МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Система моніторингу підключення USB-пристроїв для проведення аудиту кібербезпеки»

Завідувач кафедрою **Довбиш А.С.**

Керівник роботи **Кальченко В.В.**

Студент групи КБ-61 **Татарінов В.Р.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи КБ-61 Татарінов В.Р. спеціальності "Кібербезпека" денної форми навчання.

Тема: "**Система моніторингу підключень USB-пристроїв для проведення аудиту кібербезпеки**"

Затверджена наказом по СумДУ

№_____ от _____2020р.

Зміст пояснювальної записки: 1) аналіз предметної області; 2) постановка задачі та мета дослідження; 3) програмна реалізація системи моніторингу підключень USB-пристроїв

Дата видачі завдання "_____"_____20___р.

Керівник випускної роботи _____ Кальченко В.В.

Завдання приняв до виконання _____ Татарінов В.Р.

РЕФЕРАТ

Записка: 117 стор., 45 рис., 3 додатоки, 24 джерел.

Об’єкт дослідження — процес централізованого отримання інформації про USB-пристрої, які підключаються до комп’ютерів в локальній мережі.

Мета роботи — спрощення процедури контролю використання зовнішніх носіїв інформації в локальній мережі.

Методи дослідження — методи отримання інформації про реєстрацію USB-пристроїв з реєстру операційної системи Windows.

Результати — розроблено систему моніторингу підключення USB-пристроїв, яка складається з клієнтської служби, серверної служби та веб-додатку.

USB-ПРИСТРІЙ, КЛІЄНТ-СЕРВЕРНА ПРОГРАМА, ВЕБ-ДОДАТОК, КЛІЄНТ, СЕРВЕР, КОНТРОЛЬ ЗА ДІЯМИ КОРИСТУВАЧІВ, СЛУЖБА, PYTHON, C#

Зміст

Вступ

Доступність, малий розмір та збільшення ємності USB-накопичувачів ставить під великий сумнів інформаційну безпеку компаній. Ігнорування цієї загрози може призвести до значних матеріальних втрат, або навіть банкротства. Через підключення до USB-порту, зловмисник може поцупити важливу інформацію або встановити шкідливе програмне забезпечення. Усе більше компанії збільшують інвестиції в міжмережеві екрани, використовують більш надійні та сучасні алгоритми шифрування, інші засоби й технології контролю для захисту даних від розкрадання через Інтернет. Однак, найчастіше зловмисниками стають працівники компанії, які крадуть відомості про інвестиції, проектні документи, комерційні пропозиції, особисту інформацію та контактні дані клієнтів і багато іншого, заради матеріальної наживи або помсти. USB-пристрій становитися ідеальним інструментом, для втілення цих злочинних намірів.

Відключення USB-портів є радикальний захід безпеки, що призведе до істотних незручностей та неможливості підключити периферійні пристрої до комп’ютера. Задання списку дозволених пристроїв за допомогою серійного номеру є гарним заходом протидії витоку інформації, але недостатнім. Серійний номер флешки може бути з легкістю змінений, на який завгодно, що дасть змогу порушнику без яких-небудь перешкод отримати доступ до важливих даних.

Суттєвим заходом безпеки є моніторинг підключення USB-пристроїв до корпоративних комп’ютерів. Централізоване відстеження підключень є ефективним способом для розслідування інцидентів витоку інформації через USB канали, що дозволить керівництву відділу безпеки або адміністратору безпеки своєчасно відреагувати, та встановити пристрій, дату, час і навіть комп’ютер, де відбувся інцидент. Аналіз цих даних та виявлення факту

підключення пристрою, допоможе успішно встановити особу зловмисника та вжити необхідних заходів.

Метою роботи є розробка системи моніторингу підключень USB-пристроїв для невеликих компаній, які не мають можливості придбати дорогі системи з непотрібними для вирішення цієї задачі функціями.

Новизна роботи полягає в наступних чинниках:

- більшість безкоштовних рішень, які існують, не мають централізованого аналізу подій підключення USB-пристроїв до корпоративних комп'ютерів;
- нездатність рішень, які існують, розгорнутися локально, та потребують щомісячної або річної платної підписки за обслуговування даного сервісу;
- відсутність у безкоштовних рішеннях достатнього функціоналу для організації ефективного моніторингу підключення USB-пристроїв;
- громіздкість та переповнення непотрібним функціоналом платних рішеннях, який потребує додаткових витрат;
- більшість існуючих рішень мають незручний, та інтуїтивно не зрозумілий інтерфейс;
- висока вартість платних рішень, що не під силу невеликим компаніям.

Дослідивши недоліки наявних рішень та визначивши основні потреби, було прийнято рішення розробити систему моніторингу підключень USB-пристроїв лише з потрібним функціоналом, з можливістю розгорнути її локально, здатністю централізованого збору подій та подальшого представленням зібраної інформації за допомогою зручного й інтуїтивно зрозумілого веб-інтерфейсу, що зробить дану систему унікальною серед наявних рішень.

Основні задачі, які треба вирішити для досягнення мети:

- виконати аналіз предметної області;
- дослідити документацію Windows;
- обрати засоби розробки;
- спроектувати систему;
- реалізувати службу клієнт, яка буде спостерігати за підключеннями USB-накопичувача;
- реалізувати серверну службу для централізованого збору подій;
- реалізувати веб-додаток для перегляду та аналізу подій;
- виконати тестування.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз документації Windows

Об’єктом над, яким буде проводитися спостереження, є комп’ютер на базі операційної системи Windows. Звідси для вирішення поставленої задачі було досліджено документацію операційної системи Windows.

Починаючи з Windows 2000 всі операційні системи Windows містять модуль для централізованого управління та спостереженням за різними частинами комп’ютерної інфраструктури під назвою Windows Management Instrumentation[1].

Технологія WMI реалізує стандарт WBEM, основою якого є ідея створення інтерфейсу для моніторингу та управління різними частинами комп’ютерної системи, використовуючи об'єктно-орієнтований підхід[1].

WMI представляє дані, які належать до операційної системи у вигляді об’єктів. Об’єкти це члени класу, а класи це члени простору імен та весь простір імен походить від «Root». Отримання інформації про системні компоненти або зміна параметрів налаштувань відбуваються через представлені об’єкти, які можна отримати за допомогою WMI Query Language[4].

WMI Query Language – це мова запитів, яка схожа на мову запитів SQL та дозволяє отримувати об’єкти, класи та простори імен [4]. Ці запити регулярно використовуються зловмисниками для отримання контролю над системою, але також вони є гарним інструментом для забезпечення захисту системи.

Виділяють три категорії запитів:

- Запити для отримання об’єктів. Об’єкти WMI використовуються для збору інформації або зміни параметрів;

- Запити подій. Вони використовуються, щоб зареєструвати процес відслідковування над об’єктом.
- Мета запити. Використовуються для отримання схеми класів.

Оскільки майже всі дії операційної системи викликають події WMI, то це можна використовувати для відслідковування подій підключень USB-пристроїв.

WMI дає можливість запускати та зупиняти сеанси відслідковування подій. Події трасування містять заголовок події та дані, визначені постачальником події, які описують поточний стан програми або операції. Тому події можна використовувати для отримання інформації в змінах поточного стану системи.

Перед тим, як почати відслідковувати потрібно визначити вид події, які потрібно отримувати: внутрішні або зовнішні.

Внутрішні події – це події, які відбуваються під час виникнення змін в стандартній моделі даних WMI. В кожному класі відбувається зміни в об’єктах, наприклад при підключенні нового USB-пристрою, створюється новий об’єкт класу Win32_USBHub, який призведе до створення екземпляру класу InstanceCreationEvent[3].

Зовнішня подія – це подія, яку не можна безпосередньо пов’язати зі змінами в моделі даних WMI[3].

Хоча відслідкувати подію підключення USB-пристрою дозволяє два різних типи подій, для нашого випадку кращим варіантом є внутрішні події, бо вони дозволяють дізнатися не тільки про те, що подія відбулася, але й містять створений об’єкт з інформацією про під’єднаний пристрій.

1.2. Аналіз наявних аналогів

Під час аналізу наявних рішень, було виявлено п’ять програмних продуктів, які можуть повністю або частково виконати задачу моніторингу підключень USB-пристроїв. Давайте детально поглянемо на них.

USB Canary

USB Canary, цей утиліта написана на Python і на цей час працює тільки на Linux, але розробник працює над версіями для Windows і Mac OS[7].

Принцип роботи утиліти простий: коли комп’ютер заблокований, і його власник кудись відійшов, USB Canary пильно спостерігає за роз’ємами USB, помічаючи будь-яку активність. Якщо хтось спробує підключити до комп’ютера якийсь пристрій або навпаки щось відключить, USB Canary повідомить про це власника ПК, відправивши йому SMS-повідомлення через Twilio API, або пославши відповідне попередження в Slack-канал[7], який також можуть переглядати колеги користувача або системний адміністратор. Крім того, утиліту можна налаштувати на певні години, тобто вона буде виявляти підключення в неробочий час. USB Canary працює, навіть у тому випадку, якщо комп’ютер заблокований.

