модуляції відбитого сигналу несучої частоти. Антена зчитувача випромінює сигнал несучої частоти і отримує модульований сигнал, який відбивається від мітки. Пасивні мітки ВЧ також передають сигнал шляхом модуляції навантаження на несучій частоті. Кожна мітка має унікальний ідентифікаційний номер. Пасивні мітки можуть мати енергонезалежну пам'ять типу EEPROM, яка може бути перезаписана. Дальність дії пасивних міток становить 1-200 см для ВЧ міток та 1-10 метрів для НЧ та ВЧ міток.

Комерційні реалізації низькочастотних RFID міток можуть бути вбудовані в стікери або імплантовані під шкіру.

Напівпасивні. Напівпасивні мітки, також відомі як напівактивні RFID-мітки, подібні до пасивних міток, але вони мають вбудовану батарею, яка живить чіп.

Завдяки цьому, дальність дії цих міток залежить від чутливості зчитувача і вони можуть працювати на більшій відстані з кращими характеристиками.

За типом пам’яті мітки поділяються на:

RO (англ. Read Only) - дані можуть бути записані тільки один раз, зазвичай під час виготовлення. Такі мітки використовуються переважно для ідентифікації, оскільки неможливо записати нову інформацію в них, і вони майже не можуть бути підроблені.

WORM (англ. Write Once Read Many) - крім унікального ідентифікатора, такі мітки мають блок пам'яті, який можна записати лише один раз, але пізніше можна багаторазово читати.

RW (англ. Read and Write) - такі мітки містять ідентифікатор і блок пам'яті, який можна читати та записувати інформацію. Дані в них можуть бути перезаписані багаторазово.

За робочою частотою існують діапазони роботи RFID міток:

- Мітки діапазону LF (125-134 кГц)
- Мітки діапазону HF (13,56 МГц)
- Мітки діапазону UHF (860-960 МГц)
- Радіочастотні UHF-мітки ближнього поля

Види зчитувачів.

Стаціонарні. Стаціонарні зчитувачі мають фіксоване положення на стінах, дверях або на рухомих складських пристроях, таких як штабелятори та навантажувачі. Вони можуть бути встановлені як замок, вбудовані в стіл або закріплені поруч з конвеєром, що проходить виробами.

Порівняно з портативними зчитувачами, статичні зчитувачі зазвичай мають більшу зону зчитування та потужність і здатні одночасно обробляти дані з кількох десятків міток. Статичні зчитувачі підключаються до програмованих логічних контролерів (ПЛК), інтегруються в системи розподіленого керування (DCS) або підключаються до персональних комп'ютерів (ПК). Завдання таких зчитувачів

полягає у поетапному відстеженні переміщень маркованих об'єктів у реальному часі або ідентифікації положення маркованих предметів у просторі.

Мобільні. Цей тип зчитувачів має обмежену дальність дії порівняно з іншими і часто не підтримує постійне підключення до програми контролю та обліку. Переносні зчитувачі обладнані внутрішньою пам'яттю, в яку зберігаються дані, прочитані з міток (пізніше цю інформацію можна передати на комп'ютер), і, так само як стаціонарні зчитувачі, можуть записувати дані на мітки (наприклад, інформацію про проведений контроль).

Залежно від частотного діапазону мітки, дистанція стійкого зчитування і запису даних у них буде різною.

Переваги радіочастотної ідентифікації:

– Можливість перезапису. Дані RFID-мітки можуть перезаписуватися і доповнюватися багато разів, тоді як дані на штрих-коді не можуть бути змінені - вони записуються одразу під час друку.

– Відсутність необхідності в прямій видимості. RFID-зчитувачу не потрібна пряма видимість мітки, щоб зчитати її дані. Взаємна орієнтація мітки і зчитувача часто не грає ролі. Мітки можуть читатися через упаковку, що робить можливим їхнє приховане розміщення. Для читання даних мітці достатньо хоча б ненадовго потрапити в зону реєстрації, переміщаючись, зокрема, і на досить великій швидкості. Навпаки, пристрою зчитування штрих-коду завжди необхідна пряма видимість штрих-коду для його читання.

– Більша відстань читання. RFID-мітка може зчитуватися на значно більшій відстані, ніж штрих-код. Залежно від моделі мітки і зчитувача радіус зчитування може становити до кількох сотень метрів. Водночас подібні відстані потрібні не завжди.

– Більший обсяг зберігання даних. RFID-мітка може зберігати значно більше інформації, ніж штрих-код.

– Підтримка читання декількох міток. Промислові зчитувачі можуть одночасно зчитувати безліч (понад тисячу) RFID-міток за секунду,

використовуючи так звану антиколізійну функцію. Пристрій зчитування штрих-коду може одноразово сканувати тільки один штрих-код.

– Зчитування даних мітки за будь-якого її розташування. З метою забезпечення автоматичного зчитування штрихового коду комітети зі стандартів (зокрема EAN International) розробили правила розміщення штрих-міток на товарній і транспортній упаковці. До радіочастотних міток ці вимоги не стосуються. Єдина умова - знаходження мітки в зоні дії зчитувача.

– Стійкість до впливу навколишнього середовища. Існують RFID-мітки, що мають підвищену міцність і опірність жорстким умовам робочого середовища, а штрих-код легко пошкоджується (наприклад, вологою або забрудненням). У тих сферах застосування, де один і той самий об'єкт може використовуватися необмежену кількість разів (наприклад, під час ідентифікації контейнерів або зворотної тари), радіочастотна мітка виявляється більш прийнятним засобом ідентифікації, оскільки її не потрібно розміщувати на зовнішній стороні пакування. Пасивні RFID-мітки мають практично необмежений термін експлуатації.

– Багатоцільове використання. RFID-мітка може використовуватися для виконання інших завдань, крім функції носія даних. Штрих-код же не програмується і є лише засобом зберігання даних.

– Високий ступінь безпеки. Унікальне незмінне число-ідентифікатор, що присвоюється мітці під час виробництва, гарантує високий ступінь захисту міток від підробки. Також дані на мітці можуть бути зашифровані. Радіочастотна мітка має можливість закрити паролем операції запису і зчитування даних, а також зашифрувати їх передачу. В одній мітці можна одночасно зберігати відкриті та закриті дані.

Недоліки радіочастотної ідентифікації:

– працездатність мітки втрачається при частковому механічному пошкодженні.

– вартість системи вища за вартість системи обліку, заснованої на штрих-кодах.

– простота самостійного виготовлення. Штрих-код можна надрукувати на будь-якому принтері.

– схильність до перешкод у вигляді електромагнітних полів.

– недовіра користувачів, можливості використання її для збору інформації про людей.

– встановлена технічна база для зчитування штрих-кодів істотно перевершує за обсягом рішення на основі Rfid.

– недостатня відкритість вироблених стандартів.

У моєму пристрої, в якості "головного мозку", використовується одноплатний комп'ютер під назвою Raspberry Pi.

Raspberry Pi є комп'ютером, який має розмір, подібний до банківської картки. Спочатку він був розроблений як доступна система для навчання інформатиці, але з часом отримав широке застосування та став популярним. Raspberry Pi розробляється британською компанією Raspberry Pi Foundation, очолюваною Ебеном Аптоном. За останніми даними, станом на кінець 2019 року, було продано понад 30 мільйонів пристроїв Raspberry Pi.

Ідея створення доступного комп'ютера виникла в 2006 році у групи співробітників, до якої входили Ебен Аптон, Роб Маллінс, Джек Ланг і Алан Майкрофт. Вони створили кілька прототипів, після чого до них приєднався Девід Бребен. Разом вони заснували Raspberry Pi Foundation і розпочали роботу над комп'ютером.

У травні 2011 року Бребен представив перший концепт Raspberry Pi, який мав розмір USB-флеш-накопичувача. Наприкінці липня того ж року була завершена і відправлена виробництву альфа-версія плати. Вже 12 серпня Raspberry Pi Foundation отримала першу партію пристроїв. Альфа-версія комп'ютера містила тестові функції і дорогі деталі, які в подальшому були вилучені з фінальної версії. Крім того, фінальна версія плати була меншого розміру (20% менше) і складалася з чотирьох шарів, а не з шести.

28 лютого 2017 року розробники представили Raspberry Pi Zero W, який мав основною відмінністю наявність Wi-Fi і Bluetooth. 14 березня 2018 року, в день числа Пі, був випущений Raspberry Pi 3B+, який мав потужніший процесор, 1 Гбіт/с Ethernet, двосмуговий Wi-Fi і Bluetooth 4.2. У червні 2019 року була представлена модель Raspberry Pi 4B з новим чотирьохядерним процесором ARM Cortex-A72 1,5 ГГц. Початково вона була доступна в трьох варіантах з 1, 2 або 4 ГБ оперативної пам'яті, а пізніше була випущена версія з 8 ГБ ОЗП. Raspberry Pi 4B має повношвидкісний 1 Гбіт/с Ethernet, Bluetooth 5.0, два порти USB 3.0, два порти micro HDMI для підключення моніторів та використовує графічний процесор VideoCore VI (OpenGL ES 3.x) і апаратний декодер 4K для HEVC-відео.

Більшість моделей Raspberry Pi продаються у вигляді готових плат розміром приблизно з банківську картку (за винятком моделей A і A+, Zero і Zero W, які мають інший форм-фактор). У стандартний комплект поставки входить лише сам міні-комп'ютер, а корпус, блок живлення та карту пам'яті потрібно придбати окремо.

Рис. 1.11 – Зовнішній вигляд плати Raspberry Pi 4 Model B

У листопаді 2020 року Raspberry Pi Foundation випустила цікавий гаджет під назвою Raspberry Pi 400, який представляє собою одноплатний комп'ютер, вбудований у корпус клавіатури. Спочатку його було доступно лише у рожево-білому кольорі. Головний компонент - процесор Broadcom BCM2711 з чотирма ядрами Cortex-A72 (ARM v8), що працюють на частоті 1,8 ГГц. Обсяг оперативної пам'яті становить 4 ГБ, а вбудованої пам'яті в пристрої немає - замість цього

користувач повинен використовувати microSD карту. Raspberry Pi 400 має вбудовану підтримку бездротового зв'язку Wi-Fi 802.11b/g/n/ac (2,4 і 5 ГГц), а також Bluetooth 5.0. Порти гаджета розташовані на задній панелі і включають 2 × USB 3.0, USB 2.0, 2 × micro HDMI, Gigabit Ethernet, GPIO і слот для microSD. Живлення пристрою здійснюється через роз'єм USB-C. Початкова комплектація Raspberry Pi 400 (без миші, блоку живлення, microSD карти, HDMI кабелю і керівництва користувача) оцінюється в 70 доларів, а повна комплектація - в 100 доларів.

Рис. 1.12 – Зовнішний вигляд Raspberry Pi 400

Звичайно, для такого апаратного комплексу було необхідно розробити спеціальне програмне забезпечення або серйозно модифікувати вже існуюче. Давайте розглянемо наявні операційні системи для пристрою.

Raspberry Pi переважно працює на операційних системах, які базуються на ядрі Linux. Оскільки ARM11 заснований на 6-ій версії ARM, не всі версії Linux підтримують його. Для встановлення операційних систем існує інструмент NOOBS.

Також можна встановити Windows 10 IoT. Навіть можна придбати Raspberry Pi з ліцензійною версією Windows 10 IoT за 50 доларів. Windows 10 IoT (Internet of Things) є операційною системою, розробленою Microsoft для вбудованих систем та пристроїв Інтернету речей. Вона дозволяє розробникам створювати програмне забезпечення для різних пристроїв, таких як смарт-термометри, системи контролю доступу, домашні автоматизації та інші. Windows 10 IoT має три версії: Core, Enterprise та Mobile. Вона підтримує різні мови програмування, включаючи C++, C# та Python, що дозволяє розробникам легко створювати програми для різних пристроїв.

