Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій і робототехніки
(повне найменування інституту, назва факультету (відділення))
Кафедра автоматики, електроніки та телекомунікацій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи

бакалавр
(освітній рівень)

на тему **Розробка програмного комплексу візуалізації даних самохідної радіокерованої системи**

Виконав: студент 4 курсу, групи 401-тт спеціальності 172 «Телекомунікації та радіотехніка»
(шифр і назва напряму підготовки, спеціальності

Базарний А. В.
(прізвище та ініціали)
Керівник Обіход Я.Я.
(прізвище та ініціали)
Рецензент Шефер О.В.
(прізвище та ініціали)

Полтава - 2021 рік

РЕФЕРАТ

кваліфікаційної роботи «Розробка програмного комплексу візуалізації даних самохідної, радіокерованої системи»

Робота містить 56 сторінок, 19 ілюстрацій, 4 таблиці, 22 використаних джерела.

Ключові слова: веб-застосунок, HTML, PHP, CSS, візуалізація.

Об’єктом розроблення кваліфікаційної роботи є – програмний комплекс візуалізації даних самохідної радіокерованої телеметричної системи.

Метою кваліфікаційної роботи є:

- Аналіз принципу роботи сучасних веб-застосунків
- Аналіз програмних засобів, які необхідні для роботи веб-застосунку
- Аналіз мов програмування для розробки програмного комплексу
- Розробка програмного комплексу візуалізації даних самохідної радіокерованої системи

У процесі виконання даної КРБ буде проведено аналіз принципу роботи сучасних веб-застосунків та використовуючи отримані знання зможемо створити програмний комплекс візуалізації даних самохідної радіокерованої системи та подібних їй віддалено керованих систем або систем моніторингу.

ABSTRACT

Qualification work "Development of a software package for data visualization of self-propelled, radio-controlled systems"

The work contains 56 pages, 19 illustrations, 4 tables, 22 sources used.

Keywords: web application, HTML, PHP, CSS, visualization.

The object of development of qualification work is a software package for data visualization of self-propelled radio-controlled telemetry system.

The purpose of the qualification work is:

- Analysis of the principles of modern web applications
- Analysis of software required for web applications
- Analysis of programming languages for software development
- Development of a software package for data visualization of self-propelled radio-controlled system

In the process of implementing this CRB, the principle of modern web applications will be analyzed, and with the help of the acquired knowledge we will be able to create a software package for data visualization of self-propelled radio-controlled system and similar remote controlled system or system monitoring systems.

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Інститут Навчально-науковий інститут інформаційних технологій та робототехніки
Кафедра Автоматики, електроніки та телекомунікацій
Освітньо-кваліфікаційний рівень Бакалавр
Спеціальність 172 «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри автоматики, електроніки та телекомунікацій

_____О.В. Шефер
“ 10 ” травня 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРУ СТУДЕНТУ

Базарному Андрію Вікторовичу

1. Тема роботи «Розробка програмного комплексу візуалізації даних самохідної, радіокерованої системи»
керівник роботи Обіход Ярослав Якович, к.т.н., доцент
затверджена наказом вищого навчального закладу від 03.03.2021 року № 158 -фа

2. Строк подання студентом проекту (роботи) 15.06.2021 р.

3. Вихідні дані до проекту (роботи) мова програмування PHP, HTML, CSS, JavaScript.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити). Обґрунтування необхідності розробки програмного комплексу візуалізації даних самохідної радіокерованої системи. Реалізація програмного комплексу візуалізації даних самохідної радіокерованої системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових плакатів):
1) Схема роботи веб-застосунку;
2) Схема взаємодії веб-серверу з клієнтом;
3) Схематичне зображення функції отримання даних;
4) Схема взаємодії клієнту, серверу і модуля PHP.

6. Дата видачі завдання 10.05.2021 р.

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів дипломної роботи	Термін виконання етапів роботи			Примітка (плакати)
1	Аналіз розвитку та принципу роботи веб-застосунків.	18.05.21			
2	Огляд та визначення з програмними засобами та мовами програмування для реалізації проєкту.	02.06.21			
3	Реалізація програмного комплексу візуалізації даних самохідної радіокерованої системи.	09.06.21			
4	Оформлення кваліфікаційної роботи бакалавра.	15.06.21	II		

Студент _____ Базарний А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Обіход Я.Я.
(підпис) (прізвище та ініціали)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СТРП – самохідна телеметрична радіокерована платформа

ОС – операційна система

PHP – Hypertext Processor

ПЗ – програмне забезпечення

БД – база даних

СУБД – система управління базами даних

URL – Uniform Resource Locator – «єдиний вказівник на ресурс»

Зміст

ВСТУП

Системи, що здійснюють прийом і обробку телеметричної інформації та містять прийомопередаюче і допоміжне обладнання (імітатори сигналів, конвертори, нормалізатори сигналів, антенні пристрої) називаються інформаційно-телеметричними. Для деяких рухомих об’єктів інформаційно-телеметричні системи є невід’ємними компонентами. Наприклад в ракетах, космічних апаратах телеметричне обладнання застосовується для спостереження за процесом ракетного запуску, для отримання інформації про параметри зовнішнього середовища (температури, прискорень, вібрацій), про енергопостачанні і енерговитратах, про точне вирівнювання антени [1].

Проблемою візуалізації телеметричних даних є велике різноманіття операційних систем, під кожну із яких потрібно окремо писати або адаптувати додаток-клієнт. Вирішенням цієї проблеми є створення програмного комплексу на основі веб-технологій.

Веб-застосунок – це клієнт-серверний застосунок, у якому взаємодія клієнту з сервером виконується за допомогою браузеру та веб-серверу. На сьогоднішній день веб-застосунки користуються великим попитом. Вони стали невід’ємною частиною інтернет-простору завдяки тому, що будь-який користувач може скористатися функціями веб-додатку, маючи лише пристрій з доступом до мережі Internet. Великою перевагою веб-застосунків є відсутність встановлення додаткового програмного забезпечення. Для того, щоб повноцінно використовувати всі функції веб-застосунку необхідний лише зв’язок браузеру з веб-сервером [2].

Новизна даної роботи полягає у створенні та розробці програмного комплексу візуалізації даних самохідної радіокерованої системи із використанням веб-технологій.

Актуальність даної теми полягає в розробці програмного комплексу візуалізації даних самохідної, радіокерованої системи за допомогою веб-

технологій. Через значну поширеність робототехніки та дистанційно керованих систем в цілому, можна передбачити, що є потреба у зручному керуванні цих систем та обробці даних із них.

Основною перевагою використання веб-технологій є здатність програмного забезпечення працювати з декількома апаратними платформами або операційними системами без необхідності написання програмного забезпечення під кожну окрему платформу. На відміну від локальних застосунків, веб-застосунки є прості у розгортанні. Після завершення розробки програмного комплексу вони не потребують встановлення на пристрої користувачів, достатньо лише повідомити їм URL-адресу. Крім того, немає проблем з підтримкою старих версій операційних систем і апаратного забезпечення. У випадку з браузерними застосунками існує лише одна версія застосунку, в якій працюють всі користувачі і у випадку виходу нової версії всі автоматично переходять на неї, іноді навіть не помічаючи того.

1. АНАЛІЗ РОЗВИТКУ ТА ПРИНЦИПУ РОБОТИ ВЕБ-ЗАСТОСУНКІВ

1.1 Аналіз розвитку веб-технологій

Мало хто може уявити життя в сучасному світі без таких звичних для нас речей як комп'ютер та інтернет. Мала частина користувачів інтернету замислювалась коли-небудь як улаштовані та як створюються веб-сайти та інші інтернет ресурси.

Зрозуміти принципи створення веб-сайтів допоможуть веб-сторінки – це інформаційний ресурс, до якого можна отримати доступ із всесвітньої павутини. Найчастіше веб-сторінка написана у форматі файлу мови розмітки HTML. Веб-сторінки зазвичай об'єднуються в сайти за допомогою гіпертексту з використанням навігаційних гіперпосилань на інші сторінки. Їх можна отримати з віддаленого веб-серверу, або ж веб-сторінки можуть зберігатися на локальному комп'ютері [3].

Розглянемо етапи розвитку веб-технологій у світі.

Web 0.0 – в період створення були закладені та розроблені основні механізми, які забезпечили перехід до web 1.0. В ці часи виникли перші спроби в об'єднанні інформаційних мереж в одну глобальну. У цей період були розповсюдженні лінійні двоточкові системи комунікації між користувацькими терміналами та мейнфреймами. Також були розповсюджені квазіпоштові мережі по типу usenet, biznet, fidonet [4-5].

Web 1.0 – створення контенту на сайті було можливе лише його розробникам та адміністраторам. Це перша реалізація глобальної мережі інтернет, яку часто називали мережею тільки для читання. Способів взаємодії з контентом майже не було. Також в епоху web 1.0 почало з'являтись таке поняття як електронна комерція. З'явилась значна кількість сайтів, які пропонували користувачам платні послуги. Метою було представлення продуктів потенційним клієнтам у вигляді як це робить каталог, але перевагою веб-сайтів у даній сфері є можливість надати послугу чи товар для людини в будь-якому кутку світу [4].

Web 2.0 – у 2004 році видавництвом О'Рейлі було введено термін під назвою «Web 2.0» [6]. Друге покоління веб-сервісів це не є суттєвою еволюцією їх визначення, а скоріш за все це новий спосіб використовування ресурсів в мережі інтернет. Основна відмінність від попередніх поколінь є те, що користувачі веб-сайтів не є звичайними читачами, а також можуть стати учасниками у створенні контенту веб-сайту, тобто контент створюється користувачами сайту, але перевіряється адміністрацією. У цей час було створено найпопулярніші соціальні мережі такі як Facebook, ВКонтакті, Twitter[4].

Web 3.0 являє собою високоякісні сервіси та контент, який створюється на технологічній платформі web 2.0 професіоналами [6]. Через можливість у web 2.0 швидко та без значних фінансових затрат використовувати чималу кількість потужних інтернет сервісів, з'явилась велика кількість подібних між собою ресурсів і як наслідок це привело до девальвації більшості з них. Наочним прикладом переходу від web 2.0 до web 3.0 є німецький розділ Вікіпедії. У ньому по мірі наповнення контентом якісними статтями адміністратори вдаються до заборони редагування якісних статей недосвідченим учасникам [7].

1.2 Поняття та опис принципу роботи веб-застосунків

Більшості людей, які користуються персональним комп'ютером відомо, що являє собою застосунок для операційної системи Windows. Це комп'ютерна програма, яка дає змогу користувачеві вирішувати конкретні прикладні задачі [8]. Наприклад поштові клієнти, текстові редактори, медіаплеєри тощо. Приведемо кілька визначень, для розуміння, що являє собою веб-застосунок.

Веб-застосунок – розподілений застосунок у якому його логіка зосереджується на сервері, а функція зображення завантаженої із серверу інформації полягає на браузер [2]. Веб-застосунки – це програми, які виконуються на стороні веб-серверу та створюють інтерфейс між веб-сайтом та користувачем. Написані веб-застосунки найчастіше скриптовими мовами (PHP, Perl та ін.), рідше їх розробляють з використанням мов високого рівня (C, C++ та ін.), які потрібно компілювати під конкретну ОС.

Веб-сервер – сервер, який приймає запити по протоколу HTTP від клієнтів та віддає відповідь разом з HTML-сторінкою або якимись іншими даними. Доступ клієнтів до веб-серверу відбувається за допомогою URL-адреси сторінки, яка їм необхідна [9].

З розвитком всесвітньої мережі, веб-застосунки також досить стрімко розвиваються та вдосконалюються. Важливими характеристиками веб-застосунків являється доступність з практично будь-якого користувацького пристрою та з будь-якої точки світу, де наявний доступ до мережі. Також важливими характеристиками веб-застосунків є гнучкість, безпека та можливість масштабування. Окрім вище згаданих характеристик, веб-застосунок у наші часи повинен мати характеристику таку як мультиплатформність. Вона дає змогу веб-системі бути сумісною з будь-якою операційною системою та браузером, що вирішує проблему з сумісністю програмного забезпечення.

Оскільки веб-застосунок це програма, яка працює у браузері, то для його роботи потрібний доступ до мережі Інтернет та встановлений браузер на клієнтському пристрої. У такому випадку обчислення, обробка та зберігання інформації відбувається на віддаленому веб-сервері, а браузер на користувацькому пристрої являє собою програму-клієнт. На рисунку 1.1 приведено схематичне зображення принципу роботи веб-застосунку:

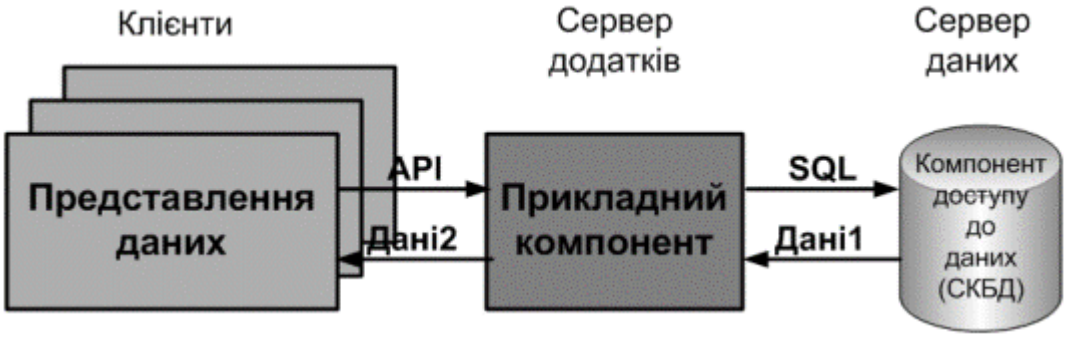

Рисунок 1.1 – Схема роботи веб-застосунку

Для прикладу візьмемо всесвітньо відомий поштовий клієнт Gmail. Даний поштовий клієнт не відрізняється від програмного забезпечення для роботи з

електронною поштою, яке потрібно встановлювати на свій пристрій. Веб-інтерфейс поштового клієнту Gmail в цілому написаний на браузерній скриптовій мові програмування JavaScript. Завдяки вибору даної мови програмування, інтерфейс клієнта отримав змогу приймати команди з клавіатури, оновлювати сторінки без перезавантаження за допомогою технології AJAX та багато інших можливостей [10].

Розглядаючи структуру веб-застосунку, можна побачити, що вся програмна логіка розташована на сервері, а до клієнтського інтерфейсу має доступ будь-який користувач за допомогою програми, яка існує з початку створення HTTP-мережі(веб-браузер). В часи зародження веб-технологій для появи повноцінних веб-застосунків не вистачало таких важливих технологій, як JavaScript та моделі DOM.

У наші часи більшість веб-ресурсів використовують технологію JavaScript для виконання деяких функцій на стороні клієнту без необхідності звертатися до віддаленого серверу. Але в період появи мови програмування JavaScript, використання даної технології значно відрізнялося від сьогодення. У ті часи технологію JavaScript використовували не через те, що вона необхідна, а тому, що вона просто існує. У якийсь момент використання JavaScript технологій того часу на веб-сторінках стало поганим тоном, адже в ті часи дана технологія використовувалась для того, щоб змусити елементи на сторінці реагувати на рух комп’ютерної миші, змінювати кольори, динамічне керування стилями елементів на сторінці.

Єдиною причиною, чому веб-застосунки почали стрімко ставати все більш популярними лише у теперішні часи це тому, що завжди слабкою стороною у створенні веб-застосунків була динамічність та інтерактивність у взаємодії з веб-сторінками у веб-застосунках. Повністю вирішити дану проблему допомагає JavaScript, який існує у сьогоденні.

Отже, ми з’ясували вище, що у звичайному програмному забезпеченні логіка програми розташовується у кожного користувача на комп’ютері, а у веб-застосунках вона розташовується на віддаленому сервері. У випадку розташування логіки застосунку на сервері є лише одна актуальна копія застосунку, у зв’язку з цією особливістю її можна легко поширювати серед усіх користувачів. По суті з використанням веб-технологій можна забути про звичний спосіб поширення застосунку, оскільки користувачі у даному випадку не отримують повну копію застосунку. Користувачі застосунку отримують на свої пристрої лише те, що їм необхідно для роботи, тобто інтерфейс програми. Із цього можемо стверджувати, що у поширенні веб-застосунків проблеми не існує, оскільки отримати доступ до нього ви можете з будь-якого пристрою та в будь-якому місці, де є підключення до мережі інтернет.

Для роботи з веб-застосунком користувачеві необхідна лише одна програма, яка називається веб-браузер. Щоб приступити до роботи з веб-застосунком, користувачеві потрібно лише набрати потрібний URL в адресному рядку браузеру.

На відмінно від випадку коли користувач встановлює на своєму пристрої застосунок, веб-застосунок розташовується на віддаленому сервері. Це звільняє користувача від ролі адміністратора. Йому не потрібно здійснювати встановлення програми, вирішувати проблеми з ними, тому, що у ролі адміністратора є людина чи група людей, які займаються розробкою веб-застосунку. Очевидно, що у такому випадку буде більше навантаження на розробника застосунку. Однак з економічної точки зору модель невеликої групи програмістів, які зосереджені у роботі над застосунком в одному місці набагато вигідніше. Адже така модель є більш ефективною та економічною тому, що не потрібно утримувати команду спеціалістів, які б займалися встановленням та налагодженням застосунків на користувацьких пристроях.

Веб-застосунок немає ніяких вимог до користувача. Заздалегідь передбачається, що користувач має на своєму пристрої браузер, адже це одна із основних програм на сучасних електронних пристроях. Основна ідея це те, що веб-

застосунок може працювати у будь-якій користувацькій операційній системі. Це обумовлено тим, що веб-застосунок не має ніяких особливих апаратних вимог. Разом із приходом веб-застосунків у минуле пішла проблема з підтримкою різноманіття версій. Як тільки розробник публікує нову версію застосунку, то усі без винятку клієнти одразу отримують останню версію, необхідно лише перезавантажити сторінку у веб-браузері. Після публікації розробником оновленої версії , усі застарілі версії веб-застосунку одразу ж зникають. Завдяки цьому клієнти можуть не помітити того, що вони вже використовують вже оновлену версію застосунку. Це також вирішує проблему зворотної сумісності програм, та підтримку старих та не актуальних версій застосунків.

Якщо у користувацьких пристроях сильно обмежений об’єм накопичувача інформації, то одним із варіантів є використання веб-застосунків. Адже відсутність потреби завантаження на свій пристрій повноцінного застосунку є значною перевагою у випадку, коли є обмеження у доступній пам’яті на пристрої. У деяких випадках можна навіть не завантажувати повністю інтерфейс, досить лише завантаження частини інтерфейсу, яка необхідна для виконання конкретної задачі. Тому завдяки цим особливостям веб-застосунки займають зовсім мало місця на накопичувачі користувацького пристрою і у зв’язку з цим швидше завантажуються та реагують на дії користувача застосунку.

Завдяки тому, що веб-браузер працює з користувачем через веб-браузер, то немає проблем із тим, що можуть бути закриті брандмауером деякі протоколи та порти. Якщо брандмауер клієнта дозволяє завантажувати веб-сторінки, то у користувача веб-застосунку не виникне проблем із його завантаженням.

Через те, що архітектура веб-застосунку для звичайного користувача є невидимою, це дає можливість розробнику масштабувати обчислювальні потужності застосунку без втручання у користувацькі пристрої. Наприклад непомітно для користувача можливо модернізувати або змінювати апаратне забезпечення серверу на якому знаходиться веб-додаток, розподіляти навантаження на застосунок між декількома серверами та багато чого іншого.