Визначення недоліків

З найбільших недоліків цієї утиліти можна виокремити відсутність версії для операційної системи Windows, яка є найбільш розповсюдженою ОС у світі. Наявність цього факту зменшує множину потенційних користувачів. Також утиліта має недостатній функціонал для забезпечення моніторингу підключень USB-пристроїв для великих компаній. Адміністраторам безпеки потрібно відслідковувати значну кількість комп’ютерів, що без централізованого аналізу подій здійснити це ефективно майже неможливо. Звідси можемо зробити

висновок, що функціонал USB Canary є достатнім для невеликих компаній, які використовують Linux, або для персонального використання, що дасть змогу повідомити користувача про спробу поцупити файл під час його відсутності.

USBDeview

USBDeview здатна відображати інформацію про USB-пристрої, не тільки, які під'єднанні на момент запуску програми, але і всі ті, які підключалися до комп'ютера в якийсь момент у минулому[8]. За допомогою USBDeview можна побачити список усіх USB-пристроїв, включно не тільки ім'я пристрою, але й короткий опис, тип пристрою, номер виробника та серійний номер. Інші відомості в цьому списку включають інформацію про те, коли пристрій використовувався востаннє, чи підключений він у цей момент і чи можна його безпечно видалити. Також отримана інформація може бути експортована в txt, HTML або навіть XML файл[8].

Уся інформація виводиться програмою в зручному табличному вигляді. Також є можливість змінити число стовпців (включати потрібні і прибирати зайві) і їхнє розташування в інтерфейсі програми, та також сортувати по стовпцям.

Меню Файл в USBDeview включає в себе опції для взаємодії з USB-пристроями, такі як включення або відключення їх, а також відключення всіх одночасно[8]. USBDeview також дозволяє кастомізувати вивід стовпців відповідно до потреб користувача.

До того ж, програма USBDeview допоможе позбутися від усіх непотрібних записів у системному реєстрі, що значно збільшити швидкість його роботи.

Визначення недоліків

Недоліком цієї програми так само, як і USB Canary є відсутність можливості централізованого перегляду подій. Без централізованого моніторингу подій USB-підключень збільшується час реагування на витік інформації, що може коштувати компанії великих збитків.

Хоча USBDeview доволі зручно та детально виводить інформацію про USB-накопичувачі, але неможливість переглянути історію підключень робить цю програму нездатною для результативного моніторингу. Відображення тільки останнього час підключення не дає можливості встановити повну хронологію дій користувача, що може завадити доказу провини зловмисника.

CleverControl

CleverControl — це рішення для моніторингу співробітників, що дає змогу відстежити їхні дії на комп'ютерах[9]. Це ідеальний інструмент для заохочення оптимальної продуктивності праці працівників, підвищення її ефективності та запобігання незаконним діям, таких, як витік інформації, крадіжка особистих даних і несанкціонований доступ. CleverControl надає користувачам у режимі реального часу віддалений контроль і моніторинг усіх різних комп'ютерних дій, включно веб-серфінг, пошук в Інтернеті, сеанси чату, електронну пошту, соціальні мережі й обмін файлами[9]. Вона також відстежує взаємодію з іншими пристроями і сховищами, такими, як принтери і USB-накопичувачі. CleverControl — це хмарна платформа для моніторингу, доступ, до якої здійснюється через веб-аккаунт. Немає необхідності у виділеному сервері, і його налаштування дуже просте й не вимагає досвіду IT-фахівця[9].

Визначення недоліків

CleverControl надає користувачу обширний і в деяких моментах надлишковий функціонал, що змушує користувача сплачувати за непотрібні йому функції.

Надання послуг у вигляді хмарного сервісу може бути одночасно, як недоліком, так і перевагою. З одного боку, вам не потрібно мати власний сервер, адміністратора, який мусить забезпечувати роботу системи, а з іншого боку, встановлюючи цей продукт на комп'ютери, компанія добровільно відаєте всі свої відомості серверам CleverControl, що в результаті може призвести до повного витоку інформації через цей сервіс.

Також, цей сервіс вимагає щомісячної підписки, без можливості одноразово придбати продукт, та самостійно розгорнути його на власних серверах.

Solarwinds Security Event Manager

Менеджер подій безпеки (SEM) — це рішення SIEM ACTIVE для моніторингу, яке автоматично виявляє, сповіщає й реагує на підозрілу поведінку на мережевих пристроях, серверах, робочих станціях і додатках різних постачальників[10]. SEM являє собою завантажувальний програмний комплекс, що швидко розгортається й забезпечує оперативний аналіз загроз і кореляцію подій у реальному часі, забезпечуючи більш швидке реагування на кібератаки.

Цей програмний продукт, має широкий функціонал, який нараховує 58 use case. Зараз детально розглянемо use case, який здатний вирішувати задачу моніторингу підключень USB-пристроїв.

За допомогою функції USB Security Analyzer Detects and Responds to Potential Threats можна підвищити безпеку USB-порту[10], використовуючи попередньо встановлені готові правила активної відповіді або, створивши свої

власні правила. Правила активного реагування призначені для блокування USB-пристроїв, коли доступ до USB становить загрозу—наприклад, якщо пристрій, занесено в чорний список. Використовуючи USB Defender, можна встановити правила активної відповіді, щоб автоматично перевірити, чи належить пристрій до білого списку USB-пристрої, що визначається користувачем. Якщо виявлено несанкціоноване підключення до USB-пору, виконуються превентивні заходи, такі як автоматичне від'єднання USB-пристрою, знищення процесів за ідентифікатором або імені, блокування IP-адрес і вимикання машин[10].

USB Defender також записує події, пов'язані з USB-пристроями, підключеними до агентів SEM за замовчуванням. Ці події можуть бути проаналізовані для отримання іншої інформації для створення більш складних порівнянь чорного списку. Для звітності, SEM пропонує, як вбудовані, так і настроюванні шаблони звітів.

Визначення недоліків

Solarwinds Security Event Manager окрім функції USB Security Analyzer Detects and Responds to Potential Threats має ще 57 різних функцій моніторингу та управління різними аспектами системи, що для кінцевого користувача може бути надлишковим, для вирішення поставленої задачі. Хоча користувач може не встановлювати непотрібні йому функції, на кінцеву ціну підписки це не впливає. Ціна залежить від кількості вузлів, на яких буде встановлений програмний продукт.

Незважаючи на те, що дане рішення є чудовим засобом для вирішення задачі моніторингу підключень USB-пристроїв і навіть захисту USB-портів від небажаного підключення, початкова ціна за річну підписку на момент написання роботи стартує від 4,805 доларів США за 30 вузлів, ціна за обслуговування понад 100 вузлів сягає від 14,694.86 доларів США, що робить це рішення доволі дорогим для невеликих або, навіть середніх компаній.

Звідси можемо зробити висновок, що цей продукт, є гарним рішенням лише для великих компаній, зі значними вимогами забезпечення інформаційної безпеки.

Alien Vault

Ця уніфікована потужна платформа дозволяє організаціям швидко виявляти загрози і реагувати на них, комбінуючи численні важливі функції безпеки. Вона включає в себе виявлення активів, управління вразливостями, виявлення вторгнень, моніторинг поведінки, SIEM та управління журналами. AlienVault розроблений для захисту хмарної, локальної та гібридної інфраструктури, забезпечуючи постійний аналіз загроз і дієве реагування на інциденти. Вона здатна контролювати Office 365, Google Suite, Microsoft Azure Cloud та Amazon Web Services[11]. Також має зручний графічний інтерфейс. Існує також візуалізація даних з діаграмами, графіками, картами та іншими графічними елементами. AlienVault заснований на хмарі, тому швидко розгортається і дозволяє користувачам почати виявляти загрози протягом декількох хвилин. Його основні можливості включають виявлення активів, виявлення вторгнень, автоматизацію безпеки, SIEM і управління журналами, виявлення загроз, аналіз загроз і оцінку вразливостей[11].

Також серед наведених можливостей, платформа здатна виявляти підключення зовнішнього пристрою до USB-порту системи. Ці пристрої включають зовнішні накопичувачі або флеш-накопичувачі.

Визначення недоліків

Як і у випадку з CleverContro, даний сервіс представляє свої послуги на базі хмарних сервісів. Кінцевий користувач не зможе розвернути процес моніторингу за USB-пристроями, за допомогою власних обчислювальних можливостей, та вимушені надсилати корпоративні дані на сервера Alien Vault, що вимагає більш детального вивчення процесу захисту даних, які здійснює цей сервіс. Також,

функція моніторингу підключень USB-накопичувачів поставляється з іншим функціоналом, який є надмірним, для вирішення поставленої задачі. Користувач не має можливості сплачувати за потрібну йому функцію. Підписку на даний сервіс також не можна назвати доступною для малих компаній, що становить на момент написання роботи 1075 доларів США на місяць.