Офіційно підтримувані операційні системи:

– Raspberry Pi OS - рекомендується для всіх тих, хто тільки починає знайомитися з Raspberry Pi.

– Pidora - Fedora для Raspberry Pi (проєкт занедбаний, оскільки наразі Fedora офіційно підтримує Raspberry Pi).

– OpenELEC медіапрогравач Kodi з відкритим вихідним кодом на базі Linux.

– OSMC (проект Open Source Media Center - раніше відомий як Raspbmc) медіапрогравач з відкритим вихідним кодом на базі Kodi Media Center і Debian GNU/Linux.

– RISC OS - "рідна" ОС для RISC-процесорів (до яких належать процесори ARM).

– підтримка Windows 10 IoT для Raspberry Pi.

– Також існує купа варіантів неофіційного програмного забезпечення, яке розробляється та супроводжується силами ентузіастів і розробників.

Для встановлення операційної системи використовується інструмент NOOBS. Також можна завантажити образ операційної системи і розгорнути його на SD-картку.

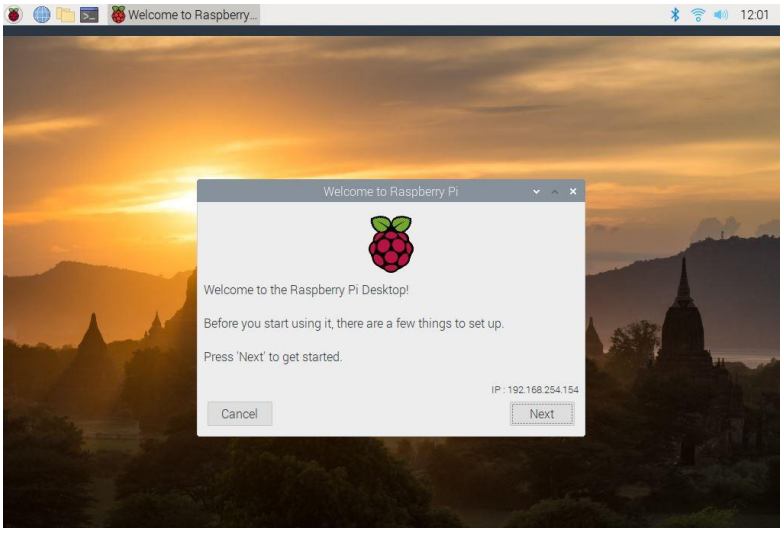


Рис. 1.13 – Стартовий екран Raspberry Pi OS

Для розробки проекту було обрано Raspberry Pi Model “B”. Це найдешевший модель із існуючих, яка містить у собі усі необхідні компоненти.

Рис. 1.14 – Зовнішній вигляд плати Raspberry Pi Model B

Raspberry Pi Model B є одним з найпопулярніших одноплатних комп'ютерів на ринку. Він має компактний розмір і може бути використаний для різних проектів, включаючи домашню автоматизацію, Інтернет речей, медіа-центри та багато іншого.

Raspberry Pi Model B обладнаний процесором ARM Cortex-A53 з тактовою частотою 1,2 ГГц, 1 ГБ оперативної пам'яті і підтримкою бездротових технологій Wi-Fi та Bluetooth. У нього є 4 USB-порти, HDMI-вихід для підключення до монітора або телевізора, Ethernet-порт для підключення до Інтернету і слот для карт пам'яті microSD.

Raspberry Pi Model B працює на операційній системі Raspbian, яка базується на Debian Linux. Вона володіє вбудованою підтримкою Python, що робить її ідеальною для розробки програмного забезпечення для Інтернету речей та інших проектів.

Загалом, Raspberry Pi Model B є потужним та доступним одноплатним комп'ютером, який може бути використаний для різних проектів. Його компактний розмір та підтримка бездротових технологій роблять його ідеальним для проектів Інтернету речей та домашньої автоматизації.

Одне з завдань полягало в зберіганні та редагуванні даних. Ми вирішили використати зручний і простий спосіб - внутрішньої мережевої сайт коледжу, який завжди доступний з будь-якого кабінету і не вимагає установки або оновлення програмного забезпечення. Для розробки сайту ми використовували HTML для розмітки, JavaScript для роботи з кнопками та вкладками, CSS для

оформлення, а також засоби Bootstrap, які значно полегшили роботу. Для розробки backend частини сайту ми використали веб-фреймворк Meteor. Звичайно, існують різні мови та можливості, такі як PHP і Python. Meteor - це веб-платформа на мові JavaScript, яка призначена для розробки реального часу веб-додатків. Для зв'язку з сучасними браузерами ми використовували протокол Distributed Data Protocol (DDP), який підтримується за допомогою WebSocket або, в разі відсутності підтримки WebSocket і DDP, - AJAX.

Meteor - це повноцінний веб-фреймворк для розробки веб-додатків на JavaScript. Він використовує Node.js на серверній стороні та MongoDB як базу даних. За допомогою Meteor розробники можуть створювати додатки, які працюють як на клієнтській, так і на серверній стороні, забезпечуючи швидку розробку та високу продуктивність. Код Meteor працює поверх Node.js, хоча він не дотримується асинхронної моделі, що характерна для Node.js, що може ускладнити інтеграцію з Node.js додатками Meteor.

Однією з ключових особливостей Meteor є його реактивність. Це означає, що коли дані змінюються на сервері, вони автоматично оновлюються на клієнтському пристрої без необхідності перезавантаження сторінки. Це дозволяє створювати більш інтерактивні та користувацькі дружні додатки.

Meteor також має вбудовану підтримку React і Angular, двох популярних фреймворків для розробки інтерфейсу користувача. Це дозволяє розробникам використовувати найкращі практики та інструменти для розробки інтерфейсу користувача.

Meteor також надає підтримку пакетів, що дозволяє розробникам легко додавати функціональність до своїх додатків. Це може бути все, від аутентифікації користувачів до інтеграції з соціальними мережами.

Однією з найбільших переваг Meteor є його простота використання та навчання. Він має документацію та підтримку спільноти, яка допомагає новачкам швидко освоїти фреймворк та почати розробку своїх додатків.

Одна з найважливіших особливостей платформи полягає в тому, що вона дає змогу використовувати один і той самий код як на серверній, так і на клієнтській стороні. Зазвичай передаються дані між сервером і клієнтом, а не HTML-код.

Meteor - це потужний та простий у використанні фреймворк для розробки веб-додатків, який дозволяє швидко створювати реактивні та інтерактивні додатки на JavaScript.

Оскільки Meteor використовує базу даних MongoDB, ми будемо використовувати її, оскільки вона відповідає всім нашим вимогам.

MongoDB - це документоорієнтована система управління базами даних, яка не вимагає опису схеми таблиць. Вона є одним з класичних прикладів NoSQL-систем і використовує JSON-подібні документи та схему бази даних. MongoDB написана мовою C++. Вона застосовується у веб-розробці, зокрема в рамках JavaScript-орієнтованого стеку MEAN.

Можливості СУБД.

Система надає можливість для виконання ad-hoc-запитів, які дозволяють користувачеві отримувати конкретні поля документів за допомогою JavaScript-функцій. Можливий пошук за регулярними виразами та налаштування запиту на повернення випадкового набору результатів. Також є підтримка індексів.

Система може працювати з набором реплік, де дані мають бути скопійовані на різних вузлах. Кожна репліка може виступати в ролі основної або допоміжної в будь-який момент. Операції запису та читання зазвичай виконуються з основною реплікою, а допоміжні репліки підтримують актуальні копії даних. У разі відмови основної репліки, набір реплік обирає нову основну репліку. Допоміжні репліки можуть використовуватись для операцій читання.

Система може масштабуватись горизонтально шляхом розподілу об'єктів бази даних на різні вузли кластера. Ключ сегментування визначає, як дані розподіляються за вузлами. Це забезпечує балансування навантаження.

Крім того, систему можна використовувати як файлове сховище з балансуванням навантаження та реплікацією даних. Grid File System дозволяє зберігати та обробляти файли шляхом розбиття їх на частини і зберігання кожної частини як окремого документа.

Система також підтримує парадигму MapReduce для агрегації даних. Існує аналог SQL-виразу GROUP BY та оператори агрегації, які можна ланцюжити подібно до Unix-конвеєрів. JavaScript використовується у запитах та функціях агрегації.

Система підтримує колекції з фіксованим розміром, які зберігають порядок вставки і можуть працювати як кільцевий буфер після досягнення заданого розміру.

У версії 4.0 була додана підтримка транзакцій, що задовольняють вимогам ACID.

Існують офіційні драйвери для багатьох мов програмування, а також багато неофіційних драйверів, які підтримуються спільнотою.

Графічний інтерфейс MongoDB Compass поставляється разом з системою, а також існують інші сторонні інструменти для адміністрування та перегляду даних.

MongoDB має багато варіантів використання у різних сферах:

- реєстрація та зберігання інформації про події;
- системи управління документами та контентом;
- електронна комерція;
- ігри;
- дані моніторингу, датчиків;
- мобільні додатки;
- сховище операційних даних веб-сторінок (наприклад, зберігання коментарів, рейтингів, профілів користувачів, сеанси користувачів).

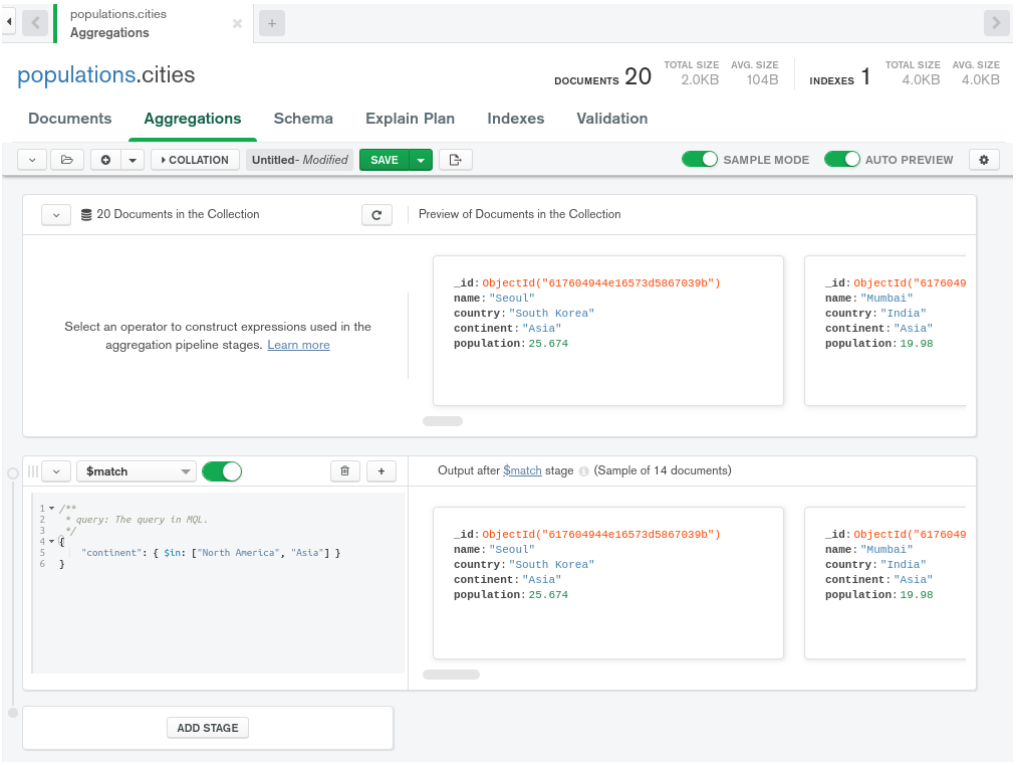


Рис. 1.15 – Інтерфейс користувача “MongoDB Compass”

Для керування та адміністрування системи бази даних у користувача є такі команди та інструменти:

– mongo - інтерактивна оболонка, яка дає змогу розробникам і адміністраторам переглядати, вставляти, видаляти й оновлювати дані у своїй базі даних, також дає змогу налаштувати реплікацію, сегментування, відключити вузли, виконати JavaScript або будь-які інші запити до бази даних;

– mongostat - інструмент командного рядка, який підсумовує список статистичних даних для виконуваного екземпляра MongoDB, це дає змогу візуалізувати кількість вставок, оновлень, видалень, запитів і команд, а також ресурсоспоживання екземпляра;

– mongotop - інструмент, який надає метод для відстеження часу, який зчитує або записує дані в екземплярі. Він також забезпечує статистику на рівні кожної колекції;

– mongosniff - інструмент, що забезпечує перехоплення, збереження і подальше відтворення команд, які надсилаються в MongoDB;

– mongoimport і mongoexport - засоби імпорту та експорту з JSON, CSV або TSV, підтримується низка інших форматів;

– mongodump і mongorestore - інструменти створення резервної копії та відновлення бази даних з неї.