Зрозуміло, що веб-застосунки не мають таких можливостей, які мають звичайні настільні застосунки. Наприклад за допомогою веб-додатку користувач не зможе із браузеру керувати апаратними компонентами пристрою, працювати зі складною тривимірною графікою та моделями. Вищесказане є актуальним на сьогодення та найближче майбутнє, адже у наш час веб-технології розвиваються досить швидко.

Дивлячись на те, як змінюються та розвиваються веб-технології можна сказати, що веб-застосунки є технологією майбутнього. Вони почали стрімко розвиватися та наближуватися до настільного програмного забезпечення, яке користувачі звикли використовувати мало не щодня. Прикладом такого програмного забезпечення є Office Online від компанії Microsoft. Це пакет офісних застосунків, який являє собою веб-застосунок. Він дозволяє користувачам створювати та редагувати файли [11]. По суті це є аналогом настільного програмного забезпечення Microsoft Office і являється облегшеною версією. Office Online не потребує інсталяції на пристрій користувача, тому це дає можливість працювати з документами з будь-якого пристрою та з будь-якого місця.

1.3 Переваги та недоліки веб-застосунків

На стадії проектування нового застосунку, перед розробником насамперед стає задача, яка являє собою визначення з типом застосунку. На цьому етапі розробник має декілька варіантів. Одним із варіантів на даному етапі є розробка застосунку, який спеціально розроблений під дану програмну та апаратну платформу з використанням рекомендованих для даної платформи компіляторів та середовищ розробки. Іншим варіантом є розробка веб-застосунку, який має особливість таку, як незалежність від програмної та апаратної платформи користувацького пристрою. Саме від вибору розробника на даному етапі будуть залежати всі наступні етапи у розробці застосунку. Даний етап у розробці насправді є дуже важливим. Саме на цій стадії визначаються з тим щоб залучити працівників, які орієнтовані на розробку під обрану платформу та саме на цій стадії можна зрозуміти приблизну вартість проєкту.

Починаючи говорити про нативні та веб-застосунки важко не згадати про гібридні рішення. Для прикладу візьмемо Adobe PhoneGap. Даний програмний комплекс дає можливість написати застосунок з використанням скриптової мови програмування. Застосунок написаний з використанням цього середовища запускається у браузері, який поставляється разом з написаним застосунком та являє собою цілісний нативний застосунок. Платою за незалежність застосунку від програмної та апаратної платформи є більші вимоги до кількості пам’яті та обчислювальної потужності, у порівнянні із нативним застосунком. Якщо використовувати інструменти гібридної розробки, то у будь-якому випадку лише частина розробленого застосунку буде універсальною. Окрім того, може виникнути складність у створенні необхідного елементу для користувацького інтересу на розробку якого потрібно витратити значну кількість часу. У таких випадках може бути простіше розробити нативний застосунок.

Веб-технології у сьогоденні служать яскравим доказом того, що технологія, яка одного разу була вдало створена може застосовуватися у найрізноманітніших ситуаціях різними людьми.

Різницею нативного застосунку та веб застосунку являється те, що нативний застосунок взаємодіє з операційною системою, яка встановлена на пристрої, а веб-застосунки взаємодіють лише з браузером, який встановлено на користувацькому пристрої. Дана особливість значно розширює аудиторію, яка може використовувати застосунок, оскільки користувач отримує можливість використовувати застосунок на будь-якому користувацькому пристрої. Мова розмітки гіпертексту HTML дозволяє робити адаптивну розмітку, тобто підлаштовувати дизайн та елементи взаємодії під розміри екрану на користувацькому пристрої. Але незважаючи на велику перевагу веб-застосунків у тому, що вони є незалежні від апаратної та програмної платформи головним недоліком веб-застосунків являється їх значно нижча продуктивність, ніж у нативних та оптимізованих під конкретну платформу застосунків.

Перевагою нативних застосунків є те, що вони на мають на порядок вищу продуктивність. Окрім більшої продуктивності при розробці нативного додатку розробник може використовувати усі специфічні можливості операційної системи та платформи в цілому. Значним недоліком нативних застосунків є те, що вони мають змогу працювати лише на одній апаратній та програмній платформі, якщо спеціально не адаптувати їх під різні платформи. Тобто, якщо з'являється вимога у перенесенні свого застосунку на інші пристрої з іншими операційними системами, то розробнику потрібно буде повністю переписати та адаптувати код застосунку під іншу програмну та апаратну платформу. У зв'язку з цим потрібна більша команда розробників, що тягне за собою збільшення витрат на розробку. Саме через те, що нативні застосунки мають більше можливостей для взаємодії з програмними та апаратними складовими пристрою користувача, багато по справжньому потужних програм доступні лише на одній апаратній та програмній платформі.

Нативні застосунки відрізняє від веб-застосунків те, що даний застосунок зберігається на пристроях користувачів. Хоч у наші дні більшість людей вже достатньо добре знайомі із завантаженням нативних додатків, але розробник ніяк не може бути впевнений, що кожен користувач буде використовувати однакову версію застосунку.

Нативні застосунки у порівнянні з веб-застосунками мають безліч переваг та недоліків. Оскільки нативні застосунки напряму працюють із операційною системою та апаратним забезпеченням пристрою користувача, то розробнику значно простіше використовувати апаратні компоненти такі, як мікрофон, камера, служби для роботи з місцезнаходженням пристрою та багато інших. Ще одною перевагою нативних застосунків є те, що більшість таких додатків розміщуються на різноманітних майданчиках, так званих магазинах застосунків. Це є перевагою тому, що перед публікацією такого застосунку у магазині, він проходить перевірку і у зв'язку з цим набагато менша вірогідність того, що застосунок буде шкідливим та не сумісним з пристроєм користувача. Але наряду з перевагами нативних

застосунків також вони мають і деякі недоліки. Як правило розробка нативного застосунку являється більш дорогою. Це більш актуально для розробника застосунку, який хоче, щоб він був сумісний з декількома апаратними платформами. Також окрім розробки, значно більша і вартість обслуговування та оновлення застосунків, особливо різниця помітна, якщо застосунок доступний хоча б для декількох платформ. Також недоліком таких застосунків є те, що користувачі використовують різні версії продукту, через це з'являються проблеми із розробкою, а також сумісності з різними версіями застосунку.

Якщо порівнювати веб-застосунок із нативним застосунком, то можемо сказати, що перші набагато простіші у використанні у зв'язку з тим, що у будь-який час та з будь-якого пристрою користувач може отримати доступ до актуальної версії застосунку.

Звичайно як і у нативних застосунків, так і у веб-застосунків також є свої переваги та свої недоліки. Однією із переваг веб-застосунків є те, що їх набагато простіше обслуговувати тому, що вони мають спільну кодову базу для всіх пристроїв, а також їх можливо достатньо легко адаптувати для сумісності з будь-якими старими та не актуальним пристроями. Веб-застосунки не потрібно повністю завантажувати на користувацький пристрій, тому найбільш актуальна ця перевага є для пристроїв у яких обмежений об'єм накопичувача. Через те, що такий застосунок працює у веб-браузері для розробника не важливо на якій апаратній платформі пристрій користувача. У зв'язку із цим використовувати застосунок можливо на всіх пристроях, де встановлений браузер. Через те, що сам додаток зберігається на віддаленому сервері, клієнт лише завантажує інтерфейс користувача. Це дає можливість використовувати єдину версію застосунку, тому із цим зникає проблема із сумісністю різних версій продукту. Хоча у сьогоденні вже багато людей добре знайомі із електронними пристроями та користуються ними мало не щодня, частка досвідчених користувачів не велика. У зв'язку із цим значною перевагою веб-застосунків є те, що застосунок не потрібно встановлювати

на свій пристрій та налаштовувати адже іноді налаштування деяких застосунків є не простою задачею.

Наряду із перевагами веб-застосунків є у деяких випадках критичним недоліком те, що такі застосунки мають дуже обмежений доступ до апаратних складових користувацького пристрою. Крім того для роботи із веб-застосунком потрібен доступ до інтернету, без нього користувач не зможе працювати із застосунком адже він розташований на віддаленому сервері, а користувач отримає лише користувацький інтерфейс.

Висновки

У даному розділі було розглянуто історію розвитку, основні поняття і принципи роботи веб-технологій, їх переваги та недоліки.

Досліджуючи історію розвитку веб-технологій, можемо зробити висновок, що вони стрімко входять у наше життя. На початку їх зародження ніхто не міг уявити, що веб переросте у дещо більше аніж звичайні статичні сайти, які наповнюють контентом лише адміністратори сайту та вони не мають ніякої інтерактивності. У сьогоденні ж веб-сайти на стадії, коли не тільки адміністратори ресурсу можуть генерувати контент, але і самі користувачі.

При розгляді принципу роботи було з’ясовано, що веб-застосунок складається із серверної та клієнтської частини. На серверна частина складається із ПЗ веб-серверу, яке приймає HTTP запити, а також на сервері розташовується сама логіка веб-застосунку написана на серверній мові програмування.

Основними превагами використання веб-застосунків є простота у використанні, можливість використовувати їх на різних платформах та відсутність проблем із закритими портами, адже зазвичай порти на яких працюють веб-застосунки не закриті фаєрволом.

2. ОГЛЯД ТА ВИЗНАЧЕННЯ З ПРОГРАМНИМИ ЗАСОБАМИ ТА МОВАМИ ПРОГРАМУВАНЯ ДЛЯ РЕАЛІЗЦІЇ ПРОЄКТУ

2.1 Визначення із засобами для розробки програмного комплексу

З розвитком інформаційних технологій, стали більш доступні для звичайного користувача електронні пристрої такі як ПК, ноутбуки, смартфони. У зв'язку із цим у розробника з'являється необхідність у тому, щоб його продукт був доступний на більшості сучасних пристроях. Проблемою є не лише те, що пристрої відрізняються своєю апаратною складовою, але і те, що на одній апаратній платформі можуть використовуватися різні операційні системи. Цю проблему вирішує мультиплатформність.

Мультиплатформністю називається властивість, що дозволяє досягти забезпечення роботи програмного забезпечення більш ніж на одній програмній або апаратній платформі [12]. Прикладом мультиплатформності є програмне забезпечення, яке розроблене та призначене для одночасної роботи в операційних системах Windows та Linux.

У сьогоденні більша частина актуальних високорівневих мов програмування є мультиплатформними. Для таких мов реалізовані перекладачі, які дозволяють програмі працювати на різних платформах. У якості прикладу візьмемо такі мови як C, C++ та Pascal. Вони являють собою мультиплатформні мови на рівні компіляції, простіше кажучи для них існують компілятори під різні платформи. Наряду із вищезгаданими мовами, ще існують такі мультиплатформні мови програмування на рівні виконання, як C# та Java. Це означає, що є можливість запуску їх виконавчих файлів на різних платформах без необхідності у попередній перекомпіляції програми. Однак для того, щоб програма могла виконуватися на різних платформах у деяких випадках є необхідність у встановлені додаткового ПЗ, наприклад середовище виконання Java Virtual Machine для застосунків написаних на мові Java.

Отже у розробці програмного комплексу візуалізації даних СТРП буде використано веб-технології. Завдяки використанню веб-технологій буде створено веб-застосунок, який працюватиме на більшості користувацьких пристроях. Використання веб-застосунку передбачає, що у користувача вже встановлений в ОС веб-браузер. Перевагою використання цієї технології є те, що відсутня необхідність компілювати застосунок під кожну платформу. А також важливою властивістю є те, що немає необхідності у встановленні додаткових програмних компонентів до ОС на користувацькому пристрої, адже єдиний компонент (веб-браузер), який необхідний для роботи застосунку встановлений за замовчуванням у більшості ОС.

У розробці серверної частини застосунку будемо використовувати скриптову мову програмування PHP. Дана мова програмування була обрана через те, що вона має значну кількість переваг. PHP є дуже популярною, адже вона має простий та зрозумілий синтаксис, який може дуже швидко освоїти та почати писати код навіть новачок. Значною перевагою у використанні мови PHP при розробці веб-застосунку є те, що дана мова із самого початку зародження розроблювалася для роботи в інтернеті, тому у неї передбачено багато особливостей у роботі в парі із HTML розміткою. Наприклад PHP код може бути вставлений безпосередньо у код сторінки HTML, який буде коректно оброблюватися інтерпретатором мови PHP. Завдяки даній особливості можемо зручно використовувати PHP у формуванні документів HTML, що дозволяє додати динамічності та гнучкості статичним сторінкам.

Головним компонентом веб-застосунку є веб-сервер. З точки зору «заліза» веб-сервер – це комп’ютер підключений до мережі інтернет, який зберігає файли сайтів та доставляє їх на пристрої користувача. Якщо дивитися з точки зору програмного забезпечення, то веб-сервер поєднує у собі декілька компонентів, які контролюють доступ клієнтів до файлів, що розташовані на сервері. Головним компонентом є HTTP-сервер – частина ПЗ, яка працює з URL-адресами та самим протоколом HTTP [13].

У якості програмного забезпечення веб-серверу будемо використовувати Apache HTTP-сервер. Веб-сервер Apache – один із найпопулярніших веб-серверів. Він є повністю безкоштовним та з відкритим початковим кодом. Головною його рисою є те, що він являє собою мультиплатформне ПЗ, яке підтримує усі популярні операційні системи такі, як Windows, Linux, BSD, Mac OS та інші. Apache також має ще безліч особливостей, переваг та недоліків, які будуть розглянуті у наступному підрозділі 2.2.

Важко уявити у сьогоденні веб-сайт чи веб-застосунок, який не використовує базу даних. Використання баз даних стало доступним з появою скриптової мови програмування PHP. Це надало можливість розробникам розділити каркас сторінок сайту від його контенту, який можна зберігати в базах даних. Така особливість полегшує адміністрування ресурсу, вдосконалення його функціоналу та дизайну. Переваги використання бази даних є достатньо очевидними. З використанням БД легко та швидко вирішуються такі питання, як сортування, пошук, пагінація змісту. Для роботи з БД у розробці програмного комплексу візуалізації даних СТРП будемо використовувати систему управління БД MySQL. Адже вона є найбільш популярною та універсальною. Перевагою MySQL є те, що вона дозволяє швидко оброблювати інформацію, є повністю безкоштовною та має відкритий початковий код. Завдяки цим рисам MySQL і стала однією із найпопулярніших СУБД.

2.2 HTTP-сервер

Основним компонентом веб-застосунку і веб-сайтів загалом є веб-сервер. Жоден веб-сайт не може працювати без веб-серверу.

Поняття веб-серверу можемо поділити на дві складові: апаратна складова та програмна. З точки зору апаратної складової веб-сервер являє собою звичайний комп’ютер, який зберігає на своїх накопичувачах файли сайту такі, як сторінки HTML, таблиці стилів CSS, файли JavaScript, зображення та інші файли. Він не просто зберігає файли сайту, а і доставляє їх на пристрої кінцевих користувачів, адже він повинен бути підключений до мережі інтернет та доступний через доменне ім’я, наприклад nupp.edu.ua. З точки зору програмної складової веб-сервер

являє собою набір компонентів, які виконують контроль доступу користувачів до ресурсів, що розташовані на веб-сервері. Головним програмним компонентом веб-серверу є HTTP-сервер – компонент, який працює із HTTP протоколом та розуміє URL-адреси [13].

Вся суть веб-серверу полягає у тому, щоб URL-адресу перекласти у ім’я файлу, який розташований на даному веб-сервері або ж у ім’я так званої програми. У першому випадку веб-сервер просто відправить запитуваний файл, тому хто його запитував. У другому випадку, коли користувач звертався до програми або ж скрипту на сервері, він отримає відповідь від серверу із результатом виконання запитуваної програми. Вищесказане являє собою головну частину функцій веб-серверу, а все інше являється допоміжними компонентами.

Коли користувач запускає на своєму пристрої браузер і звертається до якоїсь URL-адреси, наприклад https://nupp.edu.ua – клієнт надсилає повідомлення машині через мережу інтернет, яка знаходиться по даній адресі. Веб-сервер повинен бути завжди у роботі, підключений до мережі інтернет і готовий прийняти повідомлення від користувача та діяти у відповідності до нього. Для наочності на рисунку 2.1 приведемо схему взаємодії веб-серверу із клієнтом:

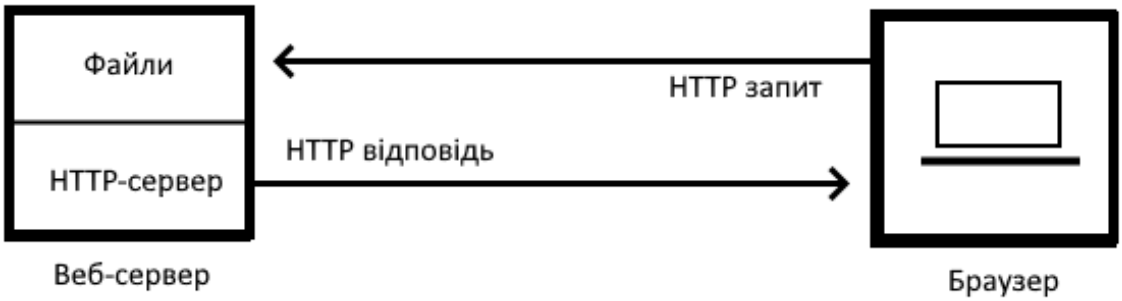

Рисунок 2.1 – Схема взаємодії веб-серверу з клієнтом

URL – це абревіатура від Uniform Resource Locator, яка в перекладі з англійської означає єдиний вказівник ресурсів [14]. URL-адреса має свою певну

уніфіковану структуру. Розглянемо структуру на прикладі URL-адреси https://nupp.edu.ua/. Дана адреса складається із трьох частин:

<схема>://<хост>/<шлях>

<схема> – схема звернення до ресурсу, найчастіше мається на увазі мережевий протокол, у нашому випадку це протокол «https».

<хост> – ім’я хоста в системі доменних імен або IP-адреса хоста, у нашому прикладі це «nupp.edu.ua»

<шлях> – інформація, яка уточнює інформацію про місцезнаходження ресурсу, у прикладі це «/», що зазвичай означає головну сторінку ресурсу.

Приведена вище структура URL-адреси є найбільш поширеною. Для повного розуміння структури приведемо повну структуру URL [15]:

<схема>:[//[<логін>[:<пароль>]@]<хост>[:<порт>]][/<шлях>][?<параметри>][#<якір>]

Після того, як користувач відправив запит до серверу, то він надходить на порт 80, який є портом HTTP за замовчуванням. Запит складається із чотирьох частин: метод (GET, PUT, POST, DELETE, CONNECT), уніфікований ідентифікатор ресурсу, версія протоколу та ряд HTTP заголовків. Після отримання повідомлення, веб-сервер який працює на цьому хості, повинен обробити отримане повідомлення. На рисунку 2.2 приведено структуру HTTP запиту:

Рисунок 2.2 – Структура HTTP запиту

У ролі веб-серверу може виступати як і домашній комп’ютер, так і цілий кластер потужних комп’ютерів. У будь-якому випадку на комп’ютері, який виконує роль веб-серверу повинен бути запущений ПЗ веб-серверу, яке у свою чергу працює з мережею, приймає та обробляє мережеві запити [14].

Отже як було з’ясовано, для того, щоб комп’ютер виконував роль веб-серверу необхідно встановити спеціалізоване ПЗ, тому необхідно обрати ПЗ веб-серверу, яке будемо використовувати у своєму проєкті. Для розробки програмного комплексу візуалізації СТРП розглянемо декілька найбільш популярних на сьогоднішній день веб-серверів.