Звідси можемо зробити висновок, що цей продукт, так само, як і Solarwinds Security Event Manager, є чудовим рішенням лише для великих компаній, з значними вимогами забезпечення інформаційної безпеки.

Отже, огляд наявних рішень показав, що для здійснення моніторингу підключень USB-пристроїв, можна застосувати, як платні так і безкоштовні продукти. Мінусом безкоштовних рішень є недостатній функціонал, який не задовольняє віх потреб для повноцінного вирішення поставленої задачі, а платні містять надлишкові функції за які треба платити, без можливості придбати, те що треба.

Визначимо основні характеристики системи моніторингу USB-підключень, які задовольняють потребам невеликої компанії, та визначимо відповідність наявних рішень цим вимогам (див. в табл.).

Таблиця 1.1 - Порівняння наявних рішень

Назва критерію	Назва рішення				
	USB Canary	USBDeview	CleverControl	Solarwinds Security Event Manager	Alien Vault
Можна розвер-нути на власних комп’ютерах	+	+	-	+	-

Продовження таблиці 1.2

Назва критерію	Назва рішення				
	USB Canary	USBDeview	CleverControl	Solarwinds Security Event Manager	Alien Vault
Централізований перегляд подій	-	-	+	+	+
Можливість перегляду історії подій	-	-	+	+	+
Перегляд даних про USB-прист. та дані про вузол(mac, ip та інш.)	-	+/-	+	+	+
Має недостатній функціонал	+	+	-	-	-
Містить зайві функції	-	-	+	+	+
Підтримка Windows	-	+	+	+	+
Зрозумілий інтерфейс	-	+	+	+	+
Ціна	Безкош-товно	Безкош-товно	Від 180$ / рік.	Від 4,805$ за 30 вузлів / рік	1075$/ місяць

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТА ДОСЛІДЖЕННЯ

2.1. Постановка задачі та визначення мети

Метою проекту є створити систему моніторингу підключень USB-пристроїв для проведення аудиту кібербезпеки.

Розробка системи орієнтована на невеликі компанії, які мають потребу забезпечити контроль за USB-пристроями та не мають змогу придбати дорогі системи з непотрібними для вирішення цієї задачі функціями. Система зменшить час реагування на витік інформації через USB-накопичувачі та створить доказову базу, яка допоможе визначити особу порушника та довести його провину.

Розроблювана система має такі вимоги:

- відслідковувати події підключення на корпоративних комп’ютерах;
- забезпечити централізований збір та збереження подій, для подальшого аналізу;
- мати інтуїтивно зрозумілий веб-інтерфейс для зручного перегляду та аналізу подій;
- забезпечувати можливість пошуку потрібних подій;
- доступ до перегляду подій має бути авторизований;
- можливість здійснювати управління користувачами веб-додатку та видалення неактуальної інформації. Доступ до цих функцій повинен надаватися тільки користувачам з правами адміністратора.
- здатність відновлювати пароль;
- здатність розгорнутися локально;

Задачі, які потрібно вирішити для досягнення цих вимог:

- здійснити аналіз предметної області;
- дослідити документацію Windows;
- обрати засоби для розробки системи;

- налаштувати середовище для розробки програмного коду;
- розробити службу клієнт, для спостереження за USB-пристроями;
- розробити серверну службу, для централізованого збору подій та запису їх до бази даних;
- спроектувати структуру бази даних;
- розробити веб-додаток для зручного перегляду подій та аналізу подій;
- протестувати систему;

Технічне завдання до проекту знаходиться в додатку А.

2.2. Вибір засобів реалізації

Для реалізації системи моніторингу підключень USB-пристроїв була обрано: мови програмування C# та Python, технології для створення веб-інтересу HTML, CSS та JavaScript, база даних PostgreSQL.

Частина системи, яка відслідковує події підключень USB-пристроїв, повинна реалізовуватися у вигляді Windows служби. Для реалізації Windows служби широко використовують платформу .NET Framework, створеною компанією Microsoft[10]. Ця платформа дозволяє запускати додатки на операційній системі Windows написані на таких мовах, як C# або Visual Basic.

C# - це сучасна, об’єктно-орієнтована, типізована мова програмування розроблена компанією Microsoft[8]. Вона розроблена для Common Language Infrastructure, яка складається з виконуваного коду та середовища виконання. На відміну від мови програмування C та C++ [13], розробнику не потрібно піклуватися за звільнення пам’яті, що зменшує число помилок та робить процес розробки більш простішим. C# має велику кількість документації, яку можна знайти на сайті Microsoft.

Система моніторингу повинна забезпечувати можливість зручного перегляду інформації. В нашому випадку цю функцію буде виконувати веб-додаток. Для реалізації бекенду використовується мова програмування Python.

Python - це інтерпретована високорівнева мова програмування загального призначення з відкритим вихідним кодом, яка чудово підходить для створення веб-додатків. Найбільшою перевагою мови Python для створення веб-додатків є те, що вона є надзвичайно легкою для застосування[15]. Вона має легкий синтаксис майже подібний до людської мови, що робить процес опановування максимально швидким в порівняні з іншими мовами. Коли діло доходить до відображення даних, Python є надзвичайно ефективною мовою для веб-

розробників. Це дозволить легко створювати звіти та візуально представляти дані. Завдяки простому синтаксису Python забезпечує відмінну читабельність коду, що зменшує час орієнтування в ньому. Python дуже гнучка мова програмування та може інтегрувати ряд мов під час програмування. Наприклад, CPython, який є версією Python з C, IronPython, який є іншою гілкою Python створеної для сумісного виконання .NET та C#[15]. Для вирішення деяких задач веб-розробник потребує можливість асинхронного програмування. Python повністю справляється з цією задачею, що дозволяє більш швидше вирішити проблеми. Велика кількість бібліотек для веб-розробки, також є гарною перевагою для використання цієї мови програмування, найбільш відомими прикладами є Django та Flask. Окрім веб-розробки, Python володіє незліченною кількістю бібліотек для вирішення будь-якої задачі.

Окрім бекенду, який реалізує логіку додатку, веб-додаток потребує створення зручного, інтуїтивно зрозумілого інтерфейсу. Базовими технологіями для фронтеду, які підтримує більшість браузерів є HTML, CSS та JavaScript.

HTML – це технологія, яка є основою для сучасних браузерів з відкритим вихідним кодом, для відображення інтернет сторінок. HTML не має можливості створювати динамічну функціональність та не має широкого інструментарію для створення привабливих сторінок. Вона підтримує систему тегів, що визначають скелет кожної сторінки. Використовуючи теги браузер розуміє те, як йому потрібно відображати сторінки. Хоча існують інші технології для створення каркасу сторінок, як XML, перевага більшістю розробників по всьому світу відається HTML[17].

Нажаль перегляд сторінок у вигляді послідовного потоку тегів, без анімацій та ефектів є нестерпним для сучасного користувача. Для виправлення цієї ситуації на краще застосовується технологія CSS.

CSS – це мова стилів, використовується для привабливого представлення HTML тегів[17]. Як і у випадку з HTML, CSS не призначений для створення

логіки додатку. Вона описує, як елементи сторінки будуть відображатися. CSS призначена для відокремлення змісту та контенту. Відокремлення покращує доступність контенту, дає більш гнучкості.

JavaScript – інтерпретована, динамічно типізована мова програмування, підтримує різні парадигми програмування[17]. Хоча мова JavaScript дозволяє створювати різні типи додатків, вона зарекомендувала себе, як мова для веб-сторінок. Виконання скриптів виконується на стороні клієнта та дозволяє реагувати на дії користувача. За допомогою мови JavaScript створено багато фремверків, які дозволяють значно пришвидшити розробку. JavaScript дозволяє зменшити час обробки сторінок так, як не потрібно підключатися до серверу. JavaScript має легкий для розуміння синтаксис, що зменшує час освоєння цієї мови. Широко підтримується сучасними браузерами[17].

Однією з потреб системи є збереження подій. Звідси виникає необхідність вибору бази даних. Для реалізацію цього проекту було обрано PostgreSQL.

PostgreSQL – це потужна, з відкритим вихідним кодом система управління базами даних[16]. Вона легка при встановленні та не потребує складних налаштувань. Використовує клієнт-серверну модель, а процес сервера, який обробляє зв'язок з клієнтом, віддаленими файлами та операціями, називається postgres процесом. PostgreSQL обробляє паралельні клієнтські сесії, створюючи нові процеси до кожного підключення[16]. Цей процес створюється та видаляється незалежно від основного. Вона має змогу забезпечувати швидкодію частих запитів до великих активних таблиць за допомогою хешування представлення. PostgreSQL сумісний з ACID та підтримує збережені процедури. Надає можливість створювати захищений зв'язок між клієнтом та сервером, використовуючи SSL[16].

3. ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ ПІДКЛЮЧЕНЬ USB-ПРИСТРОЇВ

3.1. Діаграма варіантів використання

Діаграма варіантів використання відноситься до поведінкових діаграм, та використовуються для опису набора дій, що система повинна або може виконувати у співпраці з одним або декількома зовнішніми користувачами системи[18]. Вона дозволяє визначити вимоги до системи та описати функціональні можливості.

Нижче зображена діаграма use case, що визначає роботу системи моніторингу підключень USB-пристроїв (рис. 3.1).

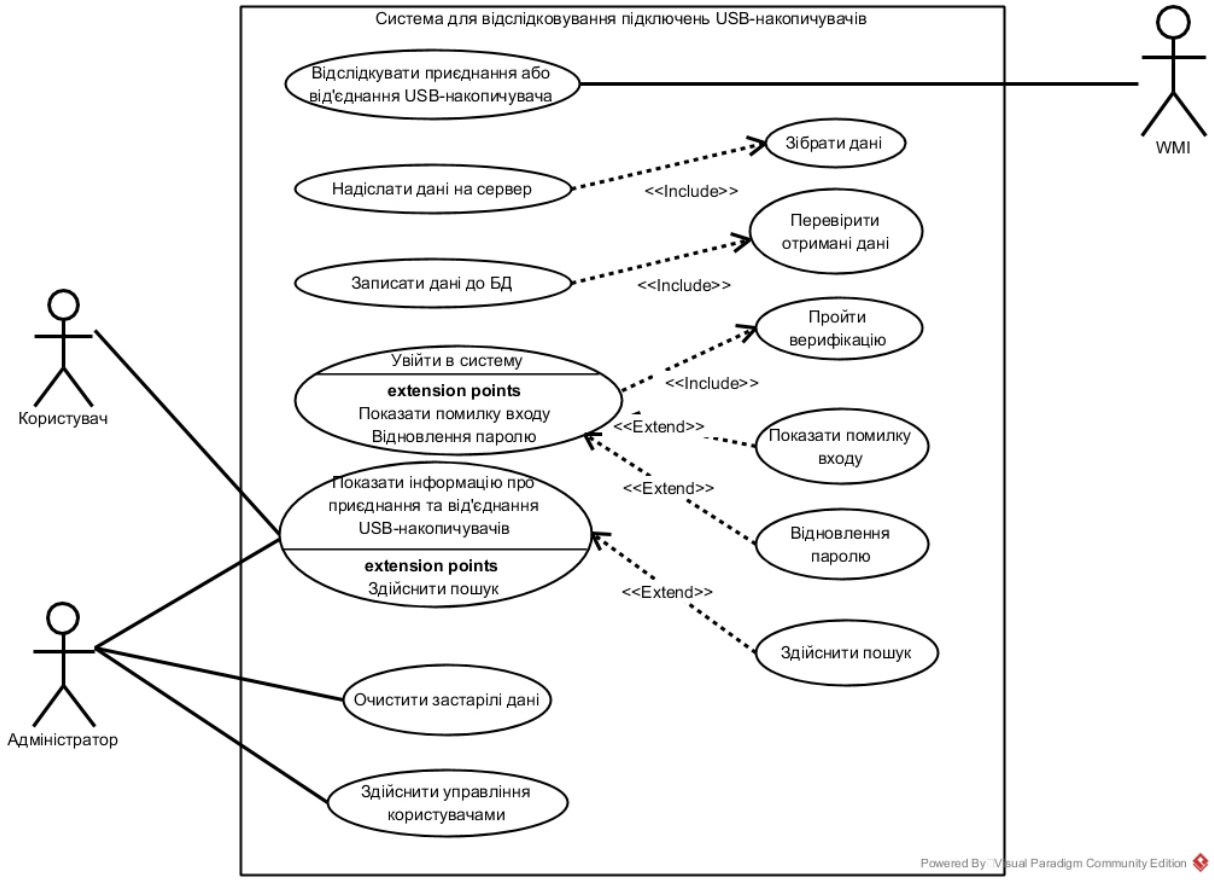


Рисунок 3.1 Діаграма варіантів використання

3.2. Діаграма діяльності

Діаграма діяльності – це одна з видів поведінкових діаграма UML, яка дозволяє моделювати динамічні аспекти розроблюваної системи[19]. Вона дозволяє визначити, як система буде виконувати свої цілі та є чудовим інструментом для відображення логіки системи. Основним поняттям в даній діаграмі – є активність. Діяльність являє собою сукупність дій, що дозволяють представити робочий процес, який призводить до кінцевого результату[19]. Діаграма діяльності акцентує увагу на стані потоку та порядок, в якому він протікає. Потік може бути послідовним, розгалуженим або паралельним, і для роботи з потоками такого типу на діаграмі дій були запропоновані розгалуження, об'єднання і т.д[19].

Графічно діаграма діяльності, зображується за допомогою орієнтованого графа, вершини якого є дії або діяльності, а дугами – переходи між ними. В UML дія – це атомарна операція, що не може бути перерваною, а діяльність – складова операція, з можливістю її перервати[19]. Перехід до наступної дії або діяльності спрацьовує миттєво по їх завершенні.

На рисунку 3.2 зображена діаграма активність, головною метою якої є відслідковування та збереження подій, які виникають при кожному під’єднанні або від’єднанні USB-накопичувачів до комп’ютерів, для подальшого їх аналізу. Дана активність здійснюється за допомогою двох служб. Перша служба встановлюється на комп’ютер над яким буде проводитися моніторинг. Дії, які виконує клієнтська служба, відображаються на доріжці «Служба клієнт». Друга служба встановлюється на сервер. Дії, які виконує серверна служба зображуються на доріжці «Служба сервер».

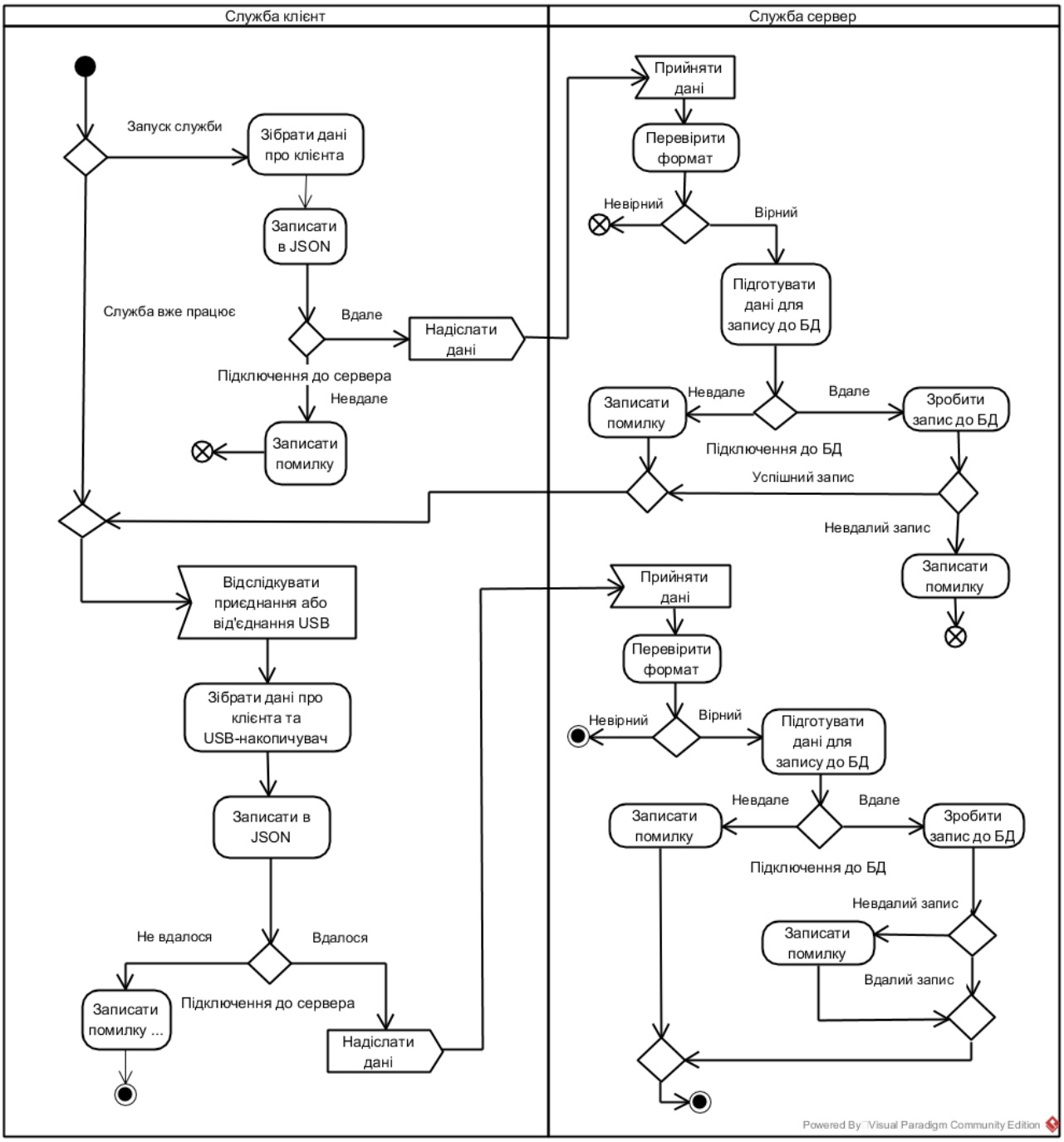


Рисунок 3.2 – Діаграма діяльності. Зображення діяльності служб

На рисунку 3.3 зображена активність веб-додатку, мета якого є здійснення аналізу подій тільки після того, як користувач авторизувався в системі, також надання доступу до функцій адміністрування тільки тим користувачам, які мають відповідні права.