Python - це загальнопризначена високорівнева мова програмування з динамічною суворою типізацією та автоматичним керуванням пам'яттю. Вона орієнтована на покращення продуктивності розробника, читабельності коду та якості програм, а також на забезпечення переносимості написаного коду. Python повністю підтримує об'єктно-орієнтоване програмування, де всі є об'єктами. Одним з унікальних рис синтаксису мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови є мінімалістичним, що дозволяє розробникам швидко вчитися та працювати з мовою без необхідності частого звертання до документації. Python використовується як інтерпретована мова і часто використовується для написання скриптів.

Python є мультипарадигмальною мовою програмування, яка підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване, метапрограмування та функціональне програмування.

Офіційна реалізація мови Python – CPython – є еталонною інтерпретацією мови. CPython підтримує більшість популярних платформ і є стандартом де-факто для мови Python. Він розповсюджується під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень у будь-яких додатках, включаючи пропрієтарні.

Стандартна бібліотека Python містить велику кількість корисних переносних функцій, які допомагають у різних сферах розробки, включаючи роботу з текстом, мережами та математичне моделювання. Для розширення можливостей мови та реалізації специфічних завдань доступні багато сторонніх бібліотек та інтеграція з кодом, написаним на мовах C або C++.

Python став одним з найпопулярніших мов програмування, широко використовуваним у галузі аналізу даних, машинного навчання, DevOps, веб-

розробки та інших сферах, включаючи розробку ігор. Читабельність, простий синтаксис та можливість інтерпретації без компіляції роблять Python популярним вибором для навчання програмуванню, дозволяючи зосередитися на освоенні алгоритмів, концепцій та парадигм програмування.

HTML (від англ. HyperText Markup Language - "мова гіпертекстової розмітки") - стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML-документ від сервера за протоколами HTTP/HTTPS або відкривають із локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою HTML різні конструкції, зображення та інші об'єкти, такі як інтерактивна веб-форма, можуть бути вбудовані в сторінку, що відображається. HTML пропонує засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML виділяються тегами, записаними з використанням кутових дужок. Такі теги, як *<img />* і *<input />*, безпосередньо вводять контент на сторінку. Інші теги, такі як *<p>*, оточують і оформляють текст у собі і можуть включати інші теги як поделементів. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

Мова XHTML є суворішим варіантом HTML, він слідує синтаксису XML і є додатком мови XML в області розмітки гіпертексту.

Мовою програмування JavaScript можна вбудовувати програмний код в HTML з метою керування поведінкою та вмістом веб-сторінок. Крім того, за допомогою CSS, що включається в HTML, можна визначати зовнішній вигляд та стиль сторінки. Текстові документи, що містять HTML-розмітку (зазвичай з розширенням .html або .htm), обробляються спеціальними програмами, які відображають документ у відформатованому вигляді. Ці програми, відомі як "браузери" або "інтернет-браузери", зазвичай надають користувачеві зручний інтерфейс для відкриття веб-сторінок, їх перегляду (включаючи виведення на зовнішні пристрої) та, при необхідності, надсилання даних, введених

користувачем, на сервер. На сьогоднішній день найпопулярнішими браузерами є Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Microsoft Edge і Safari.

HTML - тегова мова розмітки документів. Будь-який документ на мові HTML є набором елементів, причому початок і кінець кожного елемента позначається спеціальними позначками - тегами. Елементи можуть бути порожніми, тобто не містять жодного тексту та інших даних. У цьому випадку зазвичай не вказується тег, що закриває (наприклад, тег перенесення рядка <*br*> — одиночний і закривати його не потрібно) . Крім того, елементи можуть мати атрибути, що визначають будь-які їх властивості (наприклад, атрибут *href="* у посилання). Атрибути вказуються у відкритому тегу. Ось приклади фрагментів HTML-документа

<*strong*>Текст між двома тегами — відкриває та закриває.

<*a href="http://www.example.com"*>Тут елемент містить атрибут *href*, тобто гіперпосилання.</*a*>

А ось приклад порожнього елемента: <*br*>

Регістр, в якому набрано ім'я елемента та імена атрибутів, у HTML значення не має (на відміну від XHTML). Елементи можуть бути вкладені. Наприклад, наступний код:

```
<!DOCTYPE html>
<html lang="ua">
  <head>
    <meta charset="UTF-8">
    <title>HTML Document</title>
  </head>
  <body>
    <p>
      <b>
        Цей текст буде напівжирним, <i>а цей — ще й курсивним.</i>
      </b>
```

```
    </p>
  </body>
</html>
```

дасть такий результат:

Цей текст буде напівжирним, ***а цей - ще й курсивним.***

JavaScript - мова програмування, яка підтримує кілька парадигм. Вона підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування.

Зазвичай JavaScript використовується як вбудована мова для доступу до об'єктів програм та додатків. Вона широко використовується у браузерах для написання скриптів, які забезпечують інтерактивність веб-сторінок.

Основні архітектурні особливості JavaScript включають динамічну типізацію, слабку типізацію, автоматичне керування пам'яттю, прототипне програмування та функції як об'єкти першого класу.

Хоча JavaScript є об'єктно-орієнтованою мовою, прототипне спадкування, яке використовується в мові, відрізняється від традиційного класно-орієнтованого спадкування, що спостерігається в інших мовах. Крім того, JavaScript має деякі особливості, характерні для функціональних мов програмування, такі як функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання і т. д. Ці особливості надають JavaScript додаткової гнучкості.

Незважаючи на схожий із Сі синтаксис, JavaScript порівняно з мовою Сі має докорінні відмінності:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні функції.

У мові відсутні такі корисні речі, як:

- стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків щодо роботи з файловою системою, управління потоками введення-виведення, базових типів для бінарних даних;
- стандартні інтерфейси до веб-серверів і баз даних;
- система управління пакетами, яка б відстежувала залежності та автоматично встановлювала їх.

Синтаксис мови JavaScript багато в чому нагадує синтаксис Сі та Java, семантично ж мова набагато ближча до Self, Smalltalk або навіть Ліспу.

У JavaScript:

- усі ідентифікатори регістрозалежні,
- у назвах змінних можна використовувати букви, підкреслення, символ долара, арабські цифри,
- назви змінних не можуть починатися з цифри,
- для оформлення однорядкових коментарів використовуються //, багаторядкові та внутрішньорядкові коментарі починаються з /* і закінчуються */.

Мікросервісна архітектура (Microservices Architecture) - це підхід до розробки програмного забезпечення, в якому застосування розбивається на невеликі незалежні сервіси, які можуть взаємодіяти один з одним через API.

У порівнянні з монолітною архітектурою, де застосування будується як єдине ціле, мікросервіси являють собою окремі компоненти, що виконують конкретні функції. Монолітні додатки зазвичай складаються з користувацького інтерфейсу, бази даних та серверної частини. У монолітній архітектурі будь-яка зміна в системі вимагає перезбірки та розгортання всього застосування.

У мікросервісній архітектурі кожен сервіс виконує конкретну функцію і може бути розроблений, тестований та розгорнутий незалежно від інших сервісів. Це дозволяє розробникам швидко та гнучко реагувати на зміни у вимогах до застосування. Мікросервіси можуть бути написані на різних мовах

програмування, використовувати різні технології та працювати на різних серверах або контейнерах.

Мікросервісна архітектура має кілька переваг, таких як підвищена відмовостійкість, покращена масштабованість та ефективне використання ресурсів серверів. Вона також сприяє ускоренню процесу розробки та доставки нових функцій. Проте, вона може вимагати складнішого управління та моніторингу сервісів, а також збільшувати витрати на розробку та тестування.

Загалом, мікросервісна архітектура є ефективним підходом до розробки сучасних застосунків, що дозволяє підвищити гнучкість та реагування системи на зміни в вимогах та умовах експлуатації.

Bootstrap (також відомий як Twitter Bootstrap) - вільний набір інструментів для створення сайтів і веб-додатків. Містить HTML і CSS шаблони оформлення для типографіки, веб форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включно з JavaScript-розширеннями.

Bootstrap використовує сучасні напрацювання в галузі CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Основні інструменти Bootstrap:

- Сітки - заздалегідь задані розміри стовпчиків, які можна одразу використовувати, наприклад, ширина стовпчика 140 px відноситься до класу *.span2 (.col-md-2* у третій версії фреймворка), який можна використовувати в CSS-описі документа.
- Шаблони - фіксований або гумовий шаблон документа.
- Типографіка - описи шрифтів, визначення деяких класів для шрифтів, таких як код, цитати тощо.
- Медіа - надає деяке управління зображеннями та відео.
- Таблиці - засоби оформлення таблиць, аж до додавання функціональності сортування.
- Форми - класи для оформлення форм і деяких подій, що відбуваються з ними.

- Навігація - класи оформлення для панелей, вкладок, переходу сторінками, меню та панелі інструментів.
- Алерти - оформлення діалогових вікон, підказок і спливаючих вікон.

1.4 Розробка структури системи

Отже, нам відомо, що система складається с багатьох елементів, які показані у схемі. Усе, крім сайту налаштувань і бази даних, написано за допомогою мови Python. А в самій системі використовується підхід із використанням мікросервісів, що означає, що кожна невелика задача виконується окремо, своєю програмою. Усі взаємодії виконуються за допомгою HTTP протоколу. Винятком є лише зчитуючий пристрій (і модуль Raspberry Pie, які описані вище.)

Розглянемо структурну схему (рис. 1.16) пропонованої системи.

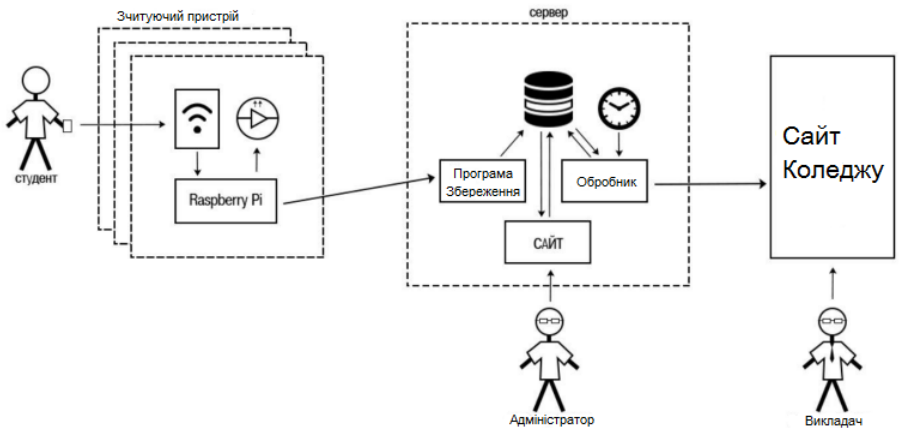


Рис. 1.16 – Схема роботи пристрою

1.4.1 Використання HTTP дієслів для зв'язку з інтерфейсами та програмами між собою

HTTP-дієслова є основною частиною "єдиного інтерфейсу", що дозволяють проводити обмежувальні та авторизаційні дії до джерела імені. Основними або найпоширенішими HTTP-дієсловами (або методами, як їх іноді називають) POST, GET, PUT і DELETE. Вони відповідають наступним функціям Створення, читання, оновлення та видалення (або CRUD в цілому). Є також інші дієслова, але

використовуються рідше. Серед менш вживаних OPTIONS і HEAD. Ми використовували лише два методи POST і GET, тому я поясню їх докладніше.