Nginx – це HTTP-сервер, а також зворотній проксі-сервер, TCP/UDP проксі-сервер загального призначення, а також поштовий проксі-сервер [16]. Даний веб-сервер являється розробкою російського розробника Ігора Сисоєва. Спочатку його розробляли для високонавантаженого ресурсу Rambler, а на сьогоднішній день цим ПЗ користуються деякі популярні сайти у світі, такі як Netflix, WordPress, Facebook та інші. Станом на квітень 2021 року, Nginx обслуговував 22.86% одних з найбільш навантажених ресурсів [16-17]. Дане серверне ПЗ від самого початку розроблялося лише для UNIX подібних ОС та успішно працював на таких системах, але він не був доступний для машин, які працювали під управлінням ОС Windows. Лише після виходу версії 0,7.52 Nginx став доступний для інсталяції на сервери, які працюють під ОС Windows, однак продуктивність Nginx на Windows нижча ніж на Unix подібних ОС. Використання Nginx є найбільш доцільною при наявності на ресурсі великої кількості статичного контенту або файлів для завантаження. Більш продуктивним у порівнянні з іншими продуктами дане ПЗ є зі статичним контентом завдяки тому, що він використовує асинхронні сокети і не створює на кожен запит новий процес. Одного процесу на ядро достатньо для обробки та підтримки тисяч одночасних з’єднань, що дозволяє значно знизити навантаження на апаратну складову серверу [17]. Nginx можна використовувати, як інтерфейсний проксі для інших веб-серверів. Часто можна зустріти у високонавантажених проєктах коли поєднують Nginx та Apache для отримання максимальної продуктивності у роботі

веб-застосунку. Для більшої наочності на рисунку 2.2 приведемо схематичне зображення роботи Nginx та Apache у парі:

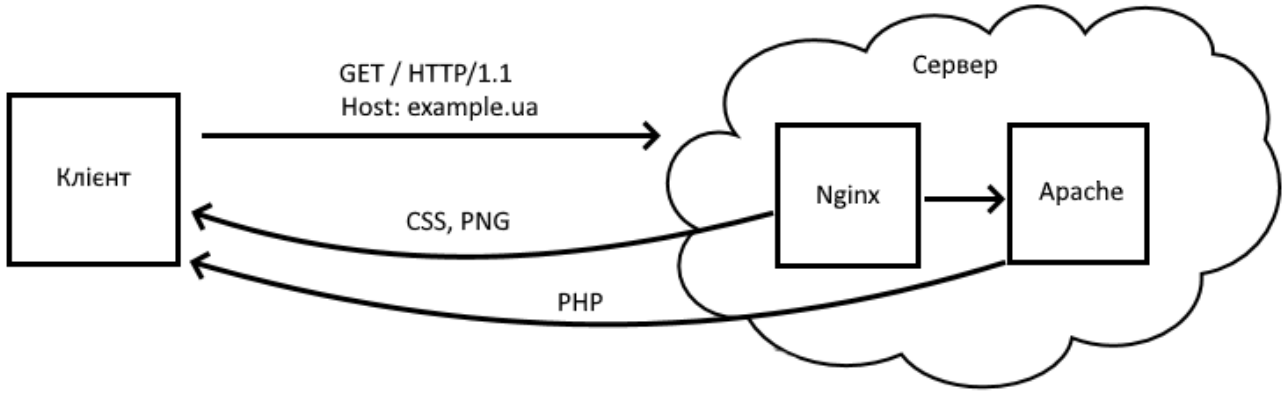

Рисунок 2.2 – Схема роботи у парі Nginx та Apache

Apache HTTP-сервер являє собою безкоштовний веб-сервер з відкритим початковим кодом. Даний веб-сервер на даний момент є одним із найпопулярніших в усьому світі. Через те, що у Apache відкритий початковий код, він може бути досліджений будь-ким. Дана особливість дає змогу тисячам людей із усього світу перевірити його на вміст помилок. Завдяки тому, що постійно велика кількість людей перевіряє код даного ПЗ, можемо припустити, що воно є більш безпечним у порівнянні з комерційним продуктом. Ще одною перевагою у відкритості початкового коду є те, що кожен може взяти початковий код даного ПЗ та змінити його під свої специфічні потреби. Зокрема у Apache існує можливість розширення базового функціоналу завдяки підтримки сторонніх модулів. Apache підходить для проєктів будь-якого розміру та типу. На ньому можна запустити, як і простий сайт-візитку, який складається з декількох сторінок, так і великий сайт, що обслуговує тисячі постійних користувачів. Якщо порівняти по продуктивності при роботі з динамічними даними Apache та Nginx, то вони є приблизно на одному рівні продуктивності. Але Apache програє у продуктивності при передачі великої кількості статичних файлів. Архітектура даного ПЗ складається із двох компонентів – ядро та модулі. У теорії ядро може працювати навіть без модулів але у такому випадку його функціональна частина буде сильно обмежена.

Завдяки модульної структури для збільшення функціональних можливостей можливе встановлення додаткових модулів. А якщо потрібно підвищити продуктивність свого веб-серверу, то існує можливість вимкнути модулі, які не використовуються. Для даного веб-серверу існує велика кількість модулів, які дозволяють додати підтримку різноманітних мов програмування. Прикладом таких модулів є «mod_php» для мови PHP, «mod python» для мови Python, «apache-ruby» для мови програмування Ruby та багато інших. У нашому проєкті буде використано ПЗ веб-серверу Apache, мову програмування PHP та відповідно модуль «mod_php» [14].

2.3 Скриптова мова загального призначення PHP

Одна із найпопулярніших мов програмування у створенні веб-застосунків PHP розпочала свій розвиток ще у далекому 1994 році. Спочатку ця мова розроблялась лише для того, щоб з’явилась змога відслідкувати відвідувачів домашньої сторінки з резюме розробника PHP. Абревіатура PHP на початку розвитку технології розшифровувалася як Personal Home Page. Але функціональність даної технології значно розширювалась, у зв’язку із чим розробники почали використовувати PHP для створення набагато складніших сайтів. З того часу абревіатуру почали розшифровувати як PHP: Hypertext Processor, що у перекладі означає гіпертекстовий процесор.

PHP являє собою продукт із відкритим початковим кодом, що дає змогу будь-кому змінювати та розповсюджувати початковий код іншим користувачам або підприємствам.

Мова програмування PHP є схожою на такі мови, як C та Perl, адже велика кількість конструкцій мови було запозичено із цих мов. Через схожість на інші мови програмування значно зменшується складність у вивченні PHP за умови, що розробником вже було вивчено деякі мови програмування. Однією із головних рис даної мови є простота. При написанні сценарію на мові PHP не важливо зі скількох рядків він складається, з одного або ж із тисяч, у будь-якому випадку відсутня

необхідність задавати спеціальні параметри компіляції або завантажувати додаткові бібліотеки.

Написаний код на мові програмування PHP завжди виконується та зберігається на стороні серверу, який у відповідь надсилає браузеру HTML сторінку. Коли користувач звертається до веб-ресурсу, то запит направляється до серверу на якому зберігається інформація. У свою чергу після отримання запиту від користувача, на стороні серверу починає виконуватися код PHP у відповідності до заданим у скрипті командам. В основному результатом виконання PHP скрипту є генерація HTML сторінки з динамічними даними які наприклад зберігаються в БД. На рисунку 2.3 приведено схематичне зображення взаємодії користувача із сервером, який використовує модуль PHP.

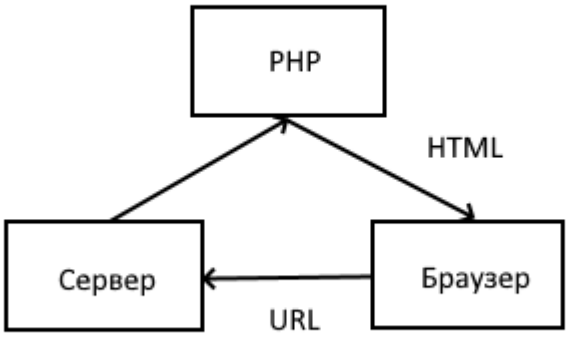

Рисунок 2.3 Схема взаємодії клієнту, серверу і модуля PHP

Якщо веб-ресурс являє собою статичні документи HTML, то на запит користувача він отримує у відповідь заздалегідь написану статичну HTML сторінку. При цьому на стороні серверу не відбувається інтерпретації даних. Приведемо схематичне зображення взаємодії користувача і серверу без модуля PHP на рисунку 2.4.

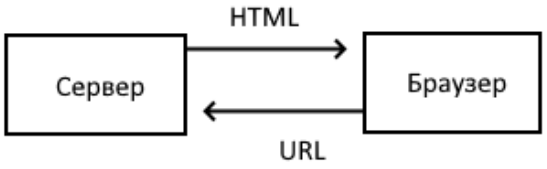

Рисунок 2.3 Схема взаємодії клієнту, серверу і модуля PHP

Серверну частину веб-застосунку можна розробити не лише на мові PHP, а і з використанням інших мов програмування таких, як Perl, Java та інших. Але перевагою використання у проєктах мови PHP є те, що дана мова із самого початку розроблялася спеціально для створення динамічних веб-сторінок, що передбачає виконання саме цих задач значно швидше та ефективніше ніж альтернативні серверні мови програмування. Однак використання мови PHP у деяких проєктах не є оптимальною, адже з використанням PHP не завжди вдається робити те, що можливо наприклад на мовах Perl або Java [18].

Одним із важливих критеріїв вибору засобів для розробки застосунків являються реалізовані механізми безпеки. У PHP засоби безпеки знаходяться під управлінням адміністратора серверу, що дає можливість забезпечити максимальну свободу дій та безпеку, при розумному налаштуванні його. PHP має багато засобів безпеки. Наприклад для того, щоб запобігти негативних впливів на швидкодію серверу у адміністратор наявна можливість обмежити максимальний час на виконання сценарію та використання пам’яті ним. Для запобігання перегляду конфіденційної інформації або виконання деяких сценаріїв адміністратор серверу має можливість обмежувати доступ до каталогів.

Завдяки тому, що PHP не є орієнтованим для роботи із конкретним веб-сервером, розробники можуть використовувати його із найрізноманітнішими серверами. PHP чудово працює на більшості популярних серверів таких, як Nginx, Microsoft IIS, Apache та інших. Враховуючи те, що ці сервери працюють на різних

платформах, можемо зробити висновок, що PHP є незалежною мовою, яку можна використовувати на різних платформах без змін у коді [18].

Основними перевагами використання мови програмування PHP є:

- Швидкодія. Оскільки PHP із самого початку розроблявся для використання його у створенні веб-сайтів, він є найбільш ефективним у роботі з ними.
- Безпека. При правильній розробці та налаштуванню PHP, його програмний код буде не доступний для перегляду користувачеві.
- Універсальність. PHP без проблем можна використовувати на різних платформах та ОС, таких як Linux, Windows, Mac OS та багато інших.
- Простота. Синтаксис даної мови програмування є досить простим для розуміння людиною. Завдяки тому, що мова розроблялась для веб, код PHP можна легко вбудувати у структуру HTML сторінки.
- Підтримка. Через велику популярність даної мови, у мережі можна знайти велику кількість потрібної інформації та підтримку від інших розробників. Все це можна отримати навіть на офіційному веб-сайті PHP.

Отже, враховуючи усі переваги, для розробки програмного комплексу візуалізації даних СТРП було обрано скриптову мову програмування PHP.

2.4 Мова гіпертекстової розмітки HTML

При розробці користувацької частини проєкту було використано мову розмітки гіпертексту HTML.

HTML – мова гіпертекстової розмітки, яка являє собою стандартизовану мову розмітки документів у мережі. Використання даної мови дає можливість визначити зміст та структуру веб-документу.

Гіпертекстом називають посилання, які зв’язують веб-сторінки між собою. Посилання можуть зв’язувати документи, як і в межах одного сайту так і між різними веб-сайтами.

У створенні HTML документу використовують розмітку для відображення у веб-браузері контенту, такого як текст, зображення та інше. Можемо сказати, що HTML документ є звичайним ASCII-файлом до якого додано керуючі HTML теги. Для виділення HTML елементу із іншого тексту у документі використовують теги, які являють собою ім’я елементу оточеного у кутові дужки «<» та «>». У більшості випадків теги HTML-документів є простими та інтуїтивно зрозумілими для розробника, адже вони створені із англомовних слів, які є загальновживаними [19].

На відміну від звичайного тексту у гіпертексті наявна можливість вбудувати у вміст сторінки спеціальних конструкцій мови HTML, за допомогою яких користувач може перейти до іншого документу – гіперпосилань.

HTML – не є мовою програмування. Її використовують лише для розмітки веб-сторінок. Для відображення сторінки браузер розпізнає параметри, які при створенні розмітки були присвоєні групі або окремому елементу.

У створенні HTML-документів немає необхідності обов’язково використовувати якийсь спеціальний інструмент чи ПЗ для розробки візуального представлення веб-сторінки, можна навіть використовувати простий текстовий редактор такий, як Блокнот. Кожен розробник обирає зручний для себе інструмент для розмітки веб-сторінок. У сьогоденні існує велика кількість редакторів початкового коду, у яких наявна підтримка мови HTML. Вони можуть бути як і візуальним редактором, який дозволяє збирати веб-сторінку як конструктор, так і текстовим редактором орієнтованим на роботу з мовою розмітки HTML, таблицями стилів CSS та іншими технологіями для створення веб-сторінок. Текстові редактори, що орієнтовані на розробку веб-сторінок мають багато особливостей. Наприклад у таких редакторів присутнє виділення помилок, автоматичне доповнення коду, що дозволяє значно пришвидшити розробку.

2.5 Каскадні таблиці стилів CSS та фреймворк Bootstrap

При розробці користувацьких веб інтерфейсів є необхідність у стилістично оформленому відображенні елементів управління та контенту. Для таких цілей використовують CSS (Cascading Style Sheets) – формальну мову опису зовнішнього вигляду документа, який написано за допомогою мови розмітки.

Основним завданням CSS є можливість розділення опису логічної структури розмітки сторінки від опису її зовнішнього вигляду. Таблиці стилів використовують для задання параметрів шрифтів, кольорів, позицію блоків та багатьох інших елементів, які представлені на веб-сторінці. Поділення структурних елементів від стилів у документі дає значно складність коду та його повторюваності у структурному вмісті. Також це дає можливість надати велику гнучкість та можливість його подання.

Для того, щоб браузер міг поєднати документ розмітки та таблиці стилів, необхідно для нього вказати до них шлях. При використанні CSS у розробника є декілька способів писати код стилів: у самому документі розмітки, зберігати код у окремому документі, який прив’язаний до документу розмітки або ж є можливість поєднати ці два способи.

Для більшої наочності приєднання стилів приведемо декілька прикладів:

- варіант, коли опис стилів знаходиться у окремому від документу розмітки файлі з використанням елементу <link>:

```
<!DOCTYPE html>
<html>
  <head>
   .....
   <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
   .....
  </body>
</html>
```

- Варіант, коли стилі описані всередині документу розмітки та включені в елемент <style>, який у свою чергу включений до елементу <head>:

```
<!DOCTYPE html>
<html>
  <head>
   .....
   <style>
     body {
      color: blue;
     }
   </style>
  </head>
  <body>
   .....
  </body>
</html>
```

На сьогоднішній день важко уявити веб-сайт, який не використовує CSS. Вони набули великої популярності завдяки тому, що CSS допомагають заощадити зусилля та час на верстку веб-сайтів завдяки можливості гнучких налаштувань стилів елементів, які відображаються на сторінці. Використання при верстці CSS значно розширює можливості стандартного HTML, а також робить розробку дизайну сторінки більш зручною та точною. Насамперед зручність використання CSS полягає у тому, що є можливість відділення оформлення сайту від його вмісту [20].

Одними із найголовніших преваг CSS є:

- Легка підтримка коду
- Швидке завантаження
- Можливість використання одних і тих же правил стилів для безлічі документів

У верстці сучасних веб-ресурсів головною характеристикою є привабливий зовнішній вигляд сторінки. Для спрощення верстки сторінок веб-сайтів почали з’являтися різноманітні CSS фреймворки. Фреймворк являє собою каркас програмної системи, який призначений для пришвидшення розробки. У сьогоденні існує велика різноманітність різних популярних CSS фреймворків. Розглянемо один із найпопулярніших – Bootstrap.

Bootstrap – це продукт із відкритим початковим кодом, який являє собою набір інструментів для створення веб-ресурсів. Він містить у собі по суті готові шаблони стилів для різних елементів HTML документу таких, як форми, кнопки, елементи навігації та багато інших. Головна задача фреймворку спростити верстку динамічних веб-сторінок[21].

Великою перевагою Bootstrap являється його велика популярність серед розробників веб-сайтів та веб-застосунків. У зв’язку із цим у мережі існує велика кількість різноманітних матеріалів, які допоможуть розробнику зрозуміти принципи верстки сторінок із використанням даного фреймворку.

Особливістю фреймворку є те, що завдяки наявній дванадцяти колонковій сітці у розробника наявна можливість верстати адаптивні веб-сторінки значно простіше та швидше ніж без використання Bootstrap. Для більшого розуміння, що являє собою сітка Bootstrap її можна порівняти із звичайною таблицею. Приведемо наочний приклад сітки Bootstrap таблицею 2.1:

Таблиця 2.1 – Приклад використання сітки Bootstrap

<table>
<tr><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td><td>.col-md-1</td></tr>
<tr><td colspan="8">.col-md-8</td><td colspan="4">.col-md-4</td></tr>
<tr><td colspan="4">.col-md-4</td><td colspan="4">.col-md-4</td><td colspan="4">.col-md-4</td></tr>
<tr><td colspan="6">.col-md-6</td><td colspan="6">.col-md-6</td></tr>
<tr><td colspan="12">.col-md-12</td></tr>
</table>

Із ростом популярності Bootstrap та появою найрізноманітніших веб-сайтів із його використанням, розробники фреймворку зрозуміли, що у деяких проєктах використання усіх його компонентів є надлишковим. У зв’язку із цим на офіційному сайті Bootstrap з’явилася можливість зібрати дистрибутив , який дозволяє включити до нього лише необхідні елементи для реалізації свого проєкту. Дана можливість дає змогу розробнику веб-ресурсів таким чином оптимізувати швидкість завантаження, та швидкодію користувацького інтерфейсу в загалом,

завдяки тому, що не завантажуються елементи фреймворку, які не використовуються у розроблюваному проєкті.

Перевагами та недоліками даного фреймворку є:

Переваги:

- Добре розроблена документація
- Можливість простої побудови адаптивного сайту
- Підтримка більшістю сучасними браузерами
- Швидкість розробки

Недоліки:

- Завдяки великій популярності фреймворку у мережі вже існує багато сайтів, які дуже схожі між собою. Через це складно створити насправді унікальний дизайн.
- Велика різноманітність та кількість класів
- Розмір фреймворку. Фреймворк є достатньо великим, що значно сповільнює швидкість завантаження сторінки.

Отже при розробці клієнтської частини нашого проєкту однозначно буде використано CSS, адже у сучасному світі є дуже важливо створювати привабливі клієнтські інтерфейси. Для пришвидшення розробки та верстки дизайну буде використано CSS фреймворк, який має назву Bootstrap. Завдяки ньому з’являється можливість створити привабливий та адаптивний дизайн у максимально короткі терміни. Головними причинами вибору фреймворку Bootstrap є можливість використання у проєкті лише потрібних елементів та наявність зручної системи сіток, яка дозволяє дуже швидко та із мінімальними зусиллями створити адаптивний дизайн.