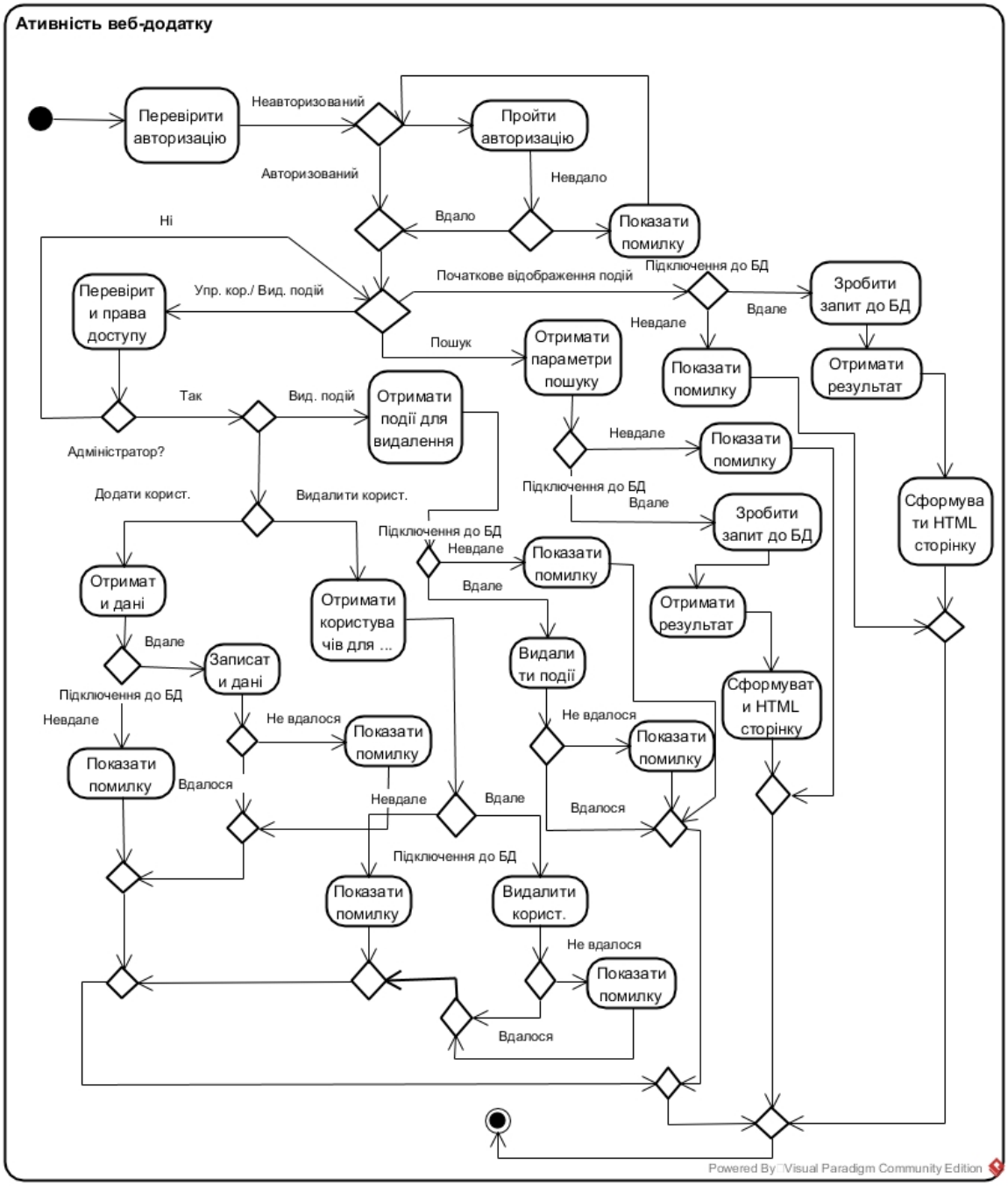


Рисунок 1.3 – Діаграма діяльності. Зображення діяльності веб-додатку

3.3. Діаграма розгортання

Діаграма розгортання застосовується, для відображення топології фізичних компонентів системи у, якій відбувається розміщення програмних компонентів системи. Вона описує статичне представлення розгортання системи.

За допомогою цієї діаграми, можна відобразити розміщення складних інформаційних систем, які потребують різні обчислювальні платформи й технології доступу до баз даних[20]. Це дає можливість раціоналізувати розміщення компонентів у корпоративній мережі, що визначає загальну продуктивність системи. Необхідність інтегрувати системи з інтернетом потребує вирішення супутніх питань забезпечення безпеки та доступність інформації для корпоративних клієнтів. При застосуванні тривимірної технології клієнт-сервер вимагає розміщення великих баз даних у різних сегментах корпоративної мережі, їх архівування, резервного копіювання, хешування, щоби забезпечити продуктивність системи. Ці аспекти теж потребують візуальне представлення, щоб специфікувати програмні та технічні особливості реалізації розподілених архітектур[20].

Звідси можна виділити основні цілі розроблення діаграми розгортання:

- розподілити на фізичних вузлам компоненти системи;
- показати фізичні зв’язки між вузлами під час виконання;
- полегшити процес конфігурації системи.

Основний складник діаграми розгортання є вузли та їхні взаємозв’язки[20]. Представлення вузлів відбувається за допомогою прямокутних паралелепіпедів з артефактами, що розташовуються в них, які зображені у вигляді прямокутників. Вузли можуть містити підузли, представлені прямокутними паралелепіпедами.

Вузол на діаграмі розгортання може переставляти комп’ютер, сервер бази даних, мережеві компоненти та інше[20].

На рисунку 3.4 зображена діаграма розгортання, за допомогою якої було розроблено представлення фізичної топології, на якій розгортається система моніторингу підключень USB-пристроїв. На діаграмі представлено чотири фізичних вузли:

- сервер – вузол, на якому розміщена серверна служба;
- клієнт – вузол над яким здійснюється моніторинг, та на якому розміщена клієнтська служба. Даним вузлом може являтися будь-який корпоративний комп’ютер;
- веб-сервер – вузол, на якому встановлений веб-додаток, головною метою якого є здійснення аналізу подій;
- сервер бази даних – вузол, на якому встановлена база даних.