GET. Метод HTTP GET використовується для отримання (або читання) представлення ресурсу.. У разі "успішного" (або безпомилкового) виконання GET повертає XML або JSON-представлення ресурсу в поєднанні з кодом статусу HTTP 200 (OK). у поєднанні з кодом статусу HTTP 200 (OK). Якщо є помилки зазвичай повертається код 404 (NOT FOUND) або 400 (BAD REQUEST).

Відповідно до специфікації HTTP, GET-запити (так само як і HEAD-запити) використовуються тільки для читання даних, а не для їх зміни. Тому, якщо вони вони відповідають цій угоді, вони вважаються безпечними. Це означає, що можна використовувати без ризику модифікації даних, незалежно від того, чи дані були отримані один раз, 10 разів або жодного разу. GET (а також HEAD) є ідентичними (ідемпотентними), що означає

Це означає, що одні і ті ж дані отримуються за допомогою одних і тих же команд

GET (як і HEAD) запити є ідентичними (idempotent), що означає, що одні й ті ж дані отримуються за допомогою одних і тих же запитів (як одного, так і декількох).

POST. POST-запит найчастіше використовується для створення нових ресурсів. На практиці він використовується для створення вкладених ресурсів. Іншими словами, коли створюється новий ресурс, POST-запит надсилається сервісу до батьківського ресурсу, і таким чином сервіс бере на себе відповідальність за зв'язування створеного ресурсу з батьківським ресурсом, присвоєння ідентифікатора новому ресурсу і так далі.

Коли ресурс успішно створено, повертається HTTP код 201 і в заголовку "Location" передається адреса створеного ресурсу.

POST не є безпечним або іденпотентним запитом. Тому адресу рекомендується використовувати для не іденпотентних запитів. Виконання

ідентичних POST-запитів призведе до дуже схожих, але не ідентичних результатів.

1.4.2 Підключення Rasbperry Pi і зчитуючого пристрою

Модуль RFID-RC522 SainSmart працює з RFID-мітками Mifare і використовує чіп RC522, для Raspberry Pi важливо використовувати модуль 3.3V, оскільки він сумісний з входами напруги Raspberry Pi.

Послідовною шиною периферійного інтерфейсу (SPI) є шина синхронної послідовний канал передачі даних, який Raspberry Pi підтримує через GPIO, PI підтримує два вхідних пристроїв із використанням CE (Chip Enable) виходів.

1.4.3 Активація SPI на Raspberry Pi

Оскільки SPI не ввімкнено за замовчуванням, мені потрібно було відредагувати *raspi-blacklsit.conf* для того, щоб увімкнути інтерфейс SPI;

Необхідно було додати *'#'* на початку рядка *spi-bcm2708*, щоб прибрати його з чорного списку. Збережіть файл і перезавантажте Raspberry Pi, після чого команда *lsmod* повинна показати, що ІПБ пристрій *(spi_bcm2708)* увімкнено.

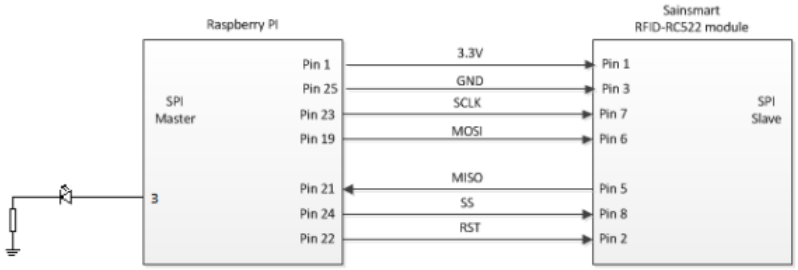


Рис. 1.17 – Схема підключення зчитуючого пристрою до Raspberry Pie

Щоб використовувати модуль з Python, необхідно завантажити бібліотеку SPI, однак нам потрібно встановити 'python-dev', щоб дозволити нам встановити оболонку SPI.

Щоб встановити *'python-dev'* необхідно ввести в консолі:

sudo apt-get install python-dev

Для зчитування даних з шини SPI в Python мені потрібен набір підпрограм, відповідний набір SPI-Py, я взяв з GitHub.

1.4.4 Сайт для внесення і редагування даних

Цей веб-сайт був створений з використанням веб-платформи Meteor, яка базується на Java Script. Архітектура Meteor ґрунтується на мікросервісах, і я покажу, як це застосовується в подальшому описі.

Сам сайт написаний з використанням HTML, CSS та Java Script, як це зазвичай буває у класичних веб-сайтах, з використанням технології Bootstrap. Однак, Java Script використовується як для розробки backend, так і для frontend. У подальшому описі буде роз'яснено код в цілому, а також будуть вказані місця, де використовується backend та де - frontend.

Основний акцент я хочу зробити на особливостях розробки веб-сайту з використанням Meteor, оскільки решта є досить типовою. Давайте розпочнемо з опису зовнішньої частини головної сторінки.

<template name="home">

У першому рядку ми використовуємо шаблон, що дозволяє нам визначити частину програмного коду та присвоїти йому ім'я, яке можна використовувати для звернення до нього в майбутньому. Коли нам знадобиться цей фрагмент коду в іншій частині програми, у деяких випадках ми можемо просто звернутися до нього за його ім'ям, як показано тут.

```
FlowRouter.route('/home', {
  name: 'home',
  action: function() {
    BlazeLayout.render("mainLayout", {
      content: "home"
    });
  }
});
```

Або написавши ось таку конструкцію.

{{>home}}

Ця частина коду написана мовою HTML, але в ній зустрічаються незвичайні конструкції *{{> studentList}}* і *{{> addStudentData}}* це означає, що замість цих конструкцій ми підставляємо сюди код позначений *<template name=" studentList">* і *<template name=" addStudentData ">*

```
<table class="table table-bordered table-condensed">
  <caption>студенти</caption>
  <thead>
   <tr>
     <th>ID</th>
     <th>MudleID</th>
     <th>ПІБ</th>
     <th>Група</th>
     <th class="vidstup">x</th>
   </tr>
  </thead>
 <tbody>
   {{> studentList}}
 </tbody>
</table>
{{> addStudentData}}
  </div>
</div>
  </div>
</div>
```

Цей фрагмент коду відображає дані користувача, який увійшов на веб-сайт. Це означає, що програма отримує дані про користувача з сервера, який успішно

авторизувався, і встановлює ці дані у відповідну форму, яка відображає ці дані на сторінці.

```
{{#with currentUser}}
    <div class="ui raised segment">
        <h1 class="ui header">Current User</h1>
        <p>First Name: {{profile.firstName}}</p>
        <p>Last Name: {{profile.lastName}}</p>
        <p>Profile Picture: <img class="ui small circular
image" src="{{profile.profPicture}}" /></p>
        <p>Organization: {{profile.organization}}</p>
 </div>
 {{/with}}
```

Давайте розглянемо бекендову частину додатка, яка відповідає за створення та безпечне зберігання особистих даних користувачів.

```
Accounts.onCreateUser(function(options, user) {
```

Можна скористатися наданим профілем в налаштуваннях або створити новий порожній об'єкт профілю.

```
user.profile = options.profile || {};
```

Призначає ім'я та прізвище новоствореному об'єкту користувача.

```
 user.profile.firstName = options.firstName;
 user.profile.lastName = options.lastName;
     user.profile.profPicture = Meteor.absoluteUrl() +
"img/default/user.jpg";
 user.profile.organization = ["Org"];
 user.roles = ["User"];
```

Повертає об'єкт користувача.

```
    return user;
});
```

Подивимося на вікно авторизації та реєстрації, оскільки вони мають однакову структуру, розглянемо одне з них. Це HTML, тому тут все просто і зрозуміло. В цих вікнах описані поля для введення даних, такі як поля для введення ім'я користувача та пароля, а також кнопка для підтвердження.

```
<template name="register">
<form id="register" class="login">
 <h1 class="ui header">Register</h1>
 <input class="email" type="email" placeholder="Email
Address" id="email" />
 <input class="text" type="text" placeholder="First Name"
id="first-name" />
 <input class="text" type="text" placeholder="Last Name"
id="last-name" />
 <input class="password" type="password"
placeholder="Password" id="password" />
 <button class="button" id="reg-button"
type="submit">Реєстрація</button>
 </form>
</template>
```

На рисунку 1.18 представлено вид цієї частини програми.

Рис. 1.18 – Вікно реєстрації

А тепер звернемося до backend.

Перші 3 рядки - обробник натискання кнопки.

```
Template.register.events({
 'click #register-button': function(e, t) {
 e.preventDefault();
```

Отримання значень полів введення

```
 var email = $('#email').val(),
 firstName = $('#first-name').val(),
 lastName = $('#last-name').val(),
 password = $('#password').val(),
 passwordAgain = $('#password-again').val();
```

Перевірка пароля на довжину не менше 6 символів

```
var isValidPassword = function(pwd, pwd2) {
 if (pwd === pwd2) {
 return pwd.length >= 6 ? true : false;
 } else {
 return swal({
 title: "Passwords don't match",
 text: "Please try again",
 showConfirmButton: true,
 type: "error"
 });
 }
 }
```

Если проверка пройдена ставим отметку, иначе нужно ввести другой пароль.

Далее идет функция Meteor.*loginWithPassword().*

```
if (isValidPassword(password, passwordAgain)) {
 Accounts.createUser({
```

```
email: email,
firstName: firstName,
lastName: lastName,
password: password
}, function(error) {
if (error) {
return swal({
title: error.reason,
text: "Please try again",
showConfirmButton: true,
type: "error"
});
} else {
FlowRouter.go('/');
}
});
}
return false;
}
});
```

Маршрутизація на сайті виконується за допомогою пакета Iron Router.

Iron Router - це пакет для маршрутизації, спеціально розроблений для Meteor додатків.

Він не лише допомагає налаштовувати шляхи маршрутизації, але також надає можливість фільтрувати дії на певних шляхах та керувати підписками на дані.

Щоб використати його, потрібно просто вказати шлях до сторінки, а саме назву шаблону на другому і четвертому рядках.

```
FlowRouter.route('/', {
```

```
 name: 'loginpromt',
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "login"
 });
 }
});
FlowRouter.route('/home', {
 name: 'home',
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "home"
 });
 }
});
FlowRouter.route('/register', {
 name: 'register',
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "register"
 });
 }
});
```

У нашій програмі також присутня компонента, відповідальна за збереження наших даних. Ця частина програми не дозволяє некористувачам отримувати або редагувати дані, для яких вони не мають прав доступу.

```
import { Meteor } from 'meteor/meteor';
Meteor.startup(() => {
 // code to run on server at startup
```

```
});
Meteor.methods({
 'addStudentData'(studentData) {
if (! Meteor.userId()) {
 throw new Meteor.Error('not-authorized');
 }
 Students.insert(studentData);
 },
 'addRozkladData'(rozkladData) {
 if (! Meteor.userId()) {
 throw new Meteor.Error('not-authorized');
 }
 Rozklad.insert(rozkladData);
 },
 'updateStudentData'(id, studentData) {
 if (! Meteor.userId()) {
 throw new Meteor.Error('not-authorized');
 }
 Students.update(id, studentData);
 },
 'updateRozkladData'(id, rozkladData) {
 if (! Meteor.userId()) {
 throw new Meteor.Error('not-authorized');
 }
 Rozklad.update(id, rozkladData);
 },

 'deleteStudent'(id){
 if (! Meteor.userId()) {
```

```
throw new Meteor.Error('not-authorized');
}
Students.remove(id);
},
'deleteRozklad'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Rozklad.remove(id);
},
'getStudent'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
return Students.findOne(id);
},
'getRozklad'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
return Rozklad.findOne(id);
},
});
```

Таким чином із бази даних дані потрапляють на сторінку.