2.6 Мова програмування JavaScript та технологія AJAX

Із розвитком веб-технологій з’явилася проблема із необхідністю створення динамічних веб-сторінок. У ті часи веб потребував у створенні легкої скриптової мови, яка має можливість працювати з DOM. Рішенням цієї проблеми стало створення скриптової мови програмування JavaScript.

JavaScript – це інтерпретована мова програмування із об’єктно-орієнтованими можливостями [22]. Мова JavaScript не лише синтаксично є дуже схожою на такі мови програмування, як Java, C, C++, а і своїми програмними конструкціями такими, як if, цикли while та оператор &&. Через те, що JavaScript є нетипізованою мовою, то при розробці відсутня необхідність визначати типи змінних.

У ядра мови JavaScript присутня підтримка засобів роботи із такими простими типами даних, як булеві значення, числа та строки. Окрім цього у ньому є вбудована підтримка роботи із масивами, датами та об’єктами регулярних виразів.

Зазвичай JavaScript використовують на клієнтських пристроях у веб-браузерах, що дозволяє зробити взаємодію користувача із веб-сторінкою більш динамічною. Завдяки використанню JavaScript розробник має можливість керувати веб-браузером та змінювати зміст документу без перезавантаження сторінки.

Так, як мова JavaScript буде використана у нашому проєкті лише для розробки клієнтського інтерфейсу, то далі буде розглянуто тільки клієнтський JavaScript.

Клієнтський JavaScript дозволяє визначати поведінку статичного вмісту веб-сторінок, адже він дозволяє працювати із об’єктною моделлю документу (DOM), яка визначається браузером. Також він є основою для деяких технологій розробки веб-застосунків. Прикладом такої технології є AJAX, яка буде розглянута далі.

Досягти поліпшення інтерактивності веб-сторінок досягається за допомогою наступних можливостей мови JavaScript:

- Доступ до контенту на сторінці, що надає можливість знайти чи виділити необхідний елемент на сторінці за певним атрибутом або ідентифікатором, зчитати інформацію, яку було записано у текстове поле та інші.
- Зміна контенту. Використовується для динамічного додавання чи видалення тексту, атрибутів чи елементів зі сторінки.

- Програмування правил. Дана особливість дає можливість задати послідовність операцій, які необхідно виконати браузеру.
- Реагування на події. Дозволяє створити сценарій, який буде запущено лише після якоїсь конкретної події. Наприклад після натискання на кнопку.

Для вбудовування до веб-сторінки HTML необхідно використовувати теги <script> та </script> при цьому сам код сценарію розміщується між ними:

```
<script>
// код JavaScript сценарію
</script>
```

Також тег <script> підтримує атрибут src. У значенні цього атрибуту задається URL-адреса файлу, який містить JavaScript-код. Приклад використання:

```
<script src=”./js/script.js”></script>
```

HTML код сторінки може містити в собі необмежену кількість елементів <script>. Якщо у коді сторінки зустрічається декілька окремих сценаріїв, то вони будуть запускатися по порядку їх слідування у документі. Не дивлячись на те, шо окремі сценарії виконуються по мірі завантаження та аналізу браузером HTML коду, вони уявляють із себе частини однієї JavaScript програми. Дана особливість дає можливість змінним та функціям, які визначені у різних файлах буди доступними для всіх сценаріїв, що виконуються на сторінці.

У об’єктній моделі документу ключовим є об’єкт Window. Через нього доступні усі інші об’єкти. Наприклад, у будь-якому об’єкті Window міститься властивість document, яка посилається на пов’язаний із вікном об’єкт Document, та властивість location. Об’єкт Window являє собою глобальний об’єкт на самому початку ланцюжка області видимості та всі клієнтські об’єкти доступні, як властивості інших об’єктів. Така особливість означає, що у JavaScript існує ієрархія об’єктів, на початку якої знаходиться об’єкт Window [22]. Проілюструємо таку ієрархію на рисунку 2.4:

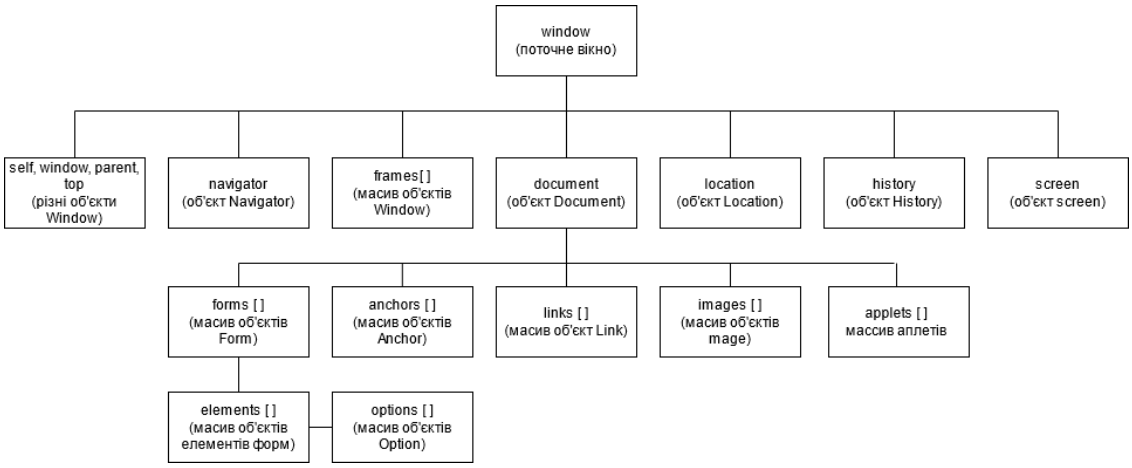

Рисунок 2.4 – Ієрархія об’єктів клієнтського JavaScript

Через велику популярність та широкі можливості мови JavaScript почали з’являтися технології на її основі. Прикладом такої технології, що основана на JavaScript являється AJAX.

AJAX розшифровується, як Asynchronous JavaScript and XML (асинхронний JavaScript та XML) [22]. Дана технологія по своїй суті являє собою сценарій написаний на мові JavaScript, який при необхідності виконує у фоновому режимі запити до серверу та отримує дані від нього. Можливість фонового запиту до серверу зазвичай використовують для оновлення окремих частин веб-сторінки, що дає можливість не перезавантажувати повністю її.

У часи до появи технології AJAX функціональні частини серверу та клієнту завжди розглядалися як незалежні, окремі частини, що працюють окремо один від одного при відповіді на дії користувача. Після появи даної технології з’явилась можливість так званого спілкування клієнту і серверу поки користувач працює із веб-сторінкою.

Веб-браузери, які існують у сьогоденні мають підтримку набору технологій, які потрібні для роботи AJAX. Завдяки даній особливості у користувача немає необхідності у встановленні якихось додаткових компонентів для взаємодії із сторінками побудованими з використанням AJAX. Дана технологія складається із таких компонентів:

- JavaScript – компонент без якого не можливо було б реалізувати функціональність на стороні клієнту.
- Об’єкт XMLHttpRequest. Він необхідний для того, щоб була можливість спілкування із сервером асинхронно, що дає можливість продовження роботи зі сторінкою в момент коли вона виконує якісь дії. Під спілкуванням із сервером слід розуміти виконування простих HTTP запитів на одержання контенту, що розташований на сервері.
- Серверні технології. Вони необхідні для того, шоб була можливість обробки запитів, які надходять з боку клієнта. Одним із прикладів такої технологій є PHP.

Виконати запит до серверу можна створивши екземпляр об’єкту XMLHttpRequest із певними властивостями. Приведемо короткий опис основних методів та властивостей об’єкту XMLHttpRequest у таблиці 2.2:

Таблиця 2.2 – Опис основних методів для об’єкту XMLHttpRequest

Метод	**Опис**
open(“method”, “URL”[, async-Flag[,”userName”[,”password”]]])	ініціалізація параметрів запиту
send(content)	виконання HTTP запиту
status	повернути код статусу запиту
statusText	повернути інформацію про стан запиту у повідомленні
responceXML	повернути відповідь від серверу у форматі XML документу
responceText	повернути відповідь серверу у форматі текстового рядку
onreadystatechange	дає можливість встановити установки функції зворотнього виклику, яка буде обслуговувати стан запиту

Продовження таблиці 2.2

readyState	повертає стан запиту: 0 = не ініціалізовано 1 = триває відправлення запиту 2 = запит успішно відправлений 3 = триває обмін 4 = завершений
setRequestHeader("label","value")	призначає пари label/value у заголовку запиту
getResponseHeader("headerLabel")	повернення заголовку однієї відповіді у вигляді рядка
getAllResponseHeaders()	повернення заголовків відповідей у вигляді рядка
abort()	виконує зупинку виконання поточного запиту

Проаналізувавши розглянуті технології стає зрозуміло, що сучасний веб-застосунок необхідно, щоб був динамічним. Тому у розробці програмного комплексу візуалізації даних СТРП буде використано такі технології, як JavaScript та AJAX. Використання даних технологій обумовлено необхідністю динамічного оновлення даних отриманих від СТРП у режимі реального часу.

Висновки

У цьому розділі були розглянуті мови програмування, які необхідні для створення програмного комплексу візуалізації даних СТРП, а також необхідне ПЗ для функціонування веб-серверу.

Для реалізації серверної частини даного проєкту було обрано та проаналізовано ПЗ веб-серверу Apache.

Завдяки тому, що ПЗ обраного веб-серверу підтримує усі популярні ОС та має приблизно однакову продуктивність, робота серверної частини розроблюваного веб-застосунку не буде залежати від ОС, яка використовується на сервері. Також у порівнянні із конкурентами, Apache має модульну структуру. Тому, у будь-який час можемо вимкнути компоненти, які не використовуються у проєкті, тим самим збільшити продуктивність застосунку.

У якості серверної мови програмування буде використано PHP. Дана мова була обрана через те, що вона легко інтегрується у документи мови розмітки HTML, яка теж буде використовуватися при розробці даного проєкту.

При розробці клієнтської частини програмного комплексу буде використано CSS фреймворк Bootstrap. Його використання дозволить значно пришвидшити та полегшити розробку адаптивного інтерфейсу.

Невід’ємним атрибутом сучасного веб-застосунку являється динамічність. При розробці клієнтського інтерфейсу є необхідність у динамічному оновленні даних на сторінці, отже буде використано такі технології, як JavaScript та AJAX.

Після того, як були оглянуті та обрані основні моменти, можемо приступити до розробки програмного комплексу візуалізації даних СТРП.

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ ВІЗУАЛІЗАЦІЇ ДАНИХ САМОХІДНОЇ РАДІОКЕРОВАНОЇ СИСТЕМИ

3.1 Загальні функції програмного комплексу

Перед початком розробки програмного комплексу СТРП необхідно визначитись із його функціональними можливостями, які необхідно розробити.

Головним завданням розробленого програмного комплексу візуалізації даних СТРП полягає у прийманні телеметричних даних від СТРП та створення зручного користувацького інтерфейсу у якому зручно відображені показники прийняті від СТРП, реалізована можливість віддаленого керування СТРП та відображення трансляції відео із неї. Під телеметричними даними мається на увазі зчитані значення із датчиків, які розташовані на СТРП, а саме:

- Датчик температури
- Датчик вологості
- Барометр
- Ультразвуковий датчик відстані
- Датчик ефекту Холла
- Акселерометр
- Гіроскоп
- Вольтметр

У результаті обробки отримаємо телеметричні дані для відображення у користувацькому інтерфейсі такі, як поточна швидкість, відстань до перешкоди, температура навколишнього середовища, вологість, пройдений шлях, атмосферний тиск, кут нахилу, рівень сигналу до терміналу та рівень заряду акумуляторів. Окрім відображення поточних показників із датчиків буде реалізовано аналіз отриманих даних, та формування графіку зі статистикою за останні сім днів.

Розроблюваний програмний комплекс буде реалізовано у вигляді веб-застосунку. Основним закладеним принципом даного веб-застосунку є зручність та простота у сприйнятті телеметричних даних СТРП.

3.2 Налаштування програмного забезпечення серверу

Перед розробкою програмного комплексу візуалізації даних СТРП необхідно запустити веб-сервер. Адже без веб-серверу не може працювати логіка веб-застосунку. Сервер являє собою не лише обчислювальну машину але і набір певних програмних компонентів в залежності від поставлених завдань.

У попередньому розділі було розглянуто та обрано засоби, які необхідні для розробки веб-застосунку.

У нашому випадку для роботи серверної частини розроблюваного веб-застосунку необхідно встановити та налаштувати деякі програмні компоненти такі, як:

- Програмне забезпечення веб-серверу Apache та модуль PHP
- Система управління базами даних MySQL та графічний інтерфейс для неї phpMyAdmin

Сервер працює під управлінням останнього на даний момент Linux дистрибутиву Ubuntu 20.04. У зв’язку із цим при конфігурації програмних засобів веб-серверу необхідно враховувати сумісність із даним дистрибутивом. Керування сервером здійснюється за допомогою захищеного протоколу SSH (Secure Shell), який дозволяє віддалено керувати комп’ютером та тунелювати TCP-з’єднання.

Перед початком встановлення прикладного програмного забезпечення необхідно увімкнути базовий брандмауер, щоб забезпечити доступ для підключення лише до дозволених служб. У якості брандмауера будемо використовувати ПЗ під назвою UFW. Після встановлення прикладних програм вони можуть реєструвати свої профілі у UFW, що дозволяє керувати дозволами програм за їх іменем. Перед увімкненням брандмауеру, необхідно дозволити приймати SSH з’єднання для того, щоб була можливість віддаленого керування сервером.

На прикладі OpenSSH продемонструємо команду додавання профілю програми у список дозволених для з’єднань:

```
ufw allow OpenSSH
```

Після виконання даної команди було дозволено з’єднання по протоколу SSH. Впевнитися, що правило збережене можемо командою:

```
ufw status
```

Після виконання даної команди буде відображено список активних правил брандмауеру, приклад приведено на рисунку 3.1.

```
clouduser@dipl-ubuntu:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Рисунок 3.1 – Список активних правил брандмауеру

Першим кроком до налаштування програмних засобів серверу є встановлення ПЗ Apache.

Завдяки тому, що Apache за замовчування доступний у репозиторіях ПЗ Ubuntu можемо інсталювати його за допомогою стандартного інструменту керування пакетами. Перед інсталяцією ПЗ необхідно виконати оновлення актуальних списків пакетів, які наявні у репозиторіях:

```
sudo apt update
```

Через те, що було заздалегідь оновлено списки актуальних пакетів, які наявні у репозиторіях, можемо бути впевненими, що буде завантажена та встановлена остання версія ПЗ.

Наступним кроком є встановлення пакету apache2:

```
sudo apt install apache2
```

Перш ніж перейти до перевірки коректності інсталяції Apache, необхідно відредагувати налаштування брандмауеру для того, щоб дозволити доступ для підключення до веб-портів. Через, те, що раніше був увімкнений брандмауер, за замовчуванням він обмежує мережевий доступ до серверу. Під час інсталяції Apache він реєструє декілька профілів у UFW, які можна використовувати для керування доступом до мережі для Apache.

Для того, щоб вивести список зареєстрованих профілів програм, необхідно ввести наступну команду:

```
sudo ufw app list
```

При перегляді зареєстрованих профілів, можемо побачити, що для Apache існує три доступних профілі:

- Apache. Даний профіль дозволяє відкрити доступ до порту 80 (веб-трафіку без шифрування).
- Apache Secure. Даний профіль дозволяє відкрити доступ до порту 443 (веб-трафік із шифруванням).
- Apache Full. Даний профіль дозволяє відкрити доступ до порту 80 та 443.

Рекомендовано використовувати максимально можливо обмежений профіль. Тому у нашому випадку буде використано профіль, який дозволяє підключення тільки для 80 порту.

```
sudo ufw allow 'Apache'
```

Після налаштування правил брандмауеру можемо перейти до перевірки коректності інсталяції ПЗ Apache. Для того, щоб перевірити, що дана служба є запущеною, використаємо команду:

```
sudo systemctl status apache2
```

Результат виконання команди приведено на рисунку 3.2:

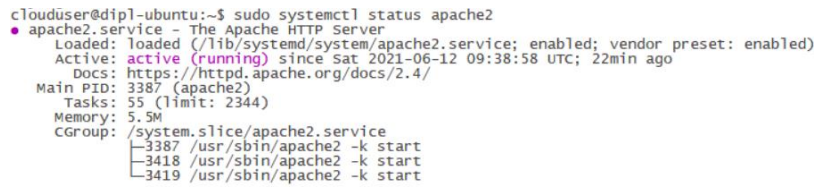

Рисунок 3.2 – Результат перевірки стану служби Apache

Результат виконання команди для перевірки стану служби Apache підтвердив, що дана служба запущена та успішно виконується. Однак для більшої впевненості у коректній роботі служби протестуємо її безпосередньо запитом сторінки із браузеру. Для запиту сторінки, яка створюється за замовчуванням перейдемо у браузері за URL-адресою http:// 31.131.23.133/, де 31.131.23.133 – IP-адреса нашого серверу.

Результат запиту веб-сторінки наведено на рисунку 3.3:

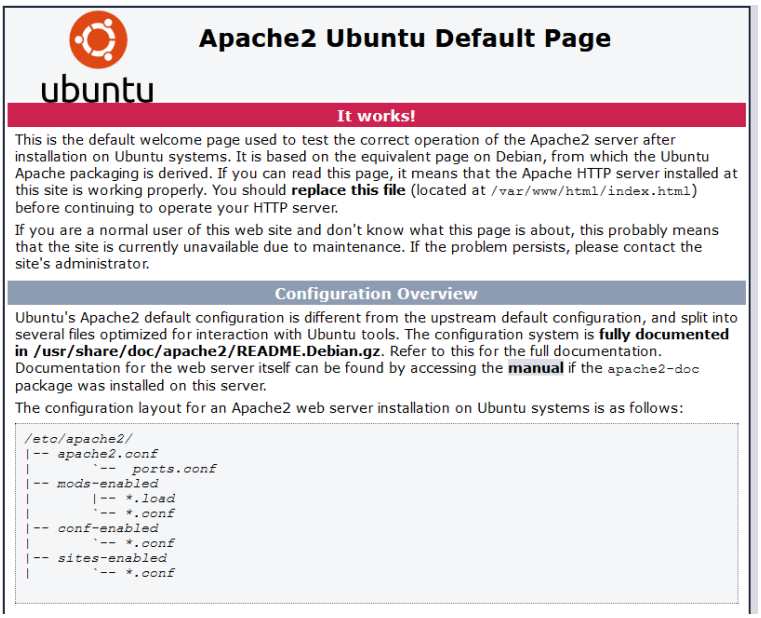

Рисунок 3.3 – Результат тестового запиту веб-сторінки

Після переходу по URL-адресі до нашого веб-серверу він повернув нам веб-сторінку, яку за замовчуванням генерує Apache після інсталяції. Це свідчить про те, що інсталяція пройшла успішно, та Apache працює коректно.

Розглянемо основні команди для керування процесом Apache у таблиці 3.1:

Таблиця 3.1 – Основні команди керування службою Apache

Команда	**Опис**
sudo systemctl stop apache2	Зупинка веб-серверу
sudo systemctl start apache2	Запуск зупиненого веб-серверу
sudo systemctl restart apache2	Перезавантаження служби Apache
sudo systemctl reload apache2	Перезавантаження служби Apache без відключення з’єднань
sudo systemctl disable apache2	Деактивація автоматичного запуску служби при завантаженні серверу
sudo systemctl enable apache2	Активація автоматичного запуску служби при завантаженні серверу

Особливістю ПЗ веб-серверу Apache є його модульність, яка дає можливість роботи із низкою серверних мов програмування. Нами було обрано у якості серверної мови програмування скриптову мову PHP. Отже для того, щоб Apache міг взаємодіяти із скриптами написаними на мові PHP, необхідно інсталювати спеціальні модулі для роботи PHP, а також одразу встановимо модулі для роботи PHP із БД:

```
sudo apt install php libapache2-mod-php php-mysql
```

У подальшому нам можуть знадобитись інші додаткові модулі, але зараз нам достатньо щойно встановлених модулів. У команді, яка приведена вище не вказано версію PHP, яку необхідно встановити, тому у такому випадку буде завжди встановлена остання доступна версія.

Після встановлення модулю PHP необхідно перевірити його роботу. Для виконання перевірки створимо у кореневій директорії /var/www/html PHP-скрипт із довільною назвою, наприклад info.php:

```
sudo nano /var/www/html/info.php
```

У тілі створеного скрипту буде викликано функцію phpinfo(), яка виведе інформацію про PHP та список параметрів:

```
<?php
phpinfo();
?>
```

Після створення та збереження тестового файлу відкриємо через браузер цей файл, який знаходиться по URL адресі http:// 31.131.23.133/info.php. У відповідь від серверу отримуємо веб-сторінку, яка приведена на рисунку 3.4. У тілі згенерованої тестовим скриптом сторінки міститься інформація про інстальований модуль PHP. Це свідчить про те, що модуль PHP був успішно інстальований та коректно працює.

PHP Version 8.0.0

System	Linux focal64 5.4.0-42-generic #46-Ubuntu SMP Fri Jul 10 00:24:02 UTC 2020 x86_64
Build Date	Nov 27 2020 12:26:22
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.0/fpm
Loaded Configuration File	/etc/php/8.0/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.0/fpm/conf.d
Additional .ini files parsed	/etc/php/8.0/fpm/conf.d/10-opcache.ini, /etc/php/8.0/fpm/conf.d/10-pdo.ini, /etc/php/8.0/fpm/conf.d/20-calendar.ini, /etc/php/8.0/fpm/conf.d/20-ctype.ini, /etc/php/8.0/fpm/conf.d/20-exif.ini, /etc/php/8.0/fpm/conf.d/20-ffi.ini, /etc/php/8.0/fpm/conf.d/20-fileinfo.ini, /etc/php/8.0/fpm/conf.d/20-ftp.ini, /etc/php/8.0/fpm/conf.d/20-gettext.ini, /etc/php/8.0/fpm/conf.d/20-iconv.ini, /etc/php/8.0/fpm/conf.d/20-phar.ini, /etc/php/8.0/fpm/conf.d/20-posix.ini, /etc/php/8.0/fpm/conf.d/20-readline.ini, /etc/php/8.0/fpm/conf.d/20-shmop.ini, /etc/php/8.0/fpm/conf.d/20-sockets.ini, /etc/php/8.0/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.0/fpm/conf.d/20-sysvsem.ini, /etc/php/8.0/fpm/conf.d/20-sysvshm.ini, /etc/php/8.0/fpm/conf.d/20-tokenizer.ini
PHP API	20200930
PHP Extension	20200930
Zend Extension	420200930
Zend Extension Build	API420200930,NTS
PHP Extension Build	API20200930,NTS
Debug Build	no

Рисунок 3.4 – Результат виконання тестового PHP скрипту

3.3 Налаштування та розробка структури бази даних

При розробці програмного комплексу візуалізації даних СТРП існує необхідність у можливості зручного структурованого зберігання даних. Вирішенням цієї проблеми є використання СУБД. Для розробки даного проєкту було обрано СУБД MySQL, адже вона являє собою оптимальне рішення для малих та середніх застосунків.

Перш ніж почати розробляти структуру БД, необхідно інсталювати систему управління базами даних. Оскільки було обрано СУДБ MySQL для її інсталяції виконаємо команду:

```
sudo apt install mysql-server
```

При використанні приведеної вище команди інсталяція MySQL без запиту змін у конфігурацію, яка встановлена за замовчуванням, а також не буде запропоновано налаштування паролю. Оскільки за такої умови MySQL залишається уразливою, то наявна необхідність виправити цей недолік.

Для щойно інстальованих СУБД MySQL має вбудований скрипт безпеки. Запустивши цей скрипт він може змінювати ряд найбільш уразливих опцій, які використовуються за замовчуванням. Прикладом таких функцій є наявність тестових користувачів та можливість віддаленого входу для користувача root.

Запустимо даний скрипт від імені адміністратора:

```
sudo mysql_secure_installation
```

Після виконання даної команди відкриється серія так званих діалогів, де буде доступна можливість внесення деяких параметрів безпеки інсталяції СУБД MySQL. У першому запиті буде запропоновано визначитися із необхідністю у налаштуванні плагіну валідації паролів, який можна використовувати для перевірки надійності паролю MySQL.

Незалежно від того чи було налаштовано плагін валідації паролів в наступному запиті необхідно буде встановити пароль для користувача root у СУБД MySQL. Важливо звернути увагу, що при вводі паролю не буде відображено ніяких символів. Після того, як буде встановлено пароль для користувача root на екран отримаємо інформацію про надійність заданого паролю. Після цього скрипт запитає підтвердження згоди на використання заданого паролю. Для згоди необхідно передати Y із клавіатури.

У подальших діалогах даного скрипту буде запропоновано видалити тестову базу даних та ряд анонімних користувачів та відключення віддаленої авторизації користувача root.

Слід звернути увагу на те, що незалежно від того, що було встановлено пароль для користувача root, при підключенні до оболонки MySQL аутентифікація за допомогою паролю для даного користувача не налаштована. Отже з’являється необхідність у налаштуванні аутентифікації.

При використанні актуальних версій MySQL у системах Ubuntu за замовчування для користувача root аутентифікація здійснюється за допомогою плагіну auth_socket, а не паролю. Дана особливість у ряді випадків забезпечує більш високий рівень безпеки та зручності, але у деяких випадках це може викликати проблеми, коли з’явиться необхідність у наданні доступу до користувача для зовнішньої програми.

Щоб замінити метод аутентифікації буде використано плагін caching_sha2_password, адже згідно офіційній документації даний плагін є рекомендованим.

Для заміни плагіну необхідно запустити командний рядок MySQL, а потім виконати команду, яка виконує заміну плагіну аутентифікації та встановлює пароль, де «password» слід замінити бажаний пароль:

```
sudo mysql
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'password';
```

Для застосування змін необхідно виконати команду:

```
Mysql> FLUSH PRIVILEGES;
```

Після виконання інсталяції та налаштування MySQL виконаємо перевірку її працездатності. Зазвичай MySQL повинна запускатися автоматично. Для впевненості перевіримо її поточний статус командою:

```
systemctl status mysql.service
```

Отриманий результат приведено на рисунку 3.5:

```
mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-04-21 12:56:48 UTC; 6min ago
 Main PID: 10382 (mysqld)
   Status: "Server is operational"
    Tasks: 39 (limit: 1137)
   Memory: 370.0M
   CGroup: /system.slice/mysql.service
           └─10382 /usr/sbin/mysqld
```

Рисунок 3.5 – Результат перевірки статусу служби MySQL

Розглянувши отриманий результат команди перевірки статусу служби можемо побачити, що служба працює та зробити висновок, що СУБД MySQL успішно інстальовано та налаштовано.

При роботі із СУБД MySQL є можливість працювати із нею лише за допомогою командного рядка. Даний спосіб взаємодії є не зручним. Вирішенням цієї проблеми є встановлення графічного веб-інтерфейсу phpMyAdmin.

Отже для спрощення роботи із СУБД виконаємо інсталяцію програмного комплексу phpMyAdmin. Виконаємо команду системи управління пакетами для завантаження та інсталяції необхідних компонентів:

```
sudo apt-get install phpmyadmin php-mbstring php-gettext
```

У процесі інсталяції буде запропоновано визначення декількох налаштувань:

- Вибір наявного ПЗ веб-серверу (у нашому випадку apache2)
- Пароль адміністратору БД
- Визначення парою для самого phpMyAdmin

У процесі інсталяції буде сформовано файл конфігурації phpMyAdmin для Apache, який буде розташовано у директорії /etc/apache2/conf-enabled/.

Для коректної роботи phpMyAdmin необхідно увімкнути такі розширення PHP, як mcrypt та mbstring. Зробимо це наступними командами:

```
sudo phpenmod mcrypt
sudo phpenmod mbstring
```

Останнім кроком у інсталяції цього програмного комплексу є перезавантаження Apache:

```
sudo systemctl restart apache2
```

Завершивши інсталяцію можемо виконати доступ до веб-інтерфейсу phpMyAdmin за URL-адресою http://31.131.23.133/phpmyadmin. Після авторизації побачимо інтерфейс користувача, який проілюстровано на рисунку 3.6:

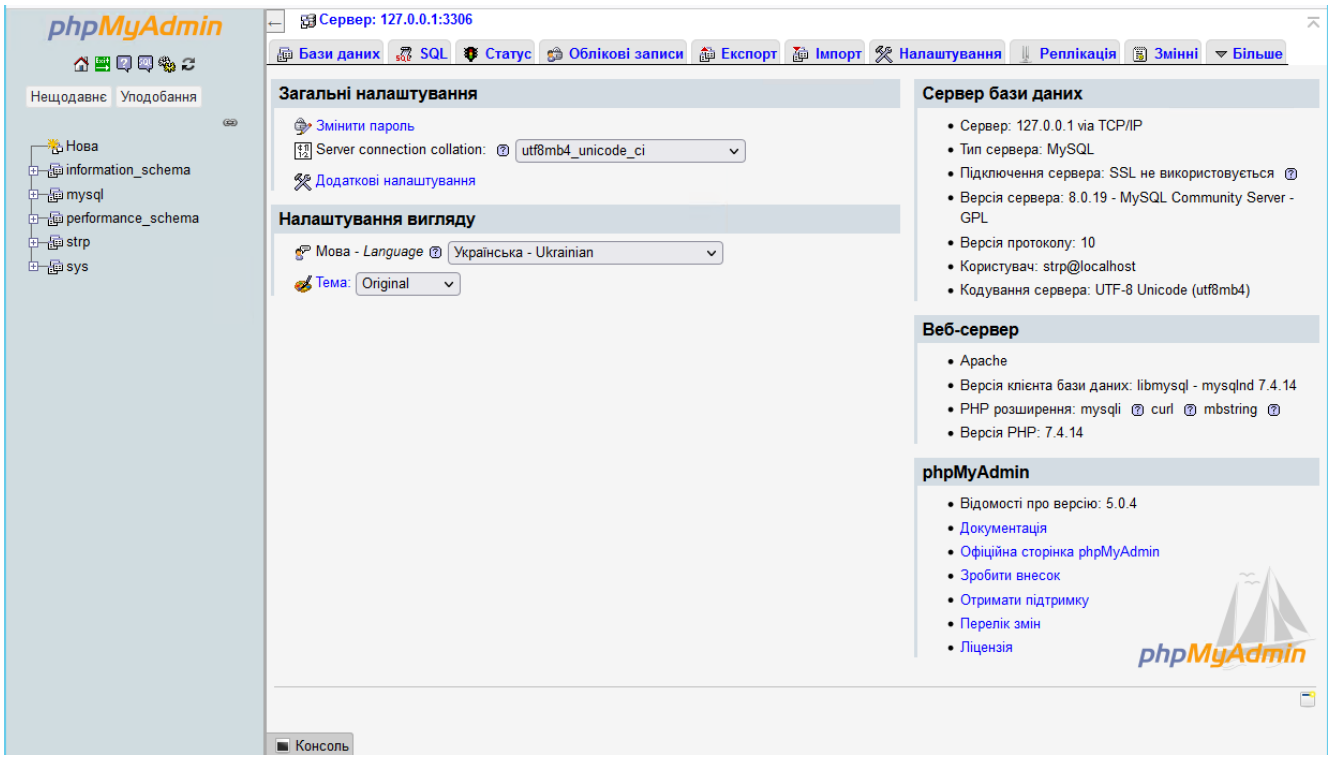

Рисунок 3.6 – Користувацький інтерфейс phpMyAdmin

Результатом встановлення вище описаного програмного забезпечення є можливість легкого створення та управління базами даних, таблицями, користувачами та багато чого іншого. Також наявна можливість зручного додавання, видалення та модифікації даних, які зберігаються у БД.

За допомогою інтерфейсу phpMyAdmin розробимо структуру бази даних програмного комплексу візуалізації СТРП.

Першим кроком є створення нової бази даних для поточного проєкту. Створимо базу даних під назвою «strp». Після створення бази даних система запропонує створити таблицю, для якої можемо задати ім’я та число колонок. Для зберігання та подальшої обробки даних із СТРП необхідно створити таблицю у якій будуть зберігатися усі отримані дані. Дані отримані із СТРП будуть у форматі json, та міститимуть у собі інформацію про поточну швидкість, вологість, відстань до перешкоди, температуру, пройдений шлях, дані акселерометру та гіроскопу, кут нахилу, висоту над рівнем моря, напругу на акумуляторах, відсоток заряду акумуляторів, рівень сигналу та текстове повідомлення.

Підрахувавши кількість даних, які будуть отримані створимо таблицю під назвою «sensors», яка буде складатись із 16 колонок. Далі необхідно визначитись із ім’ям колонок та їх типом. Приведемо у таблиці 3.2 відповідність колонок та типів даних, які вона може містити:

Таблиця 3.2 – Відповідність колонки та типу даних

Ім’я	id	time	temperature	humidity	distance	speed	mileage	accelerometer
Тип даних	INT	TIMESTAMP	FLOAT	INT	INT	FLOAT	FLOAT	JSON
Ім’я	gyroscope	angle	pressure	height	bvoltage	pbattery	Signal_level	message
Тип даних	JSON	JSON	FLOAT	FLOAT	FLOAT	INT	FLOAT	VARCHAR

3.4 Розробка серверної частини веб-застосунку

У попередньому розділі було проаналізовано та обрану PHP, як серверну мову програмування. Отже серверну частину веб-застосунку буде розроблено із використанням скриптової мови PHP. Перед початком розробки виконуємо налаштування необхідних компонентів веб-серверу та бронюємо доменне ім’я для більш зручного доступу до клієнтського інтерфейсу.

Серверна частина розроблюваного програмного комплексу буде написана без використання будь-яких систем керування контентом або фреймворків адже їх використання є надлишковим для даного проєкту.

Першим кроком є створення файлу головної сторінки index.php, створивши даний файл, залишимо його поки порожнім. Для обмеження доступу до даних, які будуть відображені у клієнтському інтерфейсі, розробимо авторизацію користувачів.

Авторизація користувачів буде здійснюватися шляхом порівняння переданих даних із форми на сторінці із тими, що містяться у БД. Тобто сторінка з формою авторизації буде містити поля для введення логіну та паролю користувача та кнопку відправки форми. Файл із сторінкою авторизації має назву signin.php. Форма буде передаватися на сервер методом POST, адже він забезпечує приховану для перегляду від звичайного користувача передачу даних. У базі даних паролі користувачів зберігаються у зашифрованій формі алгоритмом md5.

Для доступу до переданих скрипту даних використовуємо глобальну змінну $_POST. Дана змінна містить масив із даними, які передані браузером до PHP скрипту. Наприклад передані дані у полі форми із ім’ям login будуть доступні для обробки при зверненні до $_POST[‘login’].

Після того, як скрипт отримує дані із форми авторизації, він шифрує пароль алгоритмом md5, потім порівнює поле логіну та зашифрованого паролю із тими, що наявні у базі даних. Якщо передані дані ідентичні тим, що зберігаються у базі даних, то записуємо у глобальну змінну $_SESSION ідентифікатор користувача, що дозволяє не авторизуватись кожен раз при переході на іншу сторінку. Сесії є зручними у зберіганні інформації окремих користувачів із унікальним ідентифікатором сесії. Вони можуть використовуватись не лише для реалізації авторизації користувачів, а і для зберігання стану між запитами сторінок. Також до сторінки авторизації додаємо перевірку авторизації користувача, тобто, якщо у $_SESSION записано ідентифікатор користувача – перенаправляємо його на сторінку index.php.

Із коду написаного мовою PHP файл index.php містить лише перевірку авторизації. Якщо у змінній $_SESSION відсутній ідентифікатор користувача, то перенаправляємо користувача на сторінку авторизації signin.php.

Далі створимо файл sensors.php, де буде виконуватись обробка даних, які СТРП відправляє до серверу за допомогою методу GET. СТРП передає до серверу дані у форматі JSON, отже наявна необхідність у перетворенні цих даних для

подальшої обробки. У засобах PHP наявна функція json_decode(). Дана функція приймає закодований у форматі JSON рядок та перетворює його у змінну PHP.

Дані, які передає СТРП до серверу у форматі JSON:

```
{"tmp":33.5,"hum":75,"u_dist":150,"mil":115,"acc":[2,15,1],"gyr":[1,67,2],
"ang":[0,2,5],"prs":115,"alt":90,"bat":3.8,"bpc":70,"sig":-
32.4,"txt":"message"}
```

Таким чином після того, як сервер прийняв дані від СТРП їх було декодовано у формат змінної PHP. Після декодування формуємо запит до БД для запису отриманих даних.

MySQL запит виглядає так:

```
INSERT INTO `sensors` (`id`, `time`, `temperature`, `humidity`, `distance`,
`speed`, `mileage`, `accelerometr`, `gyroscope`, `angle`, `pressure`,
`height`, `bvoltage`, `pbattery`, `signal_level`, `message`) VALUES (NULL,
CURRENT_TIMESTAMP, '{$data['temperature']}', '{$data['humidity']}',
'{$data['distance']}', '{$data['speed']}', '{$data['mileage']}',
'$accelerometer', '$gyroscope', '$angle', '{$data['pressure']}',
'{$data['height']}', '{$data['bvoltage']}', '{$data['pbattery']}',
'{$data['signal_level']}', '{$data['message']}')")
```

У змінній $data зберігаються отримані декодовані дані, що дозволяє динамічно їх підставляти у запит MySQL. На рисунку 3.7 приведено схематичне зображення принципу обробки даних прийнятих із СТРП.

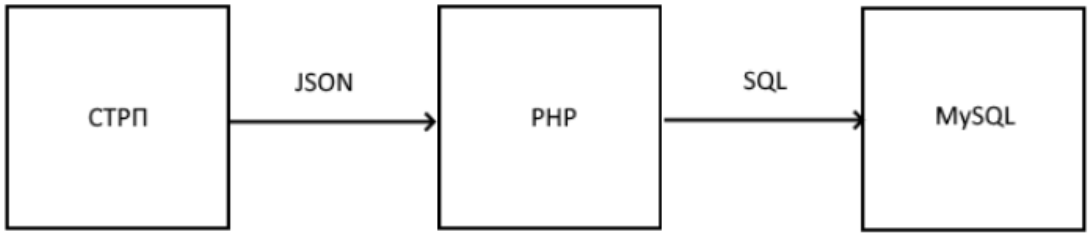

Рисунок 3.7 – Схематичне зображення обробки отриманих даних

Після виконання MySQL запиту виводимо повідомлення «Ок» при успішному виконанні запиту та «Error» при невдалому виконанні запиту. Дані повідомлення необхідні для можливості оцінки працездатності скрипту.

Також на файл sensors.php полягає функція у виборі даних із бази даних та їх відображені у форматі JSON. Відображення даних із БД у форматі JSON необхідне для подальшої обробки їх на стороні користувача.

Для вибірки збережених даних виконуємо MySQL запит:

```
SELECT * FROM `sensors` ORDER BY `id` DESC limit 1
```

Приведений вище запит зчитує із бази даних останній запис, який було зроблено у ній. Після отримання даних із БД перед їх виводом необхідно закодувати у формат JSON. Для кодування даних у формат JSON, PHP має вбудовану функцію json_encode(). Приведемо приклад закодованих даних у формат JSON:

```
{"id":"6","time":"2021-06-08
01:09:06","temperature":"33","humidity":"70","distance":"135","speed":"2",
"mileage":"12","accelerometr":"0","gyroscope":"0","angle":"0","pressure":
"110","height":"97.81","bvoltage":"3.83","pbattery":
"-36.4","signal_level":"-25","message":"TEXT"}
```

Для наочності приведемо схематичне зображення функції отримання даних із БД на рисунку 3.8:

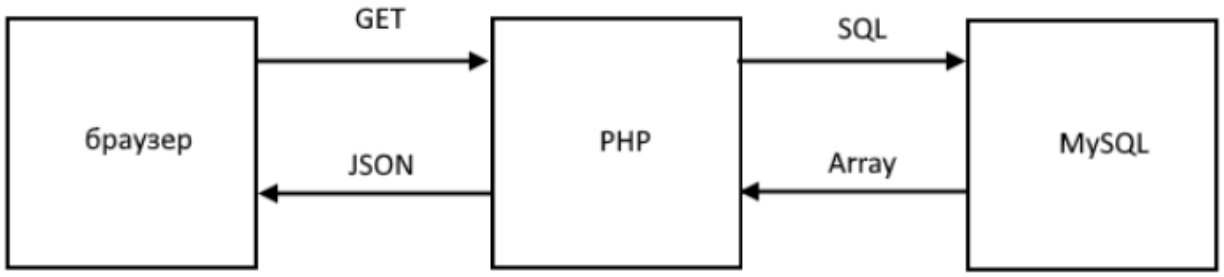

Рисунок 3.8 – Схематичне зображення функції отримання даних

3.5 Розробка користувацького інтерфейсу

Розробка користувацького інтерфейсу буде розроблено за допомогою мови розмітки гіпертексту HTML, таблиць стилів CSS (із використанням Bootstrap), та мови сценарії JavaScript (із використанням бібліотеки JQuery).

Першим кроком у файлі index.php напишемо базову структуру HTML та підключимо до сторінки набір стилів Bootstrap (bootstrap.min.js) та JQuery (jquery.min.js).

У додатку 1 наведено приклад базової структури HTML із підключенням Bootstrap та JQuery.

У результаті, в поєднанні із PHP отримаємо у відповідь сторінки з написом Hello Word, але для цього необхідно авторизуватись. Отже створимо файл signin.php та створимо макет сторінки використовуючи Bootstrap.

Сторінка авторизації повинна містити шапку сайту, а також форму для вводу логіну та паролю.

Для створення шапки використовуємо базовий клас Bootstrap .navbar, та стилізуємо його додаючи додаткові класи.

Далі за допомогою використання тегу створимо головний блок, який буде містити вміст сторінки. Присвоїмо йому клас .container. Даний клас задає максимальну ширину блоку 1320 пікселів.

За допомогою тегу <form> створимо форму для відправки на сервер введених логіну та паролю. Форма буде містити два поля input із ім’ям login та password та типами text та password відповідно, а також кнопку відправки форми. Зовнішній вигляд розробленої сторінки приведено на рисунку 3.9:

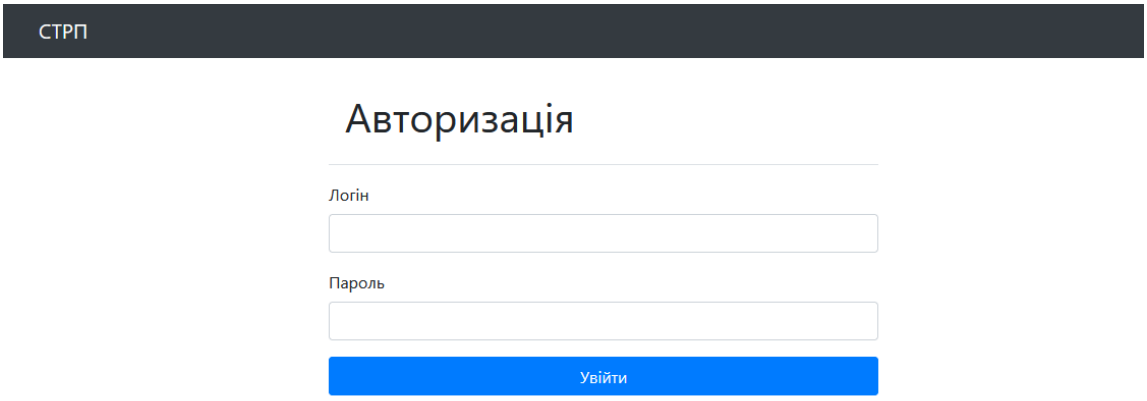

Рисунок 3.9 – Зовнішній вигляд розробленої сторінки авторизації

Після розробки сторінки авторизації, повернемось до розробки головної сторінки. На головній сторінці повинно міститись шапка із меню, показники поточного заряду акумуляторів та якість сигналу, набір показників, які отримуємо із СТРП і графік на якому зображена статистика за останні дні.

Для розробки інтерфейсу відображення поточних значень датчиків СТРП буде використано клас .card. Клас .card являє собою гнучкий контейнер вмісту. Для відображення необхідних елементів потрібно створити вісім блоків із класом .card.

Приведемо на рисунку 3.10 скріншот інтерфейсу даних із СТРП

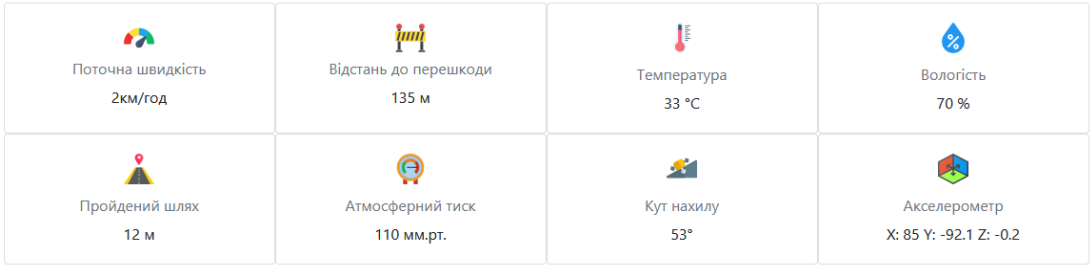

Рисунок 3.10 – Скріншот інтерфейсу даних із СТРП

Після розробки інтерфейсу візуалізації даних необхідно відобразити графік із даними середніх значень отриманих із СТРП за останні декілька днів. Для цього використаємо бібліотеку Charts.js. Особливістю графіку є те, що дані формуються автоматично в залежності від поточної дати. Приведемо на рисунку 3.11 розроблений графік зі статистикою:

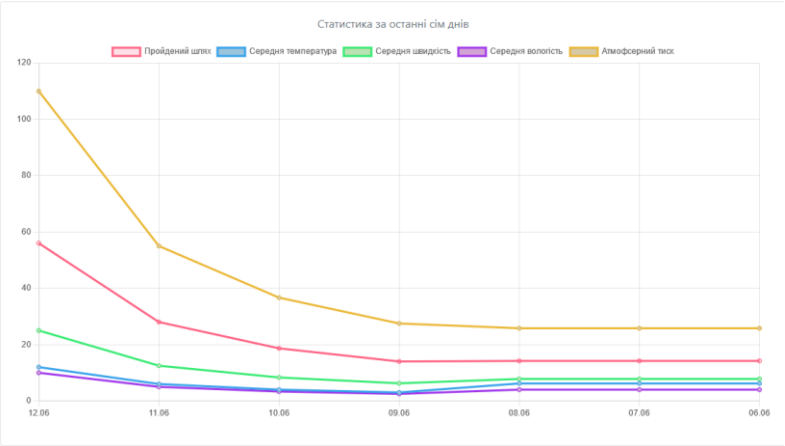

Рисунок 3.11 – Візуалізація статистики із датчиків

На рисунку 3.12 наведемо скріншот зовнішнього вигляду розробленої сторінки у цілому:

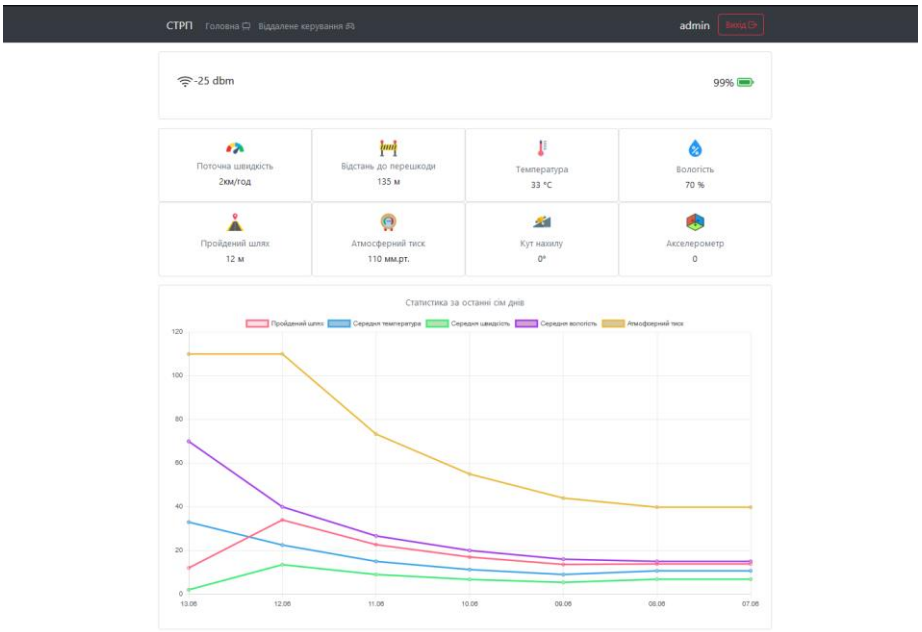

Рисунок 3.12 – Зовнішній вигляд сторінки index.php

Після завершення розробки головної сторінки, створимо сторінку під назвою control.php. Ціллю створення цієї сторінки полягає у тому, щоб надати користувачеві можливість переглядати відео трансляцію із камери, яка встановлена на СТРП, а також надавати можливість посилати команди на керування переміщенням СТРП.

Основний вміст сторінки помістимо в блок із класом .container. Потім розмістимо на сторінці дані про рівень заряду акумуляторів та рівень сигналу СТРП. Це необхідно для, того щоб можна було завжди бачити головні показники.

Відео трансляція із камери СТРП буде у форматі MJPEG, поміщена в тег <img> з атрибутом src для того, щоб задати шлях до відображуваного контенту.

Останнім кроком є створення кнопок для можливості передачі команд управління. Створимо чотири кнопки за допомогою тега <button>.

На рисунку 3.13 приведено зовнішній вигляд сторінки control.php.

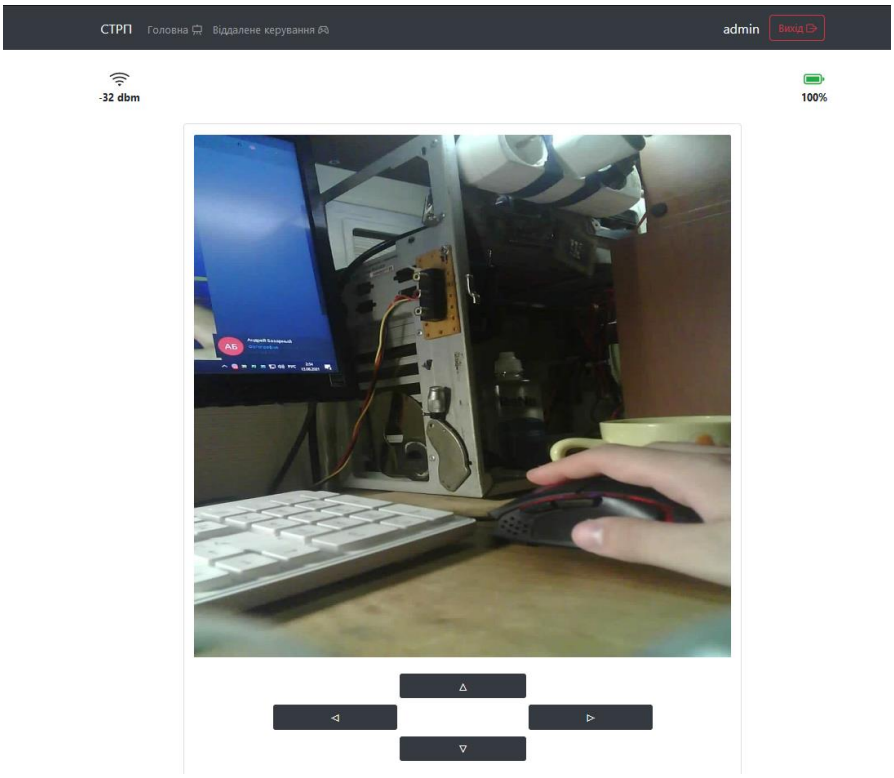

Рисунок 3.13 – Зовнішній вигляд сторінки control.php

Після завершення створення сторінок, необхідно зробити їх більш динамічними. Головним завданням створення веб-інтерфейсу є відображення поточних показників СТРП, а отже їх необхідно оновлювати без повного перезавантаження сторінки. Вирішимо цю проблему за допомогою JavaScript та технології AJAX. Використаємо JavaScript бібліотеку Jquery для полегшення та пришвидшення написання коду, який дозволяє у фоновому режимі обмінюватися даними із сервером.

Створимо файл під назвою refreshSensors.js та підключимо його до сторінки index.php за допомогою тегу <script>. У даному файлі буде розміщуватися програмний код для виконання запиту до серверу та обробки відповіді від нього. Сам програмний код буде приведено у додатках. Логіка цього файлу полягає у тому, щоб виконати запит до віддаленого серверу та отримати у відповідь дані у форматі JSON. Після отримання даних необхідно їх відобразити на сторінці. Для можливості оновлення сторінки заздалегідь при розмітці були створені елементи із певними атрибутами id. За допомогою цього атрибуту можемо легко знаходити та керувати цим елементом за допомогою JavaScript.

Висновки

У даному розділі було виконано розробку програмного комплексу СТРП. Було налаштовано програмне забезпечення необхідне для роботи веб-серверу, розроблена структура бази даних, розроблену серверну частину застосунку та клієнтську. При розробці даного веб-застосунку було реалізовано такі функціональні можливості:

- обробка даних отриманих із СТРП
- відображення отриманих даних із СТРП у клієнтському інтерфейсі з динамічним їх оновленням
- відображення відео трансляції із камери встановленої на СТРП
- можливість відправляти команди для керуванням СТРП.

У розробці програмного комплексу візуалізації даних СТРП було використано такі засоби, як:

- Програмне забезпечення веб-серверу Apache
- Програмне забезпечення СУБД MySQL
- Мова програмування PHP
- Мова гіпертекстової розмітки HTML
- Каскадні таблиці стилів CSS
- Мова програмування JavaScript

У результаті розробки програмного комплексу візуалізації даних СТРП було отримано працюючий веб-застосунок, який розміщено на віддаленому сервері, та який доступний за URL: http://web-strp.pp.ua.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аббасова Т.С., Мороз А.П., Белюченко И.М., Стреналюк Ю.В. Разработка требований к программно-техническим средствам информационно-телеметрических систем. Информационно-технологический вестник. 2017;11(1):55-67
2. Веб-приложение [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Веб-приложение.
3. Вебсторінка [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Вебсторінка.
4. Студенческий:научный журнал. –No 16(36). Часть 1. Новосибирск: Изд.АНС «СибАК», 2018. –29 с
5. FidoNet: Technology, Use, Tools, and History [Електронний ресурс] – Режим доступу до ресурсу: 1. https://www.fidonet.org/inet92_Randy_Bush.txt
6. Бородіна О.О. Еволюція Web технологій в сучасних умовах / О.О. Бородіна, А.М. Гафіяк, С.Д. Просвєтов, О.Р. Білобров // Математичне та імітаційне моделювання систем. МОДС 2019 : тези доповідей Чотирнадцятої міжнар. наук.-практ. конф. (Чернігів, 24 - 26 червня 2019 р.). - Чернігів : ЧНТУ, 2019. – С. 256-258.
7. Web 3.0 [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Web_3.0.
8. Застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Застосунок.
9. Бекирова Э.А., Халилова З.Э. Основные этапы создания web-приложений // Информационно-компьютерные технологии в экономике, образовании и социальной сфере: — "Крымский инженерно-педагогический университет имени Февзи Якубова" (Симферополь), 2019. — С. 84—91.
10. Gmail [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Gmail.
11. Office Online [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Office_Online.

12. Багатоплатформність [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Багатоплатформність.
13. Что такое веб-сервер [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Common_questions/What_is_a_web_server.
14. Laurie B. Apache: The Definitive Guide: The Definitive Guide, 3rd Edition / B. Laurie, P. Laurie., 2002. – 590 c. – (O'Reilly Media, Inc.).
15. URL [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/URL.
16. nginx [Електронний ресурс] – Режим доступу до ресурсу: https://nginx.org/ru.
17. Clément N. Nginx HTTP Server - Second Edition / Nedelcu Clément., 2013. – 301 c.
18. Разработка Web-приложений на PHP и MySQL: Пер. с англ./Лаура Томсон, Люк Вел-линг. — 2-е изд., испр. — СПб: ООО «ДиаСофтЮП», 2003. — 672 с
19. HTML | MDM [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML.
20. HTML и CSS: путь к совершенству. — СПб.: Питер, 2011. — 336с.: ил. — (Серия «Бестселлеры O’Reilly»).
21. Spurlock J. Bootstrap / Jake Spurlock., 2013. – (O'Reilly Media, Inc.).
22. JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ-Плюс,2008. – 992 с., ил.

Додаток А

Переклад першого розділу

1. ANALYSIS OF DEVELOPMENT AND THE PRINCIPLE OF WEB APPLICATION

1.1 Analysis of web technology development

Few people can imagine life in today's world without such familiar things as computers and the Internet. Few Internet users have ever thought about how websites and other Internet resources are designed and created.

Understand the principles of creating websites to help web pages - information resources that can be accessed from the Internet. Most web pages are written in HTML markup language file format. Web pages are usually hosted on hypertext sites using navigational hyperlinks on other pages. They can be obtained from a remote web server, or web pages can be stored on a local computer [3].

Consider the stage of development of web technologies in the world.

Web 0.0 - in the period of creation the basic mechanisms offering transition to web 1.0 were put and developed. At the same time, the first attempts were made to unite information networks into one global one. During this period, linear feedback systems between terminal users and mainframes were widespread. Quasi-mail networks such as usenet, biznet, fidonet were also widespread [4-5].

Web 1.0 - creating content on a website is possible only for its developers and administrators. This is the first implementation of the global Internet, often referred to as a read-only network. There were virtually no ways to interact with the content. Also in the era of Internet 1.0 began the emergence of the concept of e-commerce. There are a number of sites that offer paid services to users. The aim was to present the product to potential consumers, as the catalog does, but the advantage of websites in this area allows you to provide a service or product for a person in any part of the world [4].

Web 2.0 - In 2004, O'Reilly coined the term Web 2.0. The second generation of web services is not a significant evolution of their definition, but rather a new use of

resources on the Internet. The main difference from previous generations is that website users are not ordinary readers and can also participate in the creation of a website that contains content created by website users but is verified by the administration. At this time, the most popular social networks were created, such as Facebook, VKontakte, Twitter [4].

Web 3.0 is a high-quality services and content created on the Web 2.0 technology platform by professionals [6]. Thanks to the ability to use a large number of powerful Internet services on the 2.0 website quickly and without significant financial costs, a large number of such resources have appeared, which has resulted in the devaluation of most of them. A striking example of the transition from Web 2.0 to Web 3.0 is the German section of Wikipedia. In the future, using content filled with quality articles, administrators will be banned from editing quality articles by inexperienced participants [7].

1.2 The concept and description of the principles of web applications

Most people who use a personal computer know that Windows is such a program. It is a computer program that allows participants to solve specific complex problems [8]. For example, email clients, text editors, media players, and more. Here are some definitions to understand what a web application is.

Web application - a distributed application in which its logic, located on servers, is a function of generating information downloaded from the server, the browser [2]. Web applications are applications that run on third-party Web servers and create an interface between websites and users. Written web programs most often create written scripting languages (PHP, Perl), less often they are developed using a high-level language (C, C ++, etc.), which must be compiled for a particular OS.

A web server is a server that receives an HTTP request from clients and is responsible for an HTML page or other data. Clients access the web server using the URL of the desired page [9].

With the development of the World Wide Web, web applications are also evolving and improving rapidly. Important features of web applications are available in practice for any user device and anywhere in the world where network access is available. High

quality, security and scale are also important features of web applications. In addition to the above characteristics, a web application today must have such a characteristic as a multiplatform. This allows the web system to be compatible with any operating system and browser, which solves the problem of software compatibility.

A web application is a browser-based application that requires Internet access and a browser installed on the client device. In this case, by calculating, packaging, and storing information about the information on a remote web server, the browser on the user device is a client program.

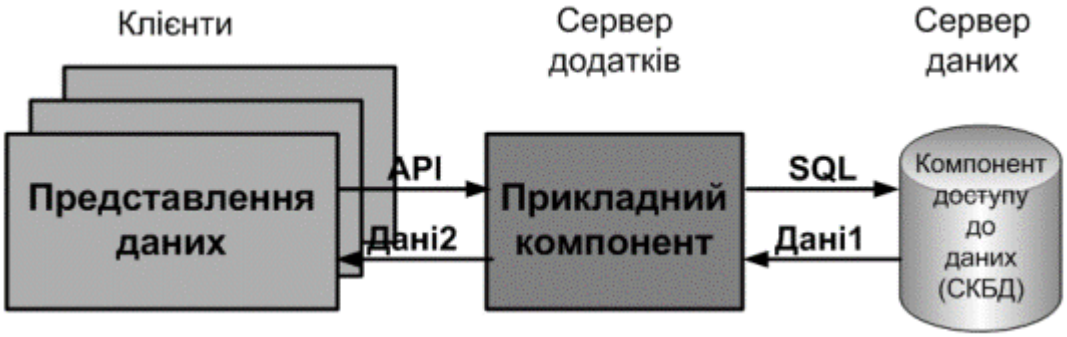

Figure A.1 - Scheme of web programs

Take, for example, the world-famous Gmail email client. This e-mail client does not appear for the e-mail software that you want to install on your device. The web interface of the Gmail email client is usually written in a browser-based JavaScript script. By selecting this programming language, the client interface can receive keyboard commands, refresh pages without reloading using AJAX technology and many other features [10].

Looking at the structure of web applications, you can see that all the logic of the program is hosted on servers, and the user interface can be accessed by any user using programs that have existed since the beginning of the HTTP network (web browser). At the dawn of web technology, a full-fledged web program lacked such important technologies as JavaScript and DOM models.

Nowadays, most web resources use JavaScript technology to perform some third-party functions without having to access a remote server. But with the advent of JavaScript programming language using these technologies is very different from today. In those days, JavaScript technology was not used because it was needed, because it

simply existed. At one time, the use of JavaScript technology on web pages became a bad tone, because this technology is used to make the elements on the page respond to the movement of a computer mouse, change colors, dynamically control the styles of elements on the page.

The only reason why web applications are becoming more popular only nowadays is because the always weak

point in creating web applications is the dynamism and interactivity in interacting with web pages in web applications. The JavaScript that exists today helps to completely solve this problem.

Thus, we found above that in normal software, the program's logic is hosted on users' computers, and in web applications on a remote server. If the program logic is on the servers, there is only one current copy of the programs, so this feature can be easily distributed to all users. With the help of web technology, you can request the usual way of distributing programs, if users in this case do not receive a full copy of the programs. Users of the program receive on their devices only what they need to work, through the program interface. With this, you can say that there are no problems with the distribution of web applications, you can access them from any device and anywhere where you connect to the Internet.

To work with a web application, a user only needs one program, which is called a web browser. To distribute the work from a web application, the user only needs to enter the URL in the address bar of the browser.

Unlike a user installing an application on their devices, a web application is hosted on a remote server. This frees the user from the administrator role. He doesn't need to install the program, troubleshoot them, or help the administrator run people or a group of people involved in web application developers. Obviously, in this case, the application developer will be more loaded. However, from an economic point of view, the model of a small group of programmers focused on working on the application in one place is much more profitable. After all, such a model is more efficient and cost-effective, so you do not need to support a team of specialists who install and debug the program on users' devices.

The web application does not require user requirements. It is further suggested that the user has a browser on their device, as this is one of the main applications on modern electronic devices. The basic idea is that a web program can run on any special operating system. We can take advantage of the fact that the web program has no special hardware requirements. With the advent of web applications, the problem of supporting different versions is a thing of the past. Once the developer publishes a new version of the program, all customers without exception receive the latest version, you only need to reload the page in a web browser. Once a developer has released an updated version, all outdated versions of web applications disappear immediately. As a result, customers may not notice that they are already running an updated version of the software. You can also resolve backward compatibility issues and support older and older versions of the program.

If your memory status is very limited, one option is to use a web application. After all, no need to download the full added application to your organization, so if the devices have limited access to available memory. In some cases, you may not fully load the interface, just download the part of the interface that is needed to perform a particular task. With this feature, web applications take up very little space on similar user devices, as well as download and respond to user actions faster.

Because the web browser works with the user through a web browser, there is no problem with some protocols and data being closed by the firewall. If the client allows you to download web pages, users of web applications will not be able to resolve the download issue.

With a web application architecture that is invisible to transient users, you can create a large-scale application enhancement program without interfering with user devices. For example, it is not possible for a user, you can change or modify the server hardware, when a web application is hosted, distribute the download to the program between several other servers.

It is clear that web programs do not have the same capabilities as regular national programs. For example, using a web application whose member cannot control the hardware components of the device from a browser works with composite 3D graphics

and models. The above is relevant today and in the near future, as nowadays web technologies are developing rapidly.

Looking at how web technologies combine and evolve, we can say that web applications are the technology of the future. They began to grow rapidly and connect to national software when users could use it almost daily. An example of such software is Microsoft Office Online. This is a set of official program that is a web application. This allows users to create and edit files [11]. Accordingly, this analogue of the national Microsoft Office software is a light version. Office Online does not require installation on users' devices, so you can work with documents from any device and anywhere.

1.3 Advantages and disadvantages of web applications

At the design stage of a new application, the developer first faces a task, which is a definition with the type of application. At this stage, the developer has several options. One of the options at this stage is to develop an application that is specifically designed for this software and hardware platform using the compilers and development environments recommended for this platform. Another option is to develop a web application that has a feature such as independence from the software and hardware platform of the user device. It is from the choice of developer at this stage will depend on all subsequent stages in the development of the application. This stage in the development is actually very important. It is at this stage that they are determined in order to attract employees who are focused on development for the selected platform, and it is at this stage that the approximate cost of the project can be understood.

When it comes to native and web applications, it's hard not to mention hybrid solutions. Take Adobe PhoneGap as an example. This software package allows you to write an application using a scripting programming language. An application written using this environment runs in the browser that comes with the written application and is a holistic native application. The fee for the independence of the application from the software and hardware platform is higher requirements for memory and computing power, compared to the native application. If you use hybrid development tools, then in any case only part of the developed application will be universal. In addition, it can be difficult to create the necessary element for user interest, the development of which

requires a significant amount of time. In such cases, it may be easier to develop a native application.

Web technologies today are a clear proof that technology that was once successfully created can be used in a variety of situations by different people.

The difference between a native application and a web application is that the native application interacts with the operating system installed on the device, and web applications interact only with the browser installed on the user's device. This feature significantly expands the audience that can use the application, as the user gets the opportunity to use the application on any user device. HTML hypertext markup language allows you to make adaptive markup, ie adjust the design and elements of interaction to the size of the screen on the user's device. But despite the great advantage of web applications in that they are hardware and software independent, the main disadvantage of web applications is their much lower performance than native and platform-specific applications.

The advantage of native applications is that they do not have an order of magnitude higher performance. In addition to greater productivity when developing a native application, the developer can use all the specific features of the operating system and the platform as a whole. A significant disadvantage of native applications is that they can only run on one hardware and software platform, unless specifically adapted to different platforms. That is, if you are required to migrate your application to other devices with other operating systems, the developer will need to completely rewrite and adapt the application code to another software and hardware platform. This requires a larger team of developers, which entails an increase in development costs. Precisely because native applications have more capabilities to interact with the software and hardware components of the user's device, many truly powerful applications are available on only one hardware and software platform.

Native applications differ from web applications in that they are stored on users' devices. Although most people these days are already fairly familiar with downloading native applications, the developer can not be sure that each user will use the same version of the application.

Native applications have many advantages and disadvantages compared to web applications. Because native applications work directly with the operating system and hardware of the user's device, it is much easier for the developer to use hardware components such as a microphone, camera, device location services, and more. Another advantage of native applications is that most of these applications are placed on various platforms, so-called application stores. This is an advantage because before the application is published in the store, it is tested and therefore the application is much less likely to be harmful and incompatible with the user's device. But along with the advantages of native applications, they also have some disadvantages. As a rule, the development of a native application is more expensive. This is more relevant for an application developer who wants it to be compatible with multiple hardware platforms. Also, in addition to development, the cost of maintaining and updating applications is much higher, especially the difference is noticeable if the application is available for at least several platforms. Another disadvantage of such applications is that users use different versions of the product, which causes problems with development, as well as compatibility with different versions of the application.

If we compare the web application with the native application, we can say that the former is much easier to use due to the fact that at any time and from any device the user can access the current version of the application.

Of course, both native applications and web applications also have their advantages and disadvantages. One of the advantages of web applications is that they are much easier to maintain because they have a common code base for all devices, and they can be easily adapted to be compatible with any old and outdated devices. Web applications do not need to be fully downloaded to a user device, so this advantage is most relevant for devices with limited storage space. Because such an application runs in a web browser for the developer, it does not matter on which hardware platform the user's device. Therefore, you can use the application on all devices where the browser is installed. Because the application itself is stored on a remote server, the client only loads the user interface. This makes it possible to use a single version of the application, so the problem with the compatibility of different versions of the product disappears. Although many

people today are familiar with electronic devices and use them almost every day, the share of experienced users is not large. Therefore, a significant advantage of web applications is that you do not need to install and configure the application on your device, because sometimes setting up some applications is not an easy task.

Along with the advantages of web applications, in some cases the critical disadvantage is that such applications have very limited access to the hardware components of the user device. In addition, to work with a web application requires access to the Internet, without it the user will not be able to work with the application because it is located on a remote server, and the user will receive only the user interface. It can also be a significant disadvantage for some users that the data is stored and processed on a remote server, so there is a risk of leaking or losing sensitive information.

Conclusions

This section discusses the history of development, basic concepts and principles of web technologies, their advantages and disadvantages.

Examining the history of web technologies, we can conclude that they are rapidly entering our lives. At the beginning of their emergence, no one could have imagined that the web would grow into something more than just static sites, filled with content only by site administrators and they have no interactivity. Today, websites are at a stage where not only resource administrators can generate content, but also users themselves.

When considering the principle of operation, it was found that the web application consists of a server and a client part. The server part consists of the web server software that accepts HTTP requests, and the server is the logic of the web application written in the server programming language.

The main advantages of using web applications are ease of use, the ability to use them on different platforms and the absence of problems with closed ports, because usually the ports on which web applications run are not closed by a firewall.

Додаток Б

Лістинг коду веб-застосунку

Файл index.php