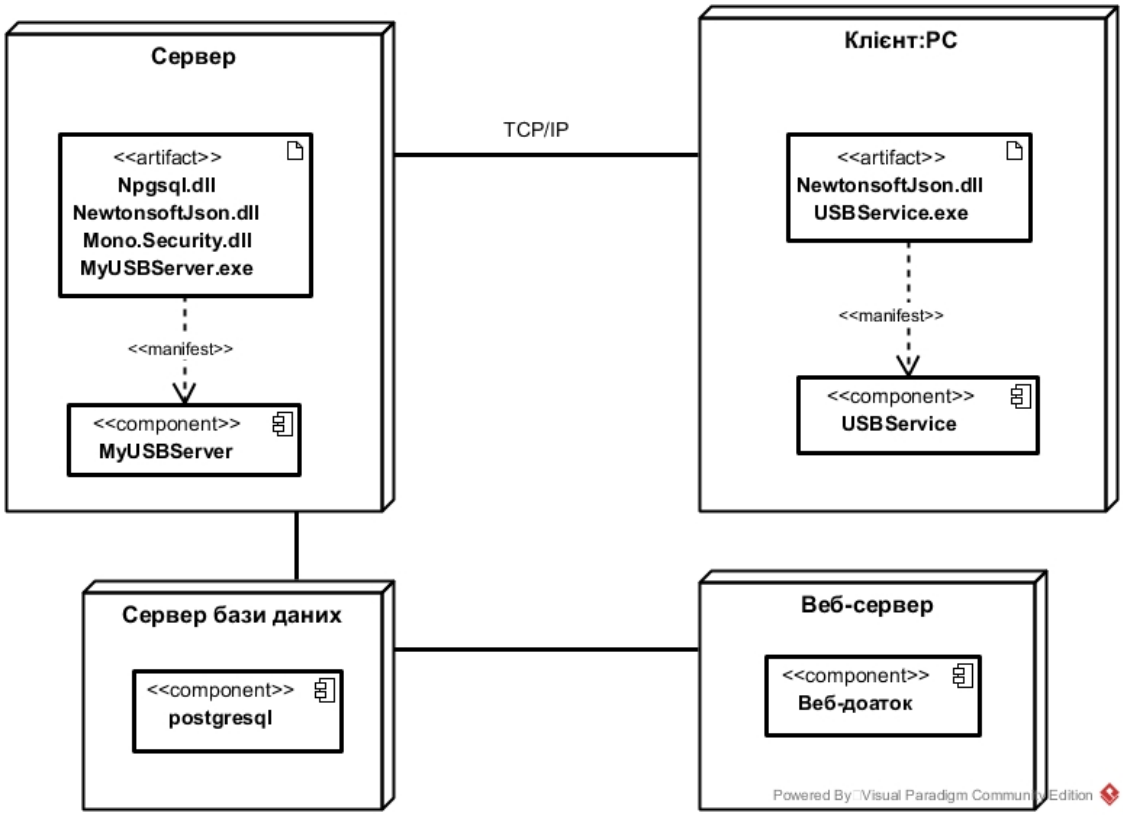


Рисунок 3.4 – Діаграма розгортання

3.4. Діаграма потоків даних

Діаграма потоків даних(DFD) — це один із традиційних способів представити потік даних у системі. Вона показує те, як інформація входить та виходить із системи, процеси, які виконуються в системі, сутності, що взаємодіють з системою та сховища даних[21].

Діаграма потоків даних дозволяє описати систему, використовуючи різні рівні деталізації. Спочатку система зображується в загальному вигляді, з подальшою деталізацію доти поки це буде потрібно[21].

Застосуємо діаграму для проектування потоків даних системи. На рисунку 3.4 система зображена в загальному вигляді, тобто визначний головний процес та зовнішні сутності, які взаємодіють з системою. Цими сутностями є Адміністратор, Користувач, Windows, Адміністратор - це сутність, яка описує користувача системи, що має право доступу до функцій управління системою. «Користувач» - це будь-яка людина, яка зареєстрована в системі та має права тільки для перегляду подій. «Windows» - операційна система, що генерує події, дані яких зчитуються та зберігаються до системи.

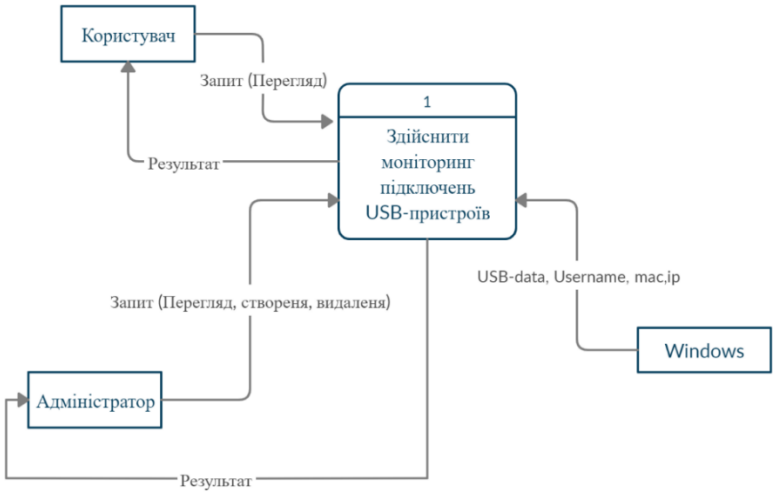


Рисунок 3.4 – Діаграма потоків даних. Рівень 0

На рисунку 3.5 здійснена деталізація нульового рівня. На ній ми можемо побачити детальніший опис процесів та сховища даних з якими вони взаємодіють. На діаграмі виділено два сховища даних, а саме «WEB-user» та «USB/PC - info». Сховище «WEB-user» містить дані, що описують користувача веб-додатку, та використовується для авторизованого доступу та визначення прав користувача. «USB/PC - info» являє собою групу таблиці, що зберігають дані, які були отримані із операційної системи.

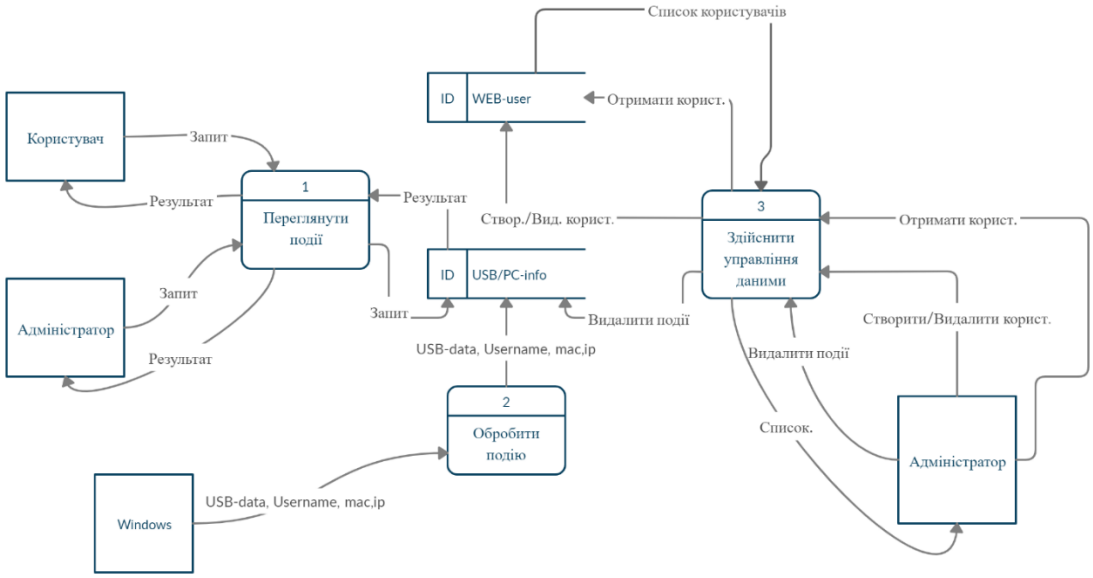


Рисунок 3.5 – Діаграма потоків даних. Рівень 1

3.5. ER-діаграма

Для високорівневого проектування баз даних застосуємо ER-діаграму. Ця модель дозволяє описати предметну область, виділити ключові сутності і визначити зв'язки, які можуть бути встановлені між цими сутностями[22].

На рисунку 3.6 показана побудована діаграма база даних, що задовольняє вимогам розроблювальній системі. Для того, щоб уникнути потенційної суперечності і надмірності даних, під час проектування модель була приведена до третьої нормальної форми.

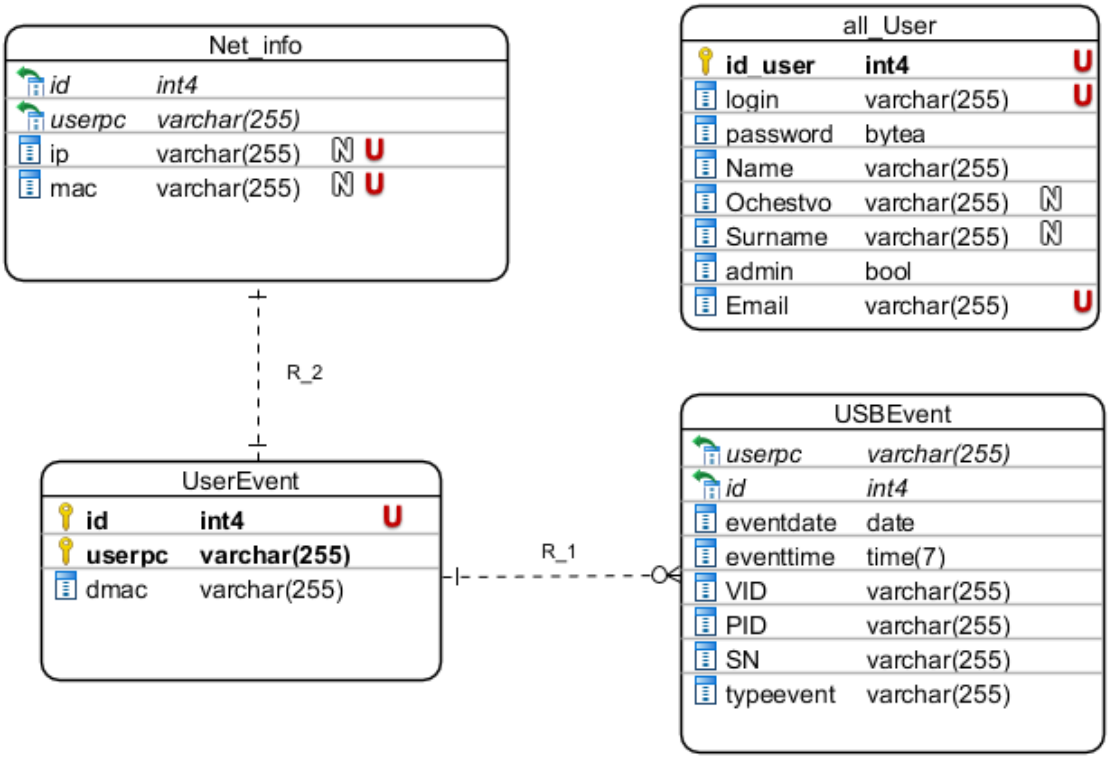


Рисунок 3.6 – ER-діаграма

Опишемо значення атрибутів (табл. 3.6).