{{#each students}} це цикл.

```
<template name="studentList">
{{#each students}}
<tr>{{> studentData}}</tr>
{{/each}}
```

</template>

А це backend цієї ж дії. *return Students.find();* це звернення до бази студентів.

```
Template.studentList.helpers({
 students: function() {
 return Students.find();
 }
});
```

Таким чином відбувається заповнення таблиці.

```
<template name="studentData">
<td>{{Id}}</td>
<td>{{MoodleId}}</td>
<td>{{PIB}}</td>
<td>{{Group}}</td>
 <td>{{> task}} {{> Sedit}}</td>
</template>
```

Ось, як це виглядає в графічному інтерфейсі (Рис. 1.19)

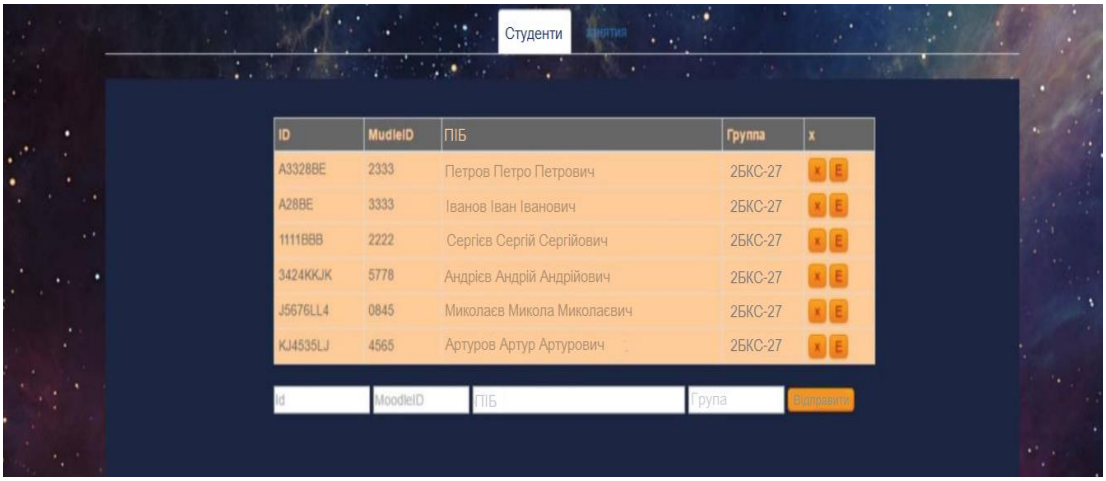


Рис. 1.19 – Таблиця студентів

Тепер розглянемо можливості видалення та редагування записів. Кожен запис має кнопку для видалення.

```
<template name="task">
 <button class="delete">&times;</button>
```

```
</template>
```

У Метеор є спеціальні функції для видалення ось так вона має має вигляд.

```
Template.task.events({
 'click .delete'() {
 Meteor.call("deleteStudent", this._id);
 Students.remove(this._id);
Meteor.call("deleteRozklad", this._id);
 Rozklad.remove(this._id);
 },
});
```

Після натискання кнопки "Редагувати", запис відображається у вікні, що з'являється внизу таблиці. Після внесення змін до запису, необхідно натиснути кнопку "Зберегти", щоб зберегти їх у новому вигляді.

```
Template.Redit.events({
 'click .Redit'() {
 console.log('Redit button click');
 id = this._id;
 rozkladData = Rozklad.findOne(id);
 $("#secretId1")[0].value = rozkladData._id;
 $(".str5")[0].value = rozkladData.cab;
 $(".str6")[0].value = rozkladData.time;
 $(".str7")[0].value =
rozkladData.weak;
 $(".str8")[0].value = rozkladData.day;
 $(".str9")[0].value = rozkladData.predmet;
 $(".str10")[0].value = rozkladData.IDpredmet;
 },
});
```

Одна з головних функцій додавання даних до бази даних

проводиться таким чином.

```
Template.addStudentData.events({
 'submit .add-student'(event)
```

Оголошення змінних.

```
event.preventDefault();
 const target = event.target;
 const secretId = target.secretId.value;
 const studentData = {
 Id : target.Id.value,
 MoodleId : target.MoodleId.value,
 PIB : target.PIB.value,
 Group : target.Group.value,
 }
```

Важливо зауважити, що якщо ми не маємо облікового запису в системі, то ми не маємо можливості додавати, видаляти або редагувати дані.

```
if(secretId !== '')
```

Додавання в базу даних.

```
Meteor.call("updateStudentData",
secretId,studentData);
 else
 Meteor.call("addStudentData",studentData);
```

Очищаємо форми.

```
target.Id.value = '';
 target.MoodleId.value = '';
 target.PIB.value = '';
 target.Group.value = '';
 target.secretId.value = '';
 },
```

1.4.5 Обробник даних

У системі для обробки даних використовуються три програми, кожна з яких виконує свої функції. Перша програма розташована на Raspberry Pi і відправляє дані на сервер.

Друга програма розташована на сервері і отримує дані від зчитувального пристрою, які потім надсилаються у базу даних.

Третя програма відповідає за обробку відвідувань. Вона порівнює існуючі дані про студентів зі списком тих, хто прийшов і позначився, та надсилає POST-запит на e.otfk.od.ua. Усі ці програми взаємодіють між собою за допомогою HTTP-запитів.

Наступним кроком я детальніше розкажу про внутрішній механізм кожної програми.

Перша програма reader.py

Підключаємо бібліотеку для роботи зі зчитувальним пристроєм і для підключення його до Raspberry Pi.

```
import MFRC522
import RPi.GPIO as GPIO
from time import sleep, time
from requests import post
```

Задаємо номер виходу для світлодіода, Задаємо адресу для надсилання даних, номер пристрою або кабінету, у якого встановлено пристрій.

```
PIN = 3
URL = 'http://10.3.73.105/id'
CAB = '301'
reader = MFRC522.MFRC522()
```

Функції для призначення входів і виходів на Raspberry Pi.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(pin, GPIO.OUT)
```

Функція для зчитування з ID-картки.

```
def read_uid():
 status,TagType=
reader.MFRC522_Request(reader.PICC_REQIDL)
 status, uid = reader.MFRC522_Anticoll()
 print(status, uid)
 return status, uid
```

Функція для свічення світлодіода.

```
def blink():
GPIO.output(pin, True)
sleep(.3)
GPIO.output(pin, False)
```

Основна ітерація включає такі кроки: зчитування даних, перевірка отриманих даних, відображення світлодіодного сигналу, конвертація ідентифікатора у потрібний формат, надсилання POST-запиту і очікування одну секунду.

```
while True:
 status,id = read_uid()
 if status == 0:
 blink()
 sid = "{0:X}{1:X}{2:X}{3:X}".format(*id)
 post(URL, {"cabinet": CAB, "id" : sid, "time" : time()})
 sleep(1)
```

Друга програма reciver.py

У даній програмі ми отримуємо інформацію, надіслану першою програмою, і зберігаємо її у нашій базі даних. Для цього ми підключаємо бібліотеку для взаємодії з MongoDB і використовуємо її методи для надсилання даних у MongoClient("localhost:3001").

```
from pymongo import MongoClient
client = MongoClient("localhost:3001")
```

```
db = client['meteor']
@app.route('/id', methods=['POST'])
def moodle_test():
print(request.form)
db.idtable.insert({"id": request.form["id"][:-2],
"cabinet": request.form["cabinet"], "time" :
request.form["time"]})
```

Третя програма process.py

Далі я роз'ясню і прокоментую деякі частини коду третьої частини частини.

Підключення бібліотеки для роботи з Mongo і для роботи з часом, відкриття сесії для обміну запитами.

```
from pymongo import MongoClient
from requests import session
import re
import time
```

Змінні для роботи з базою даних із різними таблицями.

```
dbs = MongoClient("localhost:3001")
meteor_db =dbs['meteor']
idtable = meteor_db['idtable']
students = meteor_db['students']
rozklad = meteor_db['rozklad']
```

Змінна для авторизації на e.otfk.od.ua.

```
payload = {
 'username': '******',
 'password': '******',
 'rememberusername': 1
}
```

Функція для передавання й оброблення даних за часом блок-схема роботи функції підставлена нижче (Рис. 1.20).

```
def process(para_time):
 with session() as c:
 r = c.post('https://e.oftk.od.ua/login/index.php',
data=payload)
 response = c.get('https://e.otfk.od.ua/grade/report/grader/index.php?id=1029')
 res = re.search('sesskey":"(\w*)"', response.text)
 sesskey = res.group(1)
 main(para_time, sesskey, c)
```

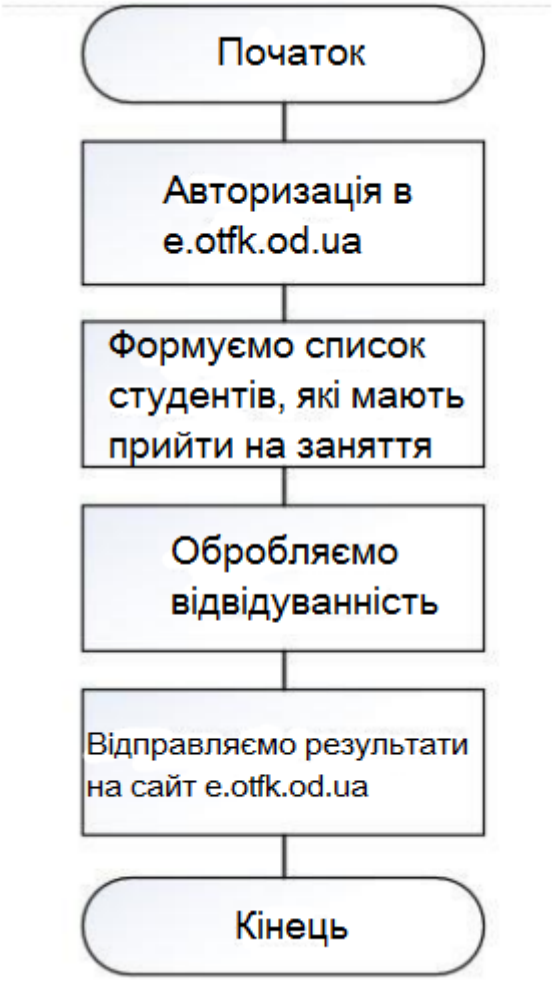


Рис. 1.20 – Блок-схема програми

Основна функція для формування списку студентів, які мають бути на занятті.

```
def main(para_time, sesskey, c):
 for predmet in get_predmet(para_time):
```

```
 print("== predmet: \n" + str(predmet))
 group_list = get_all_group(predmet)
 print("=== groups: \n" + str(group_list))
 students_list = list(get_all_student(group_list))
 print("=== students_list: \n" +
str(list(students_list)))
zanyattya = int(get_zanyattya(predmet))
 print("=== zanyattya: \n" + str(zanyattya) + " isTrue
" + str(bool(zanyattya)))
 if bool(zanyattya):
 print("= process Attandance")
 process_attandance(students_list, predmet,
para_time, zanyattya, sesskey, c)
 idtable.remove({})
```

Функція порівнює список студентів, які прийшли, і тих, хто має прийти, і за результатами порівнянь формує запит.

```
def process_attandance(students_list, predmet, para_time,
zanyattya, sesskey, c):
 print("== in process_attandance")
 print("== students_list: \n" + str(list(students_list)))
 for student in students_list:
 print("student " + str(student))
 cabinet = get_cabinet(predmet)
 if_attended = attend(student, para_time, cabinet)
 mark(if_attended, predmet["IDpredmet"], zanyattya,
student["MoodleId"], sesskey, c)
```

Функція для пошуку списку предметів у поточний час.

```
def get_predmet(para_time):
 print(list(rozklad.find()))
```

```
 return rozklad.find({"time": str(para_time)})
```

Функція для пошуку списку груп, які мають прийти на предмет.

```
def get_all_group(predmet):
predmet_str = predmet["group"]
 group_list = predmet_str.split(',')
 return [g.strip() for g in group_list]
```

Функція яка формує список з усіх студентів з цих груп.

```
def get_all_student(group_list):
 query = [ {"Group" : g } for g in group_list]
 return students.find({ '$or' : query})
```

Функція, яка повертає кабінет у корті, в якому проходить заняття.

```
def get_cabinet(predmet):
 return predmet["cab"]
```

Функція, яка формує список усіх занять предмета і потім видаляє заняття, яке пройшло.

```
def get_zanyattya(predmet):
 zanyattya_list = predmet["zanyattya"]
 if zanyattya_list:
 cur_zanyattya = zanyattya_list.pop()
 rozklad.update({"_id":predmet["_id"]}, {"$set":{
"zanyattya" : zanyattya_list}})
 return cur_zanyattya
 else:
 return 0
```

Функція перевіряє, чи прийшов студент на заняття в потрібний час.

```
def attend(student, para_time, cabinet):
 attend_data = idtable.find_one({"id": student["Id"],
"cabinet" :cabinet})
 if attend_data:
```

```
return is_correct_time(para_time,
attend_data["time"])
 else:
 return False
```

Формування запиту для сайту e.otfk.od.ua.

```
def mark(if_attended, predmetId, zanyattyaId, studentId,
sesskey, c):
 attend_data = 2 if if_attended else 1
 grade_data = {
 'id': predmetId, # // id курса
 'userid': studentId, # // id студента
 'itemid': zanyattyaId, # // id заняття
 'action':'update',
 'newvalue': attend_data , # // 1 відвідав, 2 не_відвідав
 'type':'scale',
 'sesskey': sesskey # // M.cfg.sesskey
 }
 print("mark data: " + str(grade_data))
```

Відправка запросу.

```
response=c.post('https://e.otfk.od.ua/grade/report/grader/ajax_callbacks.php',
data=grade_data)
```

Функція для переведення часу в потрібний формат.

```
def get_para_time(cur_time):
 hour = time.localtime(cur_time).tm_hour
 if hour == 8:
 return 1
 elif hour == 10:
 return 2
 elif hour == 12:
```

return 3

elif hour == 14:

return 4

elif hour == 15:

return 5

elif hour == 17:

return 6

elif hour == 19:

return

Отже у розділі було проаналізовано види існуючих систем контролю доступу, спроектовано зчитувальний пристрій та представлено код програми для обробки інформації про відвідування і надсилання даних на сайт, який стежитиме за графіком відвідування й автоматично вноситиме в нього дані.

2 ОХОРОНА ПРАЦІ

Спільні дії роботодавця та підлеглих йому служб повинні бути направлені на виконання вимог законодавства України в області охорони праці, для створення безпечних і здорових умов праці.

В даному розділі дипломного проекту розглядається питання проєктування системи обліку відвідування занять здобувачами освіти. Створення відповідної програми – це робота програміста. Тому до розгляду візьмемо його робоче місце.

Оператори і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика тощо.

На робочому місці програміста повинні бути створені умови для безпечної та високопродуктивної праці.

2.1 Розробка заходів з охорони праці

2.1.1 Виробниче середовище

Об’ємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПІН 3.3.2.007-98. Розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Площа на одне робоче місце становить не менше 6,0 м^2, а об’єм – не менше ніж 20,0 м^3. У приміщеннях слід щоденно робити вологе прибирання. Вони повинні бути оснащені аптечками першої медичної допомоги. При приміщеннях мають бути обладнанні побутові приміщення для відпочинку.

2.1.2 Освітлення

У приміщеннях, призначених для роботи з ПК, доцільно, щоб вікна були орієнтовані на північ або північний захід. На вікнах повинні бути штора або жалюзі, що регулюють рівень освітленості і захищають від прямого влучення

сонячних променів на робоче місце. При кольоровому оформленні виробничих і допоміжних приміщень необхідно враховувати орієнтацію їхніх вікон стосовно частин світу і використовувати гармонійне сполучення кольорів. Для стін і робочих поверхонь використовують мало насичені (основні) кольори, для невеликих помешкань або ділянок, що рідко потрапляють у поле зору працюючих, а також для створення контрастності – кольори середньої насиченості (допоміжні), для маленьких по площі поверхонь – насичені (акценти) – як функціональне фарбування. Стелі у всіх приміщеннях повинні бути білими. Поверхні устаткування в приміщеннях повинні бути матовими або напівматовими, для виключення випадку відблисків світла в очі працюючого, а стіни бути пофарбованими фарбами пастельних тонів.

Для штучного освітлення у приміщенні використовуються люмінесцентні лампи типу ЛБ, які в порівнянні з лампами розжарювання мають ряд істотних переваг: за спектральним складом світла вони близькі до природного світла, мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання); мають триваліший термін служби – до 10 тис годин.. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

Освітлення приміщення для роботи з ПК повинне відповідати ДБН В.2.5-28:2018 « Природне і штучне освітлення». Всі вимоги до виробничого освітлення в дипломній роботі виконані.

2.1.3 Гігієнічні нормування параметрів повітря робочої зони.

У виробничих приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря – ГОСТ 12.1.005-88, СН 4088-86.

Для підтримки в приміщеннях нормального, що відповідає гігієнічним вимогам складу повітря, видалення з нього шкідливих газів, пилу використовують вентиляцію. Механічна вентиляція (кондиціонери, вентилятори і т.д.) залежно від

напрямку руху повітряних потоків, може бути витяжною, припливною і припливно-витяжною.

Таблиця 2.1. - Значення їх відрізняються в залежності від пори року

Параметри мікроклімату	Значення параметрів	значення параметрів
	Взимку	влітку
Температура, C^0	22-24	23-25
Відносна вологість, %	40-60	40-60
Швидкість руху повітря, м/с	0,1	0,1-0,2

При природній вентиляції (за допомогою вікон) повітря надходить у приміщення і видаляється з нього внаслідок різниці температур і тиску.. Механічна вентиляція забезпечується вентиляторами, що забирають повітря зовні і направляє його до будь-якого робочого місця. або устаткування, а також видаляють забруднене повітря.

2.1.4 Розміщення робочих місць у приміщенні.

Розташування ПК, при якому працюючий звернений обличчям або спиною до вікон, неприпустимо при будь-якому способі реалізації загального висвітлення, як прямим, так і відбитим світлом. Робочі місця повинні бути розташовані так, щоб у поле зору працюючого не попадали поверхні, що мають властивість віддзеркалювання, вікна освітлювальні прилади. ПК повинні встановлюватися під кутом 90-100 градусів від вікон, так, щоб світло падало з боку. Робочі місця з ВПК доцільно розміщати в глибині приміщення.

Рис. 2.1 – Вимоги до розміщення ПК в приміщенні

Робочий стіл повинен регулюватися по висоті в границях 680-800 мм, а ширина – забезпечувати можливість виконання операцій в зоні досяжності моторного поля Рекомендовані розміри столу: висота 725 мм, ширина 600-1400 мм, глибина 800-1000 мм. Робочий стілець повинен бути оснащений підйомно-поворотним пристроєм для регулювання висоти сидіння і спинки, а також кута її нахилу. Регулювання кожного параметра повинне вироблятися легко, бути незалежним і надійно фіксуватися.

Розташування екрана ВДТ має забезпечувати зручність зорового спостереження у вертикальній площині під кутом $+30^0$ до нормальної лінії погляду працюючого.

Клавіатуру слід розташовувати на поверхні столу на відстані 100…300 мм від краю, звернутого до працюючого.

Рис. 2.2 – Правильне положення при сидінні за комп’ютером

2.1.5 Електробезпека

Використання електричної енергії на виробництві пов’язене з небезпекою дії електроструму на організм людини. Для захисту працюючих від ураження електричним струмом передбачені наступні заходи:

- недоступність струмоведучих частин;
- захисне заземлення (занулення) корпусів електрообладнання;
- передбачені рубильники закритого типу;

- блокіровка, надписи, плакати, засоби індивідуального захисту (калоші і боти діелектричні (ГОСТ 13385-78), рукавиці резинові діелектричні, коврики резинові діелектричні (ГОСТ 4997-75).

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном) мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

У приміщенні, де одночасно експлуатуються понад п’ять ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Не допускається підключати ЕОМ з ВДТ і ПК до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Штепсельні з’єднання та електророзетки для напруги 12В та 42В за своєю конструкцією та візуально (за кольором) мають відрізнятися від штепсельних з’єднань для напруги 127В та 220В

2.2 Пожежна безпека

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежогасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організацією своєчасної евакуації людей.

Для ліквідації пожеж використовують первинні засоби пожежогасіння, які призначенні для гасіння пожеж у початковій стадії їх розвитку. Вони є у всіх виробничих приміщеннях, цехах.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто питання покращення умов і полегшення частии рутини як для студентів так і для викладачів. На сьогоднішний день найефективнішим та найбільш надійним способом вирішення цього питання я вважаю встановлення системи контролю візитів занять. Монтаж такої системи забезпечує навчальний заклад від несанкціонованого проникнення сторонніх людей на територію якщо встановити головний пропускний пункт на вході в заклад, і дозволить відстежити, де знаходились здобувачі знань під час учбового дня.

На підставі проведеного дослідження ми з'ясували, що сам по собі механізм роботи дуже простий: система в індивідуальному порядку фіксує факт прибуття на заняття ПІБ і группу студента.

Замість того, щоб надавати починати кожну пару з переклички студентів, ви можете приступати до процессу навчання одразу після дзвоника, дозволяючи своїм викладачам не марнувати свій час на такі дрібниці.

Практична значимість дослідження полягала в тому, щоб дослідити принцип роботи пристрою, які нам продають спеціалізовані компанії і на основі схожих принципів роботи, сконструювати прототип, здатний на оптимізацію робочого часу. На основі цього дослідження була створенна система контролю візиту занять, яка не обмежена зарядом елементів живлення, має власне програмне забезпечення, і є ремонтопридатною, адже має дуже малу кількість апаратних компонентів.

ПЕРЕЛІК ВИКОРСИТАНИХ ДЖЕРЕЛ

1. Голубєв Л. П., Литвинов В. А. Автоматизована система авторизації доступу до ресурсів Arduino : thesis. 2017. URL: https://er.knutd.edu.ua/handle/123456789/6705
2. Стьопін В. І. Автоматизована система обліку відвідувань занять і генерації звітів : thesis. 2018. URL: https://openarchive.nure.ua/handle/document/20021
3. Goodwin S. Raspberry Pi. Smart Home Automation with Linux and Raspberry Pi. Berkeley, CA, 2013. P. 275–296. URL: https://doi.org/10.1007/978-1-4302-5888-9_8
4. Grebenyuk A. Використання системи контролю доступу для захисту інформації в офісних приміщеннях. International electronic scientific journal "Science Online". 2019. № 8. URL: https://doi.org/10.25313/2524-2695-2019-8-84
5. Permala A., Rantasila K., Pilli-Sihvola E. RFID. International Journal of Applied Logistics. 2012. Vol. 3, no. 2. P. 14–24. URL: https://doi.org/10.4018/jal.2012040102
6. Tamm G., Tribowski C. RFID. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. URL: https://doi.org/10.1007/978-3-642-11460-1
7. Что такое СКУД? URL: https://bcinform.ru/news/chto-takoe-skud-rasskazyivaem,-chto-eto-i-kakie-vidyi-byivayut.html
8. СКУД. Что это такое и где испозльуется. URL: https://ohrana.ua/stati-i-obzory/chto-takoe-skud.html
9. Raspberry Pi. URL: https://www.raspberrypi.org/
10. Вивчаємо Raspberry Pi. #1. Знайомство. URL: https://evo.net.ua/izuchaem-raspberry-pi.-chast-1.-znakomstvo/
11. Унікальні можливості Raspberry Pi: найцікавіше про одноплатні комп'ютери. URL: https://diylab.ub.ua/analitic/26599-unikalni-mojlivosti-raspberry-pi-naycikavishe-pro-odnoplatni-kompyuteri.html

12. Что такое python: чем он хорош, где пригодится, как его выучить? URL: https://netology.ru/blog/python

13. Большой туториал по MongoDB. URL: https://merrick-krg.medium.com/

14. Bootstrap: що це, з чого почати вивчення і як використовувати. URL:https://druzy.com.ua/bootstrap-sho-ce-z-chogo-pochati-vivchennia-i-iak-vikoristovyvati/

15. Розробка заходів з охорони праці та навколишнього середовища. URL: https://vuzlit.com/705752/rozrobka_zahodiv_ohoroni_pratsi_navkolishnogo_seredovischa

16. Основи охорони праці. URL: http://dspace.wunu.edu.ua/jspui/bitstream/316497/9184/1/%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D0%B8%D0%B9%20%D0%BA%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82%20%D0%BB%D0%B5%D0%BA%D1%86%D1%96%D0%B9.pdf

17. Гігієнічне нормування параметрів повітря робочої зони. URL: https://studies.in.ua/bjd-gandzyuk/930-102-ggyenchne-normuvannya-parametrv-povtrya-robochoyi-zoni.html

18. Електробезпека в виробничих приміщеннях. URL: https://allref.com.ua/uk/skachaty/Elektrobezpeka_v_virobnichih_primishennyah

19. Основи пожежної безпеки. URL: https://allref.com.ua/uk/skachaty/Elektrobezpeka_v_virobnichih_primishennyah

Сайт.

```
<template name="home">
FlowRouter.route('/home', {
 name: 'home',
 action: function() {
   BlazeLayout.render("mainLayout", {
     content: "home"
   });
 }
});
<template name=" studentList"> і <template name=" addStudentData
">
  <table class="table table-bordered table-condensed">
    <caption>студенти</caption>
    <thead>
      <tr>
        <th>ID</th>
        <th>MudleID</th>
        <th>ПІБ</th>
        <th>Група</th>
        <th class="vidstup">x</th>
      </tr>
    </thead>
    <tbody>
      {{> studentList}}
    </tbody>
  </table>
```

{{> addStudentData}}

* *

* *

* *

* *

{{#with currentUser}}

<h1 class="ui header">Current User</h1>

First Name: {{profile.firstName}}

Last Name: {{profile.lastName}}

Profile Picture:

Organization: {{profile.organization}}

{{/with}}

Accounts.onCreateUser(function(options, user) {

user.profile = options.profile || {};

user.profile.firstName = options.firstName;

user.profile.lastName = options.lastName;

user.profile.profPicture = Meteor.absoluteUrl() + "img/default/user.jpg";

user.profile.organization = ["Org"];

user.roles = ["User"];

Повертає об'єкт користувача.

return user;

});

<template name="register">

<form id="register" class="login">

<h1 class="ui header">Register</h1>

```
 <input class="email" type="email" placeholder="Email
Address" id="email" />
 <input class="text" type="text" placeholder="First Name"
id="first-name" />
 <input class="text" type="text" placeholder="Last Name"
id="last-name" />
 <input class="password" type="password"
placeholder="Password" id="password" />
 <button class="button" id="reg-button"
type="submit">Реєстрація</button>
 </form>
</template>
Template.register.events({
 'click #register-button': function(e, t) {
 e.preventDefault();
var email = $('#email').val(),
 firstName = $('#first-name').val(),
 lastName = $('#last-name').val(),
 password = $('#password').val(),
 passwordAgain = $('#password-again').val();
```

Перевірка пароля на довжину не менше 6 символів

```
var isValidPassword = function(pwd, pwd2) {
 if (pwd === pwd2) {
 return pwd.length >= 6 ? true : false;
 } else {
 return swal({
 title: "Passwords don't match",
 text: "Please try again",
 showConfirmButton: true,
```

```
type: "error"
});
}
}
Meteor.loginWithPassword().
if (isValidPassword(password, passwordAgain)) {
Accounts.createUser({
email: email,
firstName: firstName,
lastName: lastName,
password: password
}, function(error) {
if (error) {
return swal({
title: error.reason,
text: "Please try again",
showConfirmButton: true,
type: "error"
});
} else {
FlowRouter.go('/');
}
});
}
return false;
}
});
FlowRouter.route('/', {
name: 'loginpromt',
```

```
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "login"
 });
 }
});
FlowRouter.route('/home', {
 name: 'home',
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "home"
 });
 }
});
FlowRouter.route('/register', {
 name: 'register',
 action: function() {
 BlazeLayout.render("mainLayout", {
 content: "register"
 });
 }
});
import { Meteor } from 'meteor/meteor';
Meteor.startup(() => {
 // code to run on server at startup
});
Meteor.methods({
 'addStudentData'(studentData) {
if (! Meteor.userId()) {
```

```
throw new Meteor.Error('not-authorized');
}
Students.insert(studentData);
},
'addRozkladData'(rozkladData) {
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Rozklad.insert(rozkladData);
},
'updateStudentData'(id, studentData) {
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Students.update(id, studentData);
},
'updateRozkladData'(id, rozkladData) {
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Rozklad.update(id, rozkladData);
},

'deleteStudent'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Students.remove(id);
},
```

```
'deleteRozklad'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
Rozklad.remove(id);
},
'getStudent'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
return Students.findOne(id);
},
'getRozklad'(id){
if (! Meteor.userId()) {
throw new Meteor.Error('not-authorized');
}
return Rozklad.findOne(id);
},
});
```

{{#each students}} це цикл.

```
<template name="studentList">
{{#each students}}
<tr>{{> studentData}}</tr>
{{/each}}
</template>
Template.studentList.helpers({
students: function() {
return Students.find();
}
```

```
});
<template name="studentData">
<td>{{Id}}</td>
<td>{{MoodleId}}</td>
<td>{{PIB}}</td>
<td>{{Group}}</td>
 <td>{{> task}} {{> Sedit}}</td>
</template>
<template name="task">
 <button class="delete">&times;</button>
</template>
Template.task.events({
 'click .delete'() {
 Meteor.call("deleteStudent", this._id);
 Students.remove(this._id);
Meteor.call("deleteRozklad", this._id);
 Rozklad.remove(this._id);
 },
});
Template.Redit.events({
 'click .Redit'() {
 console.log('Redit button click');
 id = this._id;
 rozkladData = Rozklad.findOne(id);
 $("#secretId1")[0].value = rozkladData._id;
 $(".str5")[0].value = rozkladData.cab;
 $(".str6")[0].value = rozkladData.time;
 $(".str7")[0].value =
rozkladData.weak;
```

```
 $(".str8")[0].value = rozkladData.day;
 $(".str9")[0].value = rozkladData.predmet;
 $(".str10")[0].value = rozkladData.IDpredmet;
 },
});
Template.addStudentData.events({
 'submit .add-student'(event)
event.preventDefault();
 const target = event.target;
 const secretId = target.secretId.value;
 const studentData = {
 Id : target.Id.value,
 MoodleId : target.MoodleId.value,
 PIB : target.PIB.value,
 Group : target.Group.value,
 }
if(secretId !== '')
```

Додавання в базу даних.

```
Meteor.call("updateStudentData",
secretId,studentData);
 else
 Meteor.call("addStudentData",studentData);
target.Id.value = '';
 target.MoodleId.value = '';
 target.PIB.value = '';
 target.Group.value = '';
 target.secretId.value = '';
 },
```

Обробник даних.

Reader.py

```
import MFRC522
import RPi.GPIO as GPIO
from time import sleep, time
from requests import post
PIN = 3
URL = 'http://10.3.73.105/id'
CAB = '301'
reader = MFRC522.MFRC522()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(pin, GPIO.OUT)
def read_uid():
 status,TagType=
reader.MFRC522_Request(reader.PICC_REQIDL)
 status, uid = reader.MFRC522_Anticoll()
 print(status, uid)
 return status, uid
```

Функція для свічення світлодіода.

```
def blink():
GPIO.output(pin, True)
sleep(.3)
GPIO.output(pin, False)
while True:
 status,id = read_uid()
 if status == 0:
 blink()
 sid = "{0:X}{1:X}{2:X}{3:X}".format(*id)
 post(URL, {"cabinet": CAB, "id" : sid, "time" : time()})
```

```
 sleep(1)
```

reciver.py

```
from pymongo import MongoClient
client = MongoClient("localhost:3001")
db = client['meteor']
@app.route('/id', methods=['POST'])
def moodle_test():
print(request.form)
db.idtable.insert({"id": request.form["id"][:-2],
"cabinet": request.form["cabinet"], "time" :
request.form["time"]})
```

process.py

```
from pymongo import MongoClient
from requests import session
import re
import time
dbs = MongoClient("localhost:3001")
meteor_db =dbs['meteor']
idtable = meteor_db['idtable']
students = meteor_db['students']
rozklad = meteor_db['rozklad']
payload = {
 'username': '******',
 'password': '******',
 'rememberusername': 1
}
def process(para_time):
```

```
 with session() as c:
 r = c.post('https://e.oftk.od.ua/login/index.php',
data=payload)
 response = c.get('https://e.otfk.od.ua/grade/report/grader/index.php?id=1029')
 res = re.search('sesskey":"(\w*)"', response.text)
 sesskey = res.group(1)
 main(para_time, sesskey, c)
def main(para_time, sesskey, c):
 for predmet in get_predmet(para_time):
 print("== predmet: \n" + str(predmet))
 group_list = get_all_group(predmet)
 print("=== groups: \n" + str(group_list))
 students_list = list(get_all_student(group_list))
 print("=== students_list: \n" +
str(list(students_list)))
zanyattya = int(get_zanyattya(predmet))
 print("=== zanyattya: \n" + str(zanyattya) + " isTrue
" + str(bool(zanyattya)))
 if bool(zanyattya):
 print("= process Attandance")
 process_attandance(students_list, predmet,
para_time, zanyattya, sesskey, c)
 idtable.remove({})
def process_attandance(students_list, predmet, para_time,
zanyattya, sesskey, c):
 print("== in process_attandance")
 print("== students_list: \n" + str(list(students_list)))
 for student in students_list:
 print("student " + str(student))
```

```
 cabinet = get_cabinet(predmet)
 if_attended = attend(student, para_time, cabinet)
 mark(if_attended, predmet["IDpredmet"], zanyattya,
student["MoodleId"], sesskey, c)
def get_predmet(para_time):
 print(list(rozklad.find()))
 return rozklad.find({"time": str(para_time)})
def get_all_group(predmet):
predmet_str = predmet["group"]
 group_list = predmet_str.split(',')
 return [g.strip() for g in group_list]
def get_all_student(group_list):
 query = [ {"Group" : g } for g in group_list]
 return students.find({ '$or' : query})
def get_cabinet(predmet):
 return predmet["cab"]
def get_zanyattya(predmet):
 zanyattya_list = predmet["zanyattya"]
 if zanyattya_list:
 cur_zanyattya = zanyattya_list.pop()
 rozklad.update({"_id":predmet["_id"]}, {"$set":{
"zanyattya" : zanyattya_list}})
 return cur_zanyattya
 else:
 return 0
def attend(student, para_time, cabinet):
 attend_data = idtable.find_one({"id": student["Id"],
"cabinet" :cabinet})
 if attend_data:
```

```
        return is_correct_time(para_time,
        attend_data["time"])
         else:
         return False
        def mark(if_attended, predmetId, zanyattyaId, studentId,
        sesskey, c):
         attend_data = 2 if if_attended else 1
         grade_data = {
         'id': predmetId, # // id курса
         'userid': studentId, # // id студента
         'itemid': zanyattyaId, # // id заняття
         'action':'update',
         'newvalue': attend_data , # // 1 відвідав, 2 не_відвідав
         'type':'scale',
         'sesskey': sesskey # // M.cfg.sesskey
         }
         print("mark data: " + str(grade_data))
        response=c.post('https://e.otfk.od.ua/grade/report/grader/ajax_callbacks.php',
data=grade_data)
        def get_para_time(cur_time):
         hour = time.localtime(cur_time).tm_hour
         if hour == 8:
         return 1
         elif hour == 10:
         return 2
         elif hour == 12:
         return 3
         elif hour == 14:
         return 4
```

```
elif hour == 15:
return 5
elif hour == 17:
return 6
elif hour == 19:
return
```

ДОДАТОК Б.

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ ДО ДИПЛОМНОГО ПРОЕКТУ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ДИПЛОМНИЙ ПРОЕКТ НА ТЕМУ:
Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ

- Дипломник: Куру С.І.
- Керівник: Скорнякова О.В.

Одеса, 2023р.

Вступ

• В ВСП «Одеський технічний фаховий коледж ОНТУ», як і в усіх закладах освіти, відзначають відвідування студентами занять в закладах. Відвідування занять у вищих навчальних закладах є необхідною складовою ефективного навчання та досягнення академічних результатів. Присутність студентів на лекціях, семінарах та практичних заняттях має велике значення для їхнього особистого розвитку та успішного засвоєння навчального матеріалу. Присутність на заняттях також дозволяє студентам вчасно отримувати актуальну інформацію, оновлення в програмі навчання, а також важливі оголошення та зміни в графіку занять. Таким чином це сприяє активному долученню до навчального процессу. Але одна із таких, здавалося б, дрібниць, як відзнака відвідування занять - може сильно перешкоджати викладачам, відбираючи у них найцінніше. Час!

СКУД – як основа

•Система контролю та управління доступом, СКУД (англ. Physical Access Control System, PACS) - сукупність програмно-апаратних технічних засобів контролю та засобів управління, що мають на меті обмеження та реєстрацію входу-виходу об'єктів (людей, транспорту) на заданій території через точки проходу.»: двері, ворота, КПП.

RFID – Сучасна технологія ідентифікації

•RFID - це сучасна технологія ідентифікації, що надає істотно більше можливостей порівняно з традиційними системами маркування.
•Найзручніший спосіб автоматичної ідентифікації об'єктів, у якому за допомогою радіосигналів зчитуються або записуються дані, що зберігаються в так званих транспондерах, або RFID-мітках.

Головний "мозок"

• Raspberry Pi є комп'ютером, який має розмір, подібний до банківської картки. Спочатку він був розроблений як доступна система для навчання інформатиці, але з часом отримав широке застосування та став популярним.
• Переважно працює на операційних системах, які базуються на ядрі Linux. Оскільки ARM11 заснований на 6-ій версії ARM, не всі версії Linux підтримують його.

IoT – Інтернет речей

•Windows 10 IoT (Internet of Things) є операційною системою, розробленою Microsoft для вбудованих систем та пристроїв Інтернету речей.
• Дозволяє розробникам створювати програмне забезпечення для різних пристроїв, таких як смарт-термометри, системи контролю доступу, домашні автоматизації та інші.

Meteor (веб-фреймворк)

• Вільний та відкритий вебфреймворк, написаний на мові JavaScript, який використовує Node.js.
• Дозволяє швидко створювати крос-платформові застосунки.
•Інтегрується з MongoDB і використовує розподіли дата-протокол та шаблон проектування публікація-підписка.
•На клієнті Meteor працює з jQuery і може бути використаний з будь-якою бібліотекою віджетів JS.

СУБД MongoDB

• Документо-орієнтована система управління базами даних з відкритим вихідним кодом, яка не вимагає опису схеми таблиць.
• Код MongoDB написаний на мові C++ і розповсюджується в рамках ліцензії AGPLv3.

Мова програмуваннння

• Інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.
• Підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду.
•Інтерпретатор та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах.

Мова розмітки

• Стандартизована мову розмітки документів для перегляду веб-сторінок у браузері.
• Елементи HTML є будівельними блоками сторінок HTML.
• Можна вбудовувати програми, написані на скриптових мовах, наприклад JavaScript, які впливають на поведінку та вміст вебсторінок.
• Розмітка в HTML складається з чотирьох основних компонентів.

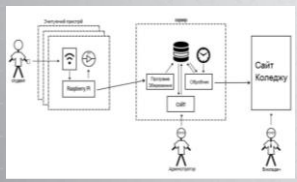
Скриптова мова програмування

• Динамічна, об'єктно-орієнтована прототипна мова програмування.
• Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.
•Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну)

Клієнтський фреймворк

• Безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебзастосунків
• Містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript.
• Спрощує розробку динамічних вебсайтів і вебзастосунків.

Схема роботи пристрою

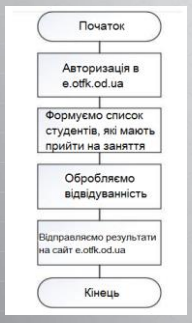


• Усе, крім сайту налаштувань і бази даних, написано за допомогою мови Python.
• В системі використовується підхід із використанням мікросервісів
• Усі взаємодії виконуються за допомгою HTTP протоколу. Винятком є лише зчитуючий пристрій

Вікно авторизації і таблиця студентів

• Веб-сайт був створений з використанням веб-платформи Meteor, яка базується на Java Script.
• Написаний з використанням HTML, CSS та Java Script, як це зазвичай буває у класичних веб-сайтах, з використанням технології Bootstrap. Однак, Java Script використовується як для розробки backend, так і для frontend.

Обробник даних



• Reader.py. Розташована в Raspberry Pi. Відправляє дані на сервер.
• Reciver.py. Розташована на сервері і отримує дані від зчитувального пристрою, які потім надсилаються у базу даних.
• Process.py. Третя програма відповідає за обробку відвідувань. Вона порівнює існуючі дані про студентів зі списком тих, хто прийшов і позначився, та надсилає POST-запит на e.otfk.od.ua. Усі ці програми взаємодіють між собою за допомогою HTTP-запитів.

Результат

• У кваліфікаційній роботі було розглянуто питання покращення умов і полегшення частини рутини як для студентів так і для викладачів.
•На підставі проведеного дослідження ми з'ясували, що сам по собі механізм роботи дуже простий: система в індивідуальному порядку фіксує факт прибуття на заняття ПІБ і групу студента.
•Практична значимість дослідження полягала в тому, щоб дослідити принцип роботи пристрою, які нам продають спеціалізовані компанії і на основі схожих принципів роботи, сконструювати прототип, здатний на оптимізацію робочого часу.

Дякую за увагу!

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015611675

Дата перевірки:
15.06.2023 12:13:10 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2023 12:16:57 EEST

ID користувача:
100011688

Назва документа: **2БКС-27 - Куру С.**

Кількість сторінок: **62** Кількість слів: **11465** Кількість символів: **84857** Розмір файлу: **1.40 MB** ID файлу: **1015259461**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

32.5%
Схожість

Найбільша схожість: **8.49%** з Інтернет-джерелом (http://elib.sfu-kras.ru/handle/2311/30954)

32.5% Джерела з Інтернету | **1000** ..... Сторінка 64

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | **12 сторінок**

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Куру Сергій Іванович,
здобувач освіти гр. 2БКС-27, та

Скорнякова Олена Володимирівна,
керівник випускної кваліфікаційної роботи,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ» (автор роботи – Куру С.І., керівник роботи – Скорнякова О.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Куру С.І. /

Керівник _____ / Скорнякова О.В./

«16» 06 2023 р.

ВСП «Одеський технічний фаховий коледж ОНАХТ»

ВІДГУК

Керівника про кваліфікаційну роботу бакалавра

Куру Сергія Івановича

(прізвище, ім'я та по батькові)

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

Тема кваліфікаційної роботи

Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

а) Обсяг і якість виконання роботи (розрахунково-пояснювальної записки)

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми кваліфікаційної роботи, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у роботі. Представлено програмний код. Є у наявності є презентація. Презентація виконана якісно, у достатньому обсязі. Презентація наочно демонструє результати роботи.

б) Самостійність роботи над кваліфікаційною роботою

Здобувач отримав завдання від керівника, провів аналіз існуючих рішень і зробив необхідні висновки для реалізації проекту, провів тестування та апробацію системи в реальних умовах. Продемонстрував уміння реагувати на вимоги та виправляти виявлені недоліки, відгукуватися на пропозиції та втілювати їх оперативно в проект. Виявив навички самостійно опрацьовувати новий матеріал та виконувати пошук необхідної літератури та інших джерел інформації.

в) Теоретична підготовка бакалавра

відповідає вимогам до бакалавра зі спеціальності

«Комп'ютерна інженерія»

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

В процесі роботи над кваліфікаційною роботою здобувач продемонстрував уміння використовувати останні досягнення науки та техніки в предметній галузі на підставі відповідної навчальної та науково-технічної літератури, достатньо впевнено користувався програмним забезпеченням при роботі над роботою та створенням презентації.

Загальна оцінка *добре*

Прізвище, ім`я, по батькові *к.п.н. Скорнякова Олена Володимирівна*

Місто роботи і посада керівника проекту: к.пед.н., викладач-методист комісії КТ та ПІ_ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету»

Підпис _____

«19» 06 2023 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ КОЛЕДЖ ОНАХТ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра
відділення комп'ютерних систем

Куру Сергію Івановичу

(прізвище, ім`я та по батькові)

Напрямку підготовки *123 «Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Скорнякова О.В.*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи **Проектування системи обліку відвідування занять здобувачами відділення комп'ютерних систем ВСП ОТФК ОНТУ**

Обсяг пояснювальної записки 90 сторінок

Обсяг графічної (презентаційної) частини проекту 17 аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаної роботи завданню

Робота відповідає технічному завданню до дипломного проекту. Виконана у відповідності з вимогами.

б) характеристика виконання кожного розділу роботи

Даний проект має на меті розробку системи обліку відвідування занять, яка допоможе установі або організації з ефективним веденням обліку присутності студентів, викладачів або співробітників на заняттях. Система буде забезпечувати автоматизовану фіксацію даних про відвідування, а адміністратор зможе аналізувати ці дані і надавати звіти та статистику для подальшого використання.

в) оцінка якості виконання графічної (презентаційної) частини роботи і пояснювальної записки

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей роботи

Практична значимість дослідження полягала в тому, щоб дослідити принцип роботи пристрою, які нам продають спеціалізовані компанії і на основі схожих принципів роботи, сконструювати прототип, здатний на оптимізацію робочого часу. На основі цього дослідження була створенна система контролю візиту занять, яка не обмежена зарядом елементів живлення, має власне програмне забезпечення, і є ремонтопридатною, адже має дуже малу кількість апаратних компонентів

д) основні недоліки роботи

У тексті пояснювальної записки відсутні посилання на використану літературу.

У розділі охорони праці наведені відомі нормативні вимоги загального плану замість конкретних розрахунків освітлення приміщення, вентиляції, рівня шуму.

Оцінка розрахункової частини 5(відмінно)

Оцінка графічної (презентаційної) частини 5 (відмінно)

Загальна оцінка 5(відмінно)

Прізвище, ім'я та по батькові рецензента Василіу Євген Вікторович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ, декан факультету інформаційних технологій та кібербезпеки

«16» 06 2023 р.

(підпис)

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ КАДРІВ ДУІТЗ

ВІДДІЛ КАДРІВ