```
<?php
session_start();
if (empty($_SESSION['u_name'])) {
    header('Location: /signin.php');
    exit();
}
if ($_GET['action'] === 'logout') {
    session_destroy();
    header('Location: /');
    exit();
}
?>
<!doctype html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width,initial-scale=1.0">
    <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
    <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
    <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
    <link rel="manifest" href="/site.webmanifest">
    <link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">
    <meta name="msapplication-TileColor" content="#00aba9">
    <meta name="theme-color" content="#ffffff">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
```

```
    <link  rel="stylesheet"  href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css">
    <title>СТРП</title>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-3">
    <div class="container">
        <a href="#" class="navbar-brand">СТРП</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarContent"      aria-controls="navbarContent"      aria-expanded="false">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarContent">
            <ul class="navbar-nav mr-auto mb-1 mt-2">
                <li class="nav-item">
                    <a  href="/"  class="nav-link">Головна <i  class="bi  bi-easel"></i></a>
                </li>
                <li class="nav-item">
                    <a    href="/control.php"    class="nav-link">Віддалене керування <i class="bi bi-controller"></i></a>
                </li>
            </ul>
            <div class="d-flex">
                <h5            class="mr-3            mt-2            text-white"><?=$_SESSION['u_name']?></h5>
                <a href="/?action=logout" class="btn btn-outline-danger">
                    Вихід <i class="bi bi-box-arrow-right"></i>
                </a>
            </div>
        </div>
    </div>
</nav>
<main>
```

```
    <div class="container">
        <div class="row">
            <div class="card w-100">
                <div class="card-body">
                    <table class="table table-borderless my-2">
                        <th scope="col">
                            <h5>
                                <svg     xmlns="http://www.w3.org/2000/svg"
width="32"   height="30"   fill="currentColor"   class="bi   bi-wifi   mb-2"
viewBox="0 0 16 16">
                                    <path d="M15.385 6.115a.485.485 0 0 0-
.048-.736A12.443 12.443 0 0 0 8 3 12.44 12.44 0 0 0 .663 5.379a.485.485 0 0
0-.048.736.518.518 0 0 0 .668.05A11.448 11.448 0 0 1 8 4c2.507 0 4.827.802
6.717 2.164.204.148.489.13.668-.049z"/>
                                    <path  d="M13.229  8.271c.216-.216.194-
.578-.063-.745A9.456 9.456 0 0 0 8 6c-1.905 0-3.68.56-5.166 1.526a.48.48 0
0 0-.063.745.525.525 0 0 0 .652.065A8.46 8.46 0 0 1 8 7a8.46 8.46 0 0 1 4.577
1.336c.205.132.48.108.652-.065zm-2.183         2.183c.226-.226.185-.605-.1-
.75A6.472  6.472  0  0  0  8  9c-1.06  0-2.062.254-2.946.704-.285.145-.326.524-
.1.75l.015.015c.16.16.408.19.611.09A5.478 5.478 0 0 1 8 10c.868 0 1.69.201
2.42.56.203.1.45.07.611-.091l.015-.015zM9.06   12.44c.196-.196.198-.52-.04-
.66A1.99  1.99  0  0  0  8  11.5a1.99  1.99  0  0  0-1.02.28c-.238.14-.236.464-
.04.66l.706.706a.5.5 0 0 0 .708 0l.707-.707z"/>
                                </svg>
                                <span id="signal_level">
                                    <span   class="spinner-border   spinner-
border-sm" role="status" aria-hidden="true">
                                    </span>
                                </span>
                                <span id="sub_signal_level" style="display:
none">
                                    dbm
                                </span>
                            </h5>
                        </th>
                        <th scope="col">
                            <h5 class="text-right">
                                <span id="pbattery">
```

```
                                    <span   class="spinner-border   spinner-
border-sm" role="status" aria-hidden="true">
                                    </span>
                                </span>
                                <span   id="sub_pbattery"   style="display:
none">
                                    %
                                </span>
                                <svg     xmlns="http://www.w3.org/2000/svg"
width="32" height="32" fill="currentColor" class="bi bi-battery-full text-
success mb-1" viewBox="0 0 16 16">
                                    <path d="M2 6h10v4H2V6z"/>
                                    <path d="M2 4a2 2 0 0 0-2 2v4a2 2 0 0 0
2 2h10a2 2 0 0 0 2-2V6a2 2 0 0 0-2-2H2zm10 1a1 1 0 0 1 1 1v4a1 1 0 0 1-1
1H2a1 1 0 0 1-1-1V6a1 1 0 0 1 1-1h10zm4 3a1.5 1.5 0 0 1-1.5 1.5v-3A1.5 1.5
0 0 1 16 8z"/>
                                </svg>
                            </h5>
                        </th>
                    </table>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="card col-md-3 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img      src="./img/speedometer.png"      alt="speed"
class="card-title">
                    <h6   class="card-subtitle   mb-2   text-muted">Поточна
швидкість</h6>
                    <p class="card-text">
                        <span id="speed">
                            <span  class="spinner-border  spinner-border-sm"
role="status" aria-hidden="true">
                            </span>
                        </span>
```

```
                        <span id="sub_speed" style="display: none">
                            см/с
                        </span></p>
                </div>
            </div>
            <div class="card col-md-3 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img src="./img/barrier.png" alt="caution" class="card-
title">
                    <h6 class="card-subtitle mb-2 text-muted">Відстань до
перешкоди</h6>
                    <p class="card-text"><span id="distance">
                            <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true">
                            </span></span>     <span     id="sub_distance"
style="display: none">
                            см
                        </span>
                    </p>
                </div>
            </div>
            <div class="card col-md-3 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img   src="./img/thermometer.png"   alt="temperature"
class="card-title">
                    <h6 class="card-subtitle mb-2 mt-sm-3 mt-lg-0 text-
muted">Температура</span></h6>
                    <p class="card-text">
                        <span id="temperature">
                            <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true">
                            </span>
                        </span>
                        <span id="sub_temperature" style="display: none">
                            °C
```

```
                    </span>
                </p>
            </div>
        </div>
        <div class="card col-md-3 mt-3 mx-auto">
            <div class="card-body text-center">
                <img      src="./img/humidity.png"      alt="humidity"
class="card-title">
                <h6  class="card-subtitle  mb-2  mt-sm-3  mt-lg-0  text-
muted">Вологість</span></h6>
                <p class="card-text">
                    <span id="humidity">
                        <span  class="spinner-border  spinner-border-sm"
role="status" aria-hidden="true"></span>
                    </span>
                    <span id="sub_humidity" style="display: none">
                        %
                    </span>
                </p>
            </div>
        </div>
        <div class="card col-md-3 mt-md-0 mt-3 mx-auto">
            <div class="card-body text-center">
                <img  src="./img/road.png"  alt="odometr"  class="card-
title">
                <h6  class="card-subtitle  mb-2  mt-sm-3  mt-lg-0  text-
muted">Пройдений шлях</span></h6>
                <p class="card-text">
                    <span id="mileage">
                        <span  class="spinner-border  spinner-border-sm"
role="status" aria-hidden="true"></span>
                    </span>
                    <span id="sub_mileage" style="display: none">
                        см
                    </span>
```