Таблиця 3.6 – Опис атрибутів

Таблиця	Поле	Ключі	Зміст
All_User	Id_user	PK	Зберігає номер, що однозначно ідентифікує екземпляр сутності.
	login		Зберігає логін користувача веб- додатку.
	password		Зберігає пароль користувача веб додатку.
	Name		Ім’я користувача веб-додатку.
	Surname		Прізвище користувача веб-додатку.
	Ochestvo		Ім’я по-батькові користувача веб додатку.
	admin		Показує наявність чи відсутність прав ад-міністратора.
	email		Адреса електронної скриньки користувача веб-додатку.
USBEvent	Id	FK	Зберігає номер, що однозначно ідентифікує екземпляр сутності, дозволяє зв’язати таблиці.
	userpc	FK	Ім’я комп’ютера
	eventdate		Дата події
	eventtime		Час події
	vid		Номер виробника USB-накопичувача
	pid		Номер моделі USB-накопичувача
	SN		Номер серійного номера
	typeevent		Тип події (підключення чи відключення)

Продовження таблиці 3.6 – Опис атрибутів

Таблиця	Поле	Ключі	Зміст
Net_info	Id	FK	Зберігає номер, що однозначно ідентифікує екземпляр сутності, дозволяє зв’язати таблиці.
	userpc	FK	Ім’я комп’ютера
	ip		Поточна IP-адреса
	mac		Поточна MAC-адреса
UserEvent	Id	FK	Зберігає номер, що однозначно ідентифікує екземпляр сутності.
	userpc	FK	Ім’я комп’ютера
	dmac		Мак-адреса ethernet карти

4. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ПІДКЛЮЧЕНЬ USB-ПРИСТРОЇВ ДЛЯ ПРОВЕДЕННЯ АУДИТУ З КІБЕРБЕЗПЕКИ

4.1. Розробка клієнтської служби

В першу чергу, був реалізований головний функціонал, а саме здатність відслідковувати приєднання або від'єднання USB-накопичувача. Після детального аналізу способів вирішення поставленої задачі, було обрано, вбудований в операційну систему Windows, інструмент WMI. WMI здатний відслідковувати події, які виникають під час змін у внутрішніх об'єктах[3]. В нашому випадку будемо відслідковувати зміни у об'єкті Win32_USBHub. Під час кожного підключення або відключення USB-пристрою параметри зазнають змін. Щоб відслідкувати зміни, WMI має клас ManagementEventWatcher[6].

Визначимо поетапні дії для початку відслідковування змін:

1. Отримаємо об'єкт Win32_USBHub, для цього використаємо WMI Query Language.

```
WqlEventQuery queryToGetUSBObj = new WqlEventQuery(
        "SELECT * FROM __InstanceOperationEvent WITHIN 10 WHERE TargetInstance ISA
'Win32_USBHub'");
```

2. Створимо об'єкт класа ManagementEventWatcher, який буде виконувати функції спостереження, для цього передамо вище створений запит, до конструктору;

```
USBEventWatcher = new ManagementEventWatcher(queryToGetUSBObj);
```

3. Зареєструємо функцію, яка буде обробляти події;

```
USBEventWatcher.EventArrived += HandleEventUSB;
```

4. Почнімо процес відстеження.

```
USBEventWatcher.Start();
```

За замовчуванням, служба має такі мережеві налаштування, для відправки повідомлення на сервер:

- IP-адреса – 127.0.0.1;
- Порт – 8888.

Для надсилання повідомлень на сервер, який знаходиться на іншому комп’ютері, було створено утиліту налаштування мережевих параметрів. Після реєстрації служби в системі, в реєстрі створюється ключ, який описує цю службу. Windows реєстр являє собою базу даних, де кожен додаток може зберігати параметри для своєї роботи.

Для налаштування служби утиліта робить запис до реєстру, та потім перезавантажує її. Після, чого дані будуть відсилатися, вже за новим мережевим адресом.

Якщо, під час виконання сталося помилка, то на цей випадок програма робить запис до журналу Windows подій. В коді наведеному нижче створюється джерело подій, якщо воно досі не існує.

```
eventLog1 = new System.Diagnostics.EventLog();

        if (!System.Diagnostics.EventLog.SourceExists("EventUSBService"))
        {
           System.Diagnostics.EventLog.CreateEventSource(
              "EventUSBService", "USBServiceLog");
        }
        eventLog1.Source = "EventUSBService";
        eventLog1.Log = "USBServiceLog";
```

Нижче наведений загальний вигляд опрацювання помилок.

```
try
{. . . }
catch(Exception ex)
{
              eventLog1.WriteEntry("Якесь повідомлення" + ex, EventLogEntryType.Error, 10);}
```

Збереження помилок до Windows Event Log, є гарним та зручним способом для виявлення та виправлення збоїв в роботі служби, а також, відстеження події, які через помилку не потрапили до бази даних.

В коді нижче виконується зчитування параметрів з реєстру відразу після запуску служби, уразі відсутності попереднього налаштування, мережеві параметри залишаються за замовчуванням.

```
RegistryKey localMachineKey = Registry.LocalMachine;
```

```
        RegistryKey USBMonService =
localMachineKey.OpenSubKey(@"SYSTEM\CurrentControlSet\Services\MyUSBService", true);
    string ImagePath = USBMonService.GetValue("ImagePath").ToString();
    string[] SplitMainPath = ImagePath.Split(new char[] { '^' });
    if (SplitMainPath.Length > 1)
        { присвоєння отриманих значень...  }
```

При кожній обробці подій, службі потрібно дізнатися такі дані, як PID, VID, серійний номер USB-пристрою, а також поточну IP-адресу та мак-адресу, мак-адресу ethernet карти та ім’я комп’ютера.

Вся інформація про подію міститься в об’єкті EventArrivedEventArgse, який є параметром функції HandleEventUSB. Нижче показаний код, на якому ми отримуємо цільовий об’єкт, у нашому випадку це Win32_USBHub, який містить всю важливу інформацію про під’єднаний USB-пристрій.

```
        ManagementBaseObject targetObj =
(ManagementBaseObject)e.NewEvent.Properties["TargetInstance"].Value;
```

Інформацію про PID, VID, серійний номер, містить поле PNPDeviceID. Нажаль, ці дані не знаходяться в окремих полях, а зберігаються в рядку який має вже відомий вигляд. Для отримання з рядка лише потрібної інформації був розроблений регулярний вираз, яркий ви можете побачити нижче.

```
private string pattern =
@"(?<VID>(?<=VID_).*(?=&))|(?<PID>(?<=PID_).*[^\\](?=\\))|(?<SN>(?<=PID_.*\\)[^\\].*)";
```

Далі використаємо регулярний вираз, для виокремлення PID, VID та серійний номер з рядка.

```
MatchCollection PidVidNumber = regex.Matches(Convert.ToString(targetObj["PNPDeviceID"]));
```

Після отримання даних про USB-пристрій виникає необхідність конкретизувати комп’ютер, на якому сталася подія. Щоб визначити локальну поточну IP-адресу, використаємо безкоштовний DNS-сервер. Суть даного способу доволі проста. Ми здійснюємо підключення до Google Public DNS, та відстежуючи траєкторію IP пакету, можемо чітко визначити IP-адрес остатнього вузла.

```
using (Socket SetNewSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, 0))
    {
    SetNewSocket.Connect("8.8.8.8", 65530);
    IPEndPoint IpendPoint = SetNewSocket.LocalEndPoint as IPEndPoint;
    return IpendPoint.Address;
                }
```

Використаємо отриману IP-адресу для встановлення мак-адреси мережевої карти, через яку відбувається інтернет-підключення. На сучасних пристроях, як правило встановлюють дві мережеві карти. Одна з яких відповідає за зв'язок по бездротовій мережі WI-Fi, а інша створює інтерфейс для ethernet підключення. Виникає необхідність визначення мережевої карти, через яку відбувається підключення до Інтернету. Для вирішення проблеми використаємо функцію SendARP бібліотеки iphlpapi.dll. Вона відправляє ARP запит, щоб отримати фізичну адресу, яка відповідає вказаній IPv4-адресі.

```
private string GetMacUseingIP(IPAddress ip)
    {   byte[] macAddrByIP = new byte[6];
        int length = macAddrByIP.Length;
        SendARP(ip.GetHashCode(), 0, macAddrByIP, ref length);
        return BitConverter.ToString(macAddrByIP, 0, 6).Replace("-",":"); }
    [DllImport("iphlpapi.dll", ExactSpelling = true)]
    public static extern int SendARP(int DeIP, int SoIP, [Out] byte[] pcMACAddr, ref int pPhyAddrLen);
```

В якості унікального ідентифікатора було обрано фізичну адресу ethernet. Вона дозволить однозначно розрізняти один пристрій від іншого.

```
IPGlobalProperties objPCProp = IPGlobalProperties.GetIPGlobalProperties();
NetworkInterface[] NetIntf = NetworkInterface.GetAllNetworkInterfaces();
PhysicalAddress macAddress = NetIntf[0].GetPhysicalAddress();
```

Всі отримані дані, записуються до словника SaveData.

```
Dictionary<string, string> SaveData = new Dictionary<string, string>();
```

Перед тим, як відправити зібрану інформацію на сервер, переведемо SaveData в строку JSON формату. Для роботи з JSON, використаємо Json.NET.

```
USBDataAndHostInfo = collectData.CollectData();
string output = JsonConvert.SerializeObject(USBDataAndHostInfo);
```

Для зв’язку між сервером і клієнтом обрано TCP-протокол, який гарантує доставку повідомлення.

```
TCPConnection.SendData(Convert.ToString(output), ip, port, eventLog1);
```

4.2. Розробка серверної служби

Завершивши реалізацію клієнтської служби, приступимо до серверної служби. Головне питання, яке потрібно вирішити під час розробки, є організація процесу прийому повідомлень від клієнтів та запис їх до бази даних. Реалізація обробки помилок, та отримання параметрів для запуску відбувається так само, як і для клієнтської служби.

Параметри, які потребує служба для своєї роботи:

- port – порт, який відкриває служба для прослуховування. За замовчуванням програма відкриває 8888;
- Username – ім’я користувача бази даних, до якої буде збережено повідомлення;
- Password – пароль, щоб підключитися до бази даних;
- DB – ім’я бази даних;
- Host – інтернет адреса, за якою СУБД.

Параметри можна налаштувати за допомогою утиліти. Принцип збереження параметрів залишається таким самим, як і для клієнтської служби.

Перейдемо до вирішення задачі прийому повідомлень від користувачів. Для того, щоб отримати дані від клієнта, в першу чергу службі треба відкрити порт для прослуховування. Для того, щоб відкрити порт, в мові програмування C# існує два способи. Ми можемо відкрити порт за допомогою TcpListener або Socket. Socket є більш гнучкий клас для роботи з мережею, але зважаючи на той факт, що наша служба мусить використовувати для свої роботи TCP-протокол, то вибір був здійснений в бік класу TcpListener. Для коректної роботи служби прослуховування потрібно здійснювати в окремому потоці, інакше службу не вийде зареєструвати в системі, через роботу циклу While. Для роботи з потоками застосуємо клас Thread.

Нижче наведений код, який розпочинає прослуховування порту.

```
public void Start() {
```

```
listener = TcpListener.Create(port);
listener.Start();
myListenerThread = new Thread(new ThreadStart(this.listenerProcessAsync));
myListenerThread.Start();}
```

Розглянимо те, як відбувається процес прослуховування. В першу чергу запускаємо функцію TcpListener.AcceptTcpClientAsync. В нашому випадку використовується асинхронна функція. Вибір саме асинхронної функції, зумовлений тим, що це дасть змогу коректною зупинити роботу служби, інакше служба не зможе добровільно припинити свою роботу, доки від клієнта не прийде повідомлення. Для припинення виконання асинхронної функції застосовується CancellationTokenSource. Token буде повертати маркер відміни, при виклику методу Cancel джерела. Щоб зупинити цикл While, використовується змінна _Alive.

Після того як, клієнт робить спробу підключитися до серверу, метод TcpListener.AcceptTcpClientAsync повертає об’єкт класу TcpClient, який містить в собі підключення з клієнтом. Оброблювати підключення з клієнтом здійснюється в окремих потоках, для цього створений клас ClientObj, в якому відбувається отримання повідомлення та подальша його обробка. Код вище сказаного продемонстрований нижче.

```
cancellationToken = new CancellationTokenSource();
        try
        {
           while (_Alive)
           {
TcpClient client = await Task.Run(() => listener.AcceptTcpClientAsync(), cancellationToken.Token);
ClientObj clientObjWorker = new ClientObj(client, eventLog1, Username, Host, Password, DB);
Thread clientThreadWorker = new Thread(new ThreadStart(clientObjWorker.Process));
clientThreadWorker.Start();}
        }
        catch (Exception ex)
        {
    eventLog1.WriteEntry("Виникла помилка при прослуховуванні порту : " + ex, EventLogEntryType.Error, 12);
           if (listener != null)
              listener.Stop();
        }
```

Поглянимо на сому реалізацію отримання повідомлення та запис його до бази даних. Для отримання потоку NetworkStream, який використовується для прийому або відправлення повідомлення використаємо метод TcpClient.GetStream. Повідомлення зчитується в масив типу byte. В коді нижче

отримується потік, з якого потім відбувається зчитування повідомлення від клієнту.

```
streamUSB = clientUSB.GetStream();
byte[] dataUSB = new byte[64];
StringBuilder builderUSB = new StringBuilder();
int bytes = 0;
do{
  bytes = streamUSB.Read(dataUSB, 0, dataUSB.Length);
  builderUSB.Append(Encoding.Unicode.GetString(dataUSB, 0, bytes));
}while (streamUSB.DataAvailable);
```

Після повного зчитування повідомлення з потоку, перевіримо, чи відповідає воно формату JSON з тими змістом, який повинен нам відправляти клієнт. Для роботи з форматом JSON використовується Json.NET. Перш ніж зробити перевірку, потрібно задати загальний вигляд повідомлень, які можуть надходити від клієнта.

```
string schemaJson1 = @"{ 'date' : 'string', 'time':'string',
                  'USBeventType':'string', 'VID':'string',
                  'PID':'string', 'SN':'string',
                  'macAddress':'string','hostName':'string',
                  'ip':'string', 'mac': 'string', 'massageType': 'string'}";

string schemaJson2 = @"{
                'macAddress':'string', 'hostName':'string',
                'ip':'string','mac':'string', 'massageType': 'string'}";
```

Переведемо вище задані шаблони в об’єкти типу JsonSchema, який далі приймає участь у перевірці.

```
JsonSchema schema1 = JsonSchema.Parse(schemaJson1);
JsonSchema schema2 = JsonSchema.Parse(schemaJson2);
```

Далі переведемо отримане повідомлення в об’єкт класу JObject, це дозволить працювати з повідомленням, як зі звичайним словником. Перед тим, як почати запис даних до БД, перевіримо отриманий об’єкт на відповідність одному із шаблонів об’явленому вище.

```
JsonSchema schema1 = JsonSchema.Parse(schemaJson1);
JsonSchema schema2 = JsonSchema.Parse(schemaJson2);
JObject usbEventInfo = JObject.Parse(message);
bool valid1 = usbEventInfo.IsValid(schema1);
bool valid2 = usbEventInfo.IsValid(schema2);
if (valid1 || valid2)
{ //Запим до БД }
```

Якщо повідомлення не пройшло перевірку, то вважається, що воно надійшло від якоїсь сторонньої програми та просто відкидається.

Завершивши перевірку, перейдемо до збереження даних до БД. Для взаємодії з базою даною PostgreSQL була використана бібліотека Npgsql.

До початку будь-яких дії з базою даною, потрібно встановити з нею підключення. Щоб встановити підключення треба задати пароль, ім’я користувача, інтернет адрес, за яким розташовується сервер бази даних та назву бази даних. Для встановлення зв’язку використаємо конструктор класу NpgsqlConnection, передавши в нього потрібні параметри підключення. Потім із створеного об’єкта викличемо метод Open().

```
var cs = String.Format("Host={0};Username={1};Password={2};Database={3}", Host, Username, Password, DB);
MyDBCon = new NpgsqlConnection(cs); MyDBCon.Open();
```

Після встановлення зв’язку з базою даною, можна розпочати взаємодію з