```
                    </p>
                </div>
            </div>
            <div class="card col-md-3 mt-md-0 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img  src="./img/gauge.png"  alt="dauge"  class="card-
title">
                    <h6  class="card-subtitle  mb-2  mt-sm-3  mt-lg-0  text-
muted">Атмосферний тиск</span></h6>
                    <p class="card-text">
                        <span id="pressure">
                            <span  class="spinner-border  spinner-border-sm"
role="status" aria-hidden="true"></span>
                        </span>
                        <span id="sub_pressure" style="display: none">
                            кПа
                        </span></p>
                </div>
            </div>
            <div class="card col-md-3 mt-md-0 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img      src="./img/friction.png"      alt="friction"
class="card-title">
                    <h6  class="card-subtitle  mb-2  mt-sm-3  mt-lg-0  text-
muted">Кут нахилу</span></h6>
                    <p class="card-text">
                        <span id="angle">
                            <span  class="spinner-border  spinner-border-sm"
role="status" aria-hidden="true"></span>
                        </span><span id="sub_angle" style="display: none">
                            °
                        </span>
                    </p>
                </div>
            </div>
```

```
            <div class="card col-md-3 mt-md-0 mt-3 mx-auto">
                <div class="card-body text-center">
                    <img src="./img/3d.png" alt="xyz" class="card-title">
                    <h6 class="card-subtitle mb-2 mt-sm-3 mt-lg-0 text-
muted">Акселерометр</span></h6>
                    <p class="card-text">
                        <span id="accelerometr">
                            <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span>
                        </span>
                    </p>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="card w-100 my-3">
                <div class="card-body">
                    <h6 class="card-subtitle mb-2 mt-sm-3 mt-lg-0 text-muted
text-center">Статистика за останні сім днів</span></h6>
                    <canvas id="statistic" width="100%" height="50%"
class="my-3"></canvas>
                </div>
            </div>
        </div>
    </div>
</main>
<script src="/js/jquery-3.6.0.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="./js/charts/getData.js"></script>
<script src="js/charts/conf.js"></script>
<script src="js/refreshSensors.js"></script>
</body>
</html>
```

Файл signin.php

```
<?php
session_start();
include './lib/auth.php';
if (isset($_POST['login'], $_POST['password'])) {
    $auth = auth($_POST['login'], $_POST['password']);
    exit($auth);
}
if (isset($_SESSION['u_name'])){
    header('Location: /');
    exit();
}
?>
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <title>Авторизація | СТРП</title>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <div class="container">
        <a href="/" class="navbar-brand">СТРП</a>
    </div>
</nav>
<main>
    <div class="container">
        <div class="row">
            <div class="col-12 col-md-6 mt-3 mx-auto">
                <div class="modal-header">
```

```
                    <h1>Авторизація</h1>
                </div>
                <div   class="alert   alert-danger   mt-3"   role="alert"
id="formAlert" style="display: none">
                </div>
                <form method="POST" id="loginForm">
                    <div class="mb-3 mt-3">
                        <label for="login" class="form-label">Логін</label>
                        <input  type="text"  class="form-control"  id="login"
name="login">
                    </div>
                    <div class="mb-3">
                        <label for="password" class="form-label">
                            Пароль
                        </label>
                        <input     type="password"     class="form-control"
id="password" name="password">
                    </div>
                    <button  id="btn_signin"  class="btn  btn-primary  w-100"
type="button">
                        Увійти
                    </button>
                </form>
            </div>
        </div>
    </div>
</main>
<script src="/js/jquery-3.6.0.min.js"></script>
<script src="./js/auth.js"></script>
</body>
</html>
```

Файл control.php

```
<?php
session_start();
if (empty($_SESSION['u_name'])) {
    header('Location: /signin.php');
    exit();
}
?>
<!doctype html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.4.1/font/bootstrap-icons.css">
    <title>Віддалене керування | СТРП</title>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-3">
    <div class="container">
        <a href="#" class="navbar-brand">СТРП</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarContent" aria-controls="navbarContent" aria-
expanded="false"><span class="navbar-toggler-icon"></span>
        </button>


        <div class="collapse navbar-collapse" id="navbarContent">
            <ul class="navbar-nav mr-auto mb-1 mt-2">
                <li class="nav-item">
                    <a href="/" class="nav-link">Головна <i class="bi bi-
easel"></i></a>
                </li>
                <li class="nav-item">
```

```
                    <a href="/control.php" class="nav-link">Віддалене
керування <i class="bi bi-controller"></i></a>
                </li>
            </ul>
            <div class="d-flex">
                <h5 class="mr-3 mt-2 text-
white"><?=$_SESSION['u_name']?></h5>
                <a href="/?action=logout" class="btn btn-outline-
danger">Вихід <i class="bi bi-box-arrow-right"></i></a>
            </div>
        </div>
    </div>
</nav>
<main>
    <div class="container">
        <div class="row sticky-top">
            <table class="table table-borderless">
                <thead>
                    <tr>
                        <th scope="col" class="text-left">
                            <svg xmlns="http://www.w3.org/2000/svg"
width="32" height="32" fill="currentColor" class="bi bi-wifi mx-3"
viewBox="0 0 16 16">
                                <path d="M15.385 6.115a.485.485 0 0 0-.048-
.736A12.443 12.443 0 0 0 8 3 12.44 12.44 0 0 0 .663 5.379a.485.485 0 0 0-
.048.736.518.518 0 0 0 .668.05A11.448 11.448 0 0 1 8 4c2.507 0 4.827.802
6.717 2.164.204.148.489.13.668-.049z"/>
                                <path d="M13.229 8.271c.216-.216.194-.578-
.063-.745A9.456 9.456 0 0 0 8 6c-1.905 0-3.68.56-5.166 1.526a.48.48 0 0 0-
.063.745.525.525 0 0 0 .652.065A8.46 8.46 0 0 1 8 7a8.46 8.46 0 0 1 4.577
1.336c.205.132.48.108.652-.065zm-2.183 2.183c.226-.226.185-.605-.1-
.75A6.472 6.472 0 0 0 8 9c-1.06 0-2.062.254-2.946.704-.285.145-.326.524-
.1.75l.015.015c.16.16.408.19.611.09A5.478 5.478 0 0 1 8 10c.868 0 1.69.201
2.42.56.203.1.45.07.611-.091l.015-.015zM9.06 12.44c.196-.196.198-.52-.04-
.66A1.99 1.99 0 0 0 8 11.5a1.99 1.99 0 0 0-1.02.28c-.238.14-.236.464-
.04.66l.706.706a.5.5 0 0 0 .708 0l.707-.707z"/>
                            </svg><br>
                            <span id="signal">-</span> dbm
```

```
                        </th>
                        <th scope="col" class="text-right">
                            <svg xmlns="http://www.w3.org/2000/svg"
width="32" height="32" fill="currentColor" class="bi bi-battery-full text-
success mx-1" viewBox="0 0 16 16">
                                <path d="M2 6h10v4H2V6z"/>
                                <path d="M2 4a2 2 0 0 0-2 2v4a2 2 0 0 0 2
2h10a2 2 0 0 0 2-2V6a2 2 0 0 0-2-2H2zm10 1a1 1 0 0 1 1 1v4a1 1 0 0 1-1
1H2a1 1 0 0 1-1-1V6a1 1 0 0 1 1-1h10zm4 3a1.5 1.5 0 0 1-1.5 1.5v-3A1.5 1.5
0 0 1 16 8z"/>
                            </svg><br>
                            <span id="battery">-</span>%
                        </th>
                    </tr>
                </thead>
            </table>
        </div>
        <div class="row">
            <div class="card mx-auto col-9 mb-3">
                <img src="http://twfi.pp.ua:27500/" alt="Відео відсутнє"
class="card-img-top text-center mt-3">
                <div class="card-body">
                    <table class="table text-center table-borderless table-
sm">
                        <thead>
                        </thead>
                        <tbody>
                            <tr>
                                <td colspan="3"><button title="W"
id="ctrl_up" class="btn btn-dark w-100">Δ</button></td>
                            </tr>
                            <tr>
                                <td class="w-33"><button title="A"
id="ctrl_left" class="btn btn-dark w-100">◁</button></td>
                                <td class="w-33"><button title="CTRL"
id="ctrl_stop" class="btn btn-dark w-100">stop</button></td>
```

```
                                <td class="w-33"><button title="D"
id="ctrl_right" class="btn btn-dark w-100">▷</button></td>
                            </tr>
                            <tr>
                                <td colspan="3"><button title="S"
id="ctrl_down" class="btn btn-dark w-100" type="button">▽</button></td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</main>
<script src="/js/jquery-3.6.0.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
<script src="/js/control.js"></script>
</body>
</html>
```

Файл /lib/auth.php

```
<?php
function auth($login, $pass) {
    if(!empty($login) && !empty($pass)){
        include 'db.php';
        $db_query = mysqli_query($db_link, "SELECT id, password FROM users
WHERE login ='".mysqli_real_escape_string($db_link, $login)."' LIMIT 1");
        $db_data = mysqli_fetch_assoc($db_query);
        if($db_data['password'] === md5($pass)){
            $_SESSION['u_name'] = $login;
            $_SESSION['u_id'] = $db_data['id'];
        return 'Ok';
        } else {
```

```
            return 'Невірний логін або пароль';
        }
    } else {
        return "Не введено логін або пароль";
    }
}
?>
```

Файл /lib/command.php

```
<?php
$commandFile = 'current_command.txt';
if (!empty($_GET['command'])){
    $file = fopen($commandFile, 'w');
    fputs($file, $_GET['command']);
    fclose($file);
    print('Writed '.$_GET['command']);
    exit();
}
if ($_GET['control'] == 'get') {
    $resp = file_get_contents($commandFile);
    header('Content-Type: application/json');
    print("{\"control\":".$resp."}");
    exit();
}
header('Location: /');
?>
```

Файл /lib/db.php

```
<?php
$host = "localhost";
$db_user = "strp";
$db_password = "hhdAQ12MjjdOOs";
```

```
$db_table = "strp";
$db_link = mysqli_connect($host, $db_user, $db_password, $db_table);
?>
```

Файл sensors.php

```
<?php
    function writeSensors()
    {
        include './db.php';
        $data = $_GET['data'];
        $data = json_decode($data, true);
        $accelerometer = json_encode($data['acc']);
        $gyroscope = json_encode($data['gyr']);
        $angle = json_encode($data['ang']);
        $db_query = mysqli_query($db_link, "INSERT INTO `sensors` (`id`, `time`, `temperature`, `humidity`, `distance`, `speed`, `mileage`, `accelerometr`, `gyroscope`, `angle`, `pressure`, `height`, `bvoltage`, `pbattery`, `signal_level`, `message`) VALUES (NULL, CURRENT_TIMESTAMP, '{$data['tmp']}', '{$data['hum']}', '{$data['u_dst']}', '{$data['spd']}', '{$data['mil']}', '$accelerometer', '$gyroscope', '$angle', '{$data['prs']}', '{$data['alt']}', '{$data['bat']}', '{$data['bpc']}', '{$data['sig']}', '{$data['message']}')");
        if (!$db_query) {
            print_r("ERROR");
        } else {
            print_r("Ok");
        }
    }
    function dbRequest($sql)
    {
        include 'db.php';
        $db_query = mysqli_query($db_link, $sql);
        $db_data = mysqli_fetch_assoc($db_query);
        return $db_data;
    }
```

```
    function getCharts()
    {
        include './db.php';
        $keys = array('speed', 'temperature', 'mileage', 'humidity',
'pressure');
        $speed = $temperature = $mileage = $humidity = $pressure = array();
        $date = array();

        foreach ($keys as $index) {
            for ($i = 1; $i <=7; $i++){
                $sql = mysqli_query($db_link, "SELECT AVG({$index}) FROM
sensors WHERE time >= (curdate()-{$i}) AND time < curdate()");
                $data = mysqli_fetch_assoc($sql);
                if(empty($data['AVG('.$index.')'])) {
                    $value = 0;
                } else {
                    $value = round($data['AVG('.$index.')'], 2);
                }
                array_push($$index, $value);
            }

        }
        for ($i =1; $i <= 7; $i++) {
            $oDate = new DateTime();
            $oDate->modify('-'.$i.' day');
            $pushDate = $oDate->format('d.m');
            array_push($date, $pushDate);
        }
        $meanValues = array('speed' => array($speed), 'temperature' =>
array($temperature), 'mileage' => array($mileage), 'humidity' =>
array($humidity), 'pressure' => array($pressure), 'date' => array($date));
        header('Content-Type: application/json');
        print(json_encode($meanValues));
    }
```

```
    if($_GET['action'] === "getSensors") {
        $sql = "SELECT * FROM `sensors` ORDER BY `id` DESC limit 1";
        $sql2 = "SELECT `id` FROM `sensors` WHERE `time` >= NOW() - INTERVAL
15 SECOND ORDER BY `id` DESC limit 1";;
        $data = dbRequest($sql);
        $data2 = dbRequest($sql2);
        if ($data['id'] != $data2['id']) {
            print('offline');
            exit();
        } else {
            $data = json_encode($data);
            header('Content-Type: application/json');
            print($data);
            exit();
        }
    }
    if($_GET['action'] === "getSensorsMin") {
        header('Content-Type: application/json');
        $sql = "SELECT `signal_level`, `pbattery` FROM `sensors` ORDER BY
`id` DESC limit 1";
        $data = dbRequest($sql);
        $data = json_encode($data);
        print($data);
    exit();
    }
    if($_GET['action'] === "getCharts") {
        getCharts();
        exit();
    }
    if(isset($_GET['data'])){
        writeSensors();
        exit();}
?>
```

Файл /js/charts/conf.js

```
const labels = values['date']['0'];
const data = {
    labels: labels,
    datasets: [
        {
            label: 'Пройдений шлях',
            data: values['mileage'][0],
            borderColor: [
                'rgba(255, 99, 132, 1)'
            ],
            backgroundColor: [
                'rgba(255,99,132,0.2)'
            ],
        },
        {
            label: 'Середня температура',
            data: values['temperature'][0],
            borderColor: [
                'rgba(54, 162, 235, 1)'
            ],
            backgroundColor: [
                'rgba(155,196,221)'
            ],
        },
        {
            label: 'Середня швидкість',
            data: values['speed'][0],
            borderColor: [
                'rgb(54,235,114)'
            ],
            backgroundColor: [
```

```
                'rgb(187,226,179)'
            ],
        },
        {
            label: 'Середня вологість',
            data: values['humidity'][0],
            borderColor: [
                'rgb(160,54,235)'
            ],
            backgroundColor: [
                'rgb(212,163,214)'
            ],
        },
        {
            label: 'Атмофсерний тиск',
            data: values['pressure'][0],
            borderColor: [
                'rgb(235,184,54)'
            ],
            backgroundColor: [
                'rgb(214,201,163)'
            ],
        }
    ]
};
const options = {
    responsive: true,
    plugins: {
        legend: {
            position: 'top',
        },
        title: {
            display: false,
```

```
            text: 'Статистика'
        }
    }
};

var ctx = document.getElementById('statistic');
var statictic = new Chart(ctx, {
    type: 'line',
    data: data,
    options: options
    });
```

Файл /js/charts/getData.js

```
var values;
function getChartsData() {
    $.ajax({
        url: '/lib/sensors.php?action=getCharts',
        type: 'GET',
        async: false,
        success: function (resp){
            values = resp;
        },
        error: function (){
            alert('Failed load charts data');
        }
    });
}
getChartsData();
```

Файл /js/auth.js

```
$(document).ready(function(){
    var loadingText = "<span class=\"spinner-border spinner-border-sm\" role=\"status\" aria-hidden=\"true\"></span> <span class=\"visually-hidden\">Почекайте...</span>";
    $('#btn_signin').click(function(){
        $('#btn_signin').html(loadingText);
        $('#btn_signin').prop("disabled", true);
        sendForm('loginForm', 'signin.php');
        return false;
    });
    function sendForm(form, url) {
        $.ajax({
            url: url,
            type: "POST",
            dataType: "html",
            data: $("#"+form).serialize(),
            success: function (responce){
                if(responce == 'Ok'){
                    $('#formAlert').css('display', 'none');
                    $('#btn_signin').html('Увійти');
                    $('#btn_signin').prop("disabled", false);
                    location.reload();
                } else {
                    $('#btn_signin').html('Увійти');
                    $('#btn_signin').prop("disabled", false);
                    $('#formAlert').html(responce);
                    $('#formAlert').css('display', 'block');
                }
            },
            error: function(responce){
                $('#btn_signin').html('Увійти');
                $('#btn_signin').prop("disabled", false);
```

```
                $('#formAlert').html('Перевірте підключення до мережі')
            }
        });
    }
    $('#password').keypress(function (e) {
        if (e.which == 13) {
            $('#btn_signin').html(loadingText);
            $('#btn_signin').prop("disabled", true);
            sendForm('loginForm', 'signin.php');
            return false;
        }
    })
});
```

Файл /js/control.js

```
$(document).ready(function(){
    function WriteCommand(command){
        $.get('/lib/command.php?command='+command).done (
            function(resp){
                console.log(resp);
            }
        )
    }
    $('#ctrl_up').click(function () {
        WriteCommand('5');
    });
    $('#ctrl_down').click(function () {
        WriteCommand('1');
    });
    $('#ctrl_left').click(function () {
        WriteCommand('2');
    });
```

```
$('#ctrl_right').click(function () {
    WriteCommand('3');
});
$('#ctrl_stop').click(function () {
    WriteCommand('4');
});
$(document).keyup(function(e) {
    if (e.keyCode === 87) {
        WriteCommand('5');
    }
    if (e.keyCode === 65) {
        WriteCommand('2');
    }
    if (e.keyCode === 83) {
        WriteCommand('1');
    }
    if (e.keyCode === 68) {
        WriteCommand('3');
    }
    if (e.keyCode === 17) {
        WriteCommand('4');
    }
});
function refreshMinSensors(){
    $.get('/lib/sensors.php?action=getSensorsMin').done (
        function(resp){
            var sensors = resp;
            $('#signal').html(sensors['signal_level']);
            $('#battery').html(sensors['pbattery']);
            console.log(resp);
        }
    )
}
```

```
    setInterval(function(){
        if (!document.hidden) {
            refreshMinSensors();
        }
    }, 1500);
});
```

Файл /js/refreshSensors.js

```
$(document).ready(function(){
    function refreshSensors() {
            $.get('/lib/sensors.php?action=getSensors').done
(function(responce){
                var attrArray = ['signal_level', 'pbattery', 'speed',
'distance', 'temperature', 'humidity', 'mileage', 'pressure', 'angle',
'accelerometr'];
                if (responce !== 'offline') {
                    var sensorsArray = responce;
                    attrArray.forEach(function (item, i, arr) {
                        $('#sub_' + item).css('display', 'inline');
                        $('#' + item).html(sensorsArray[item]);
                    });
                } else {
                    attrArray.forEach(function (item, i, arr) {
                        $('#' + item).html('-');
                        $('#sub_' + item).css('display', 'none');
                    });
                    $('#signal_level').html('offline');
                }
            });
    }
    setInterval(function (){
        if (!document.hidden) {
            refreshSensors();}}, 1500);});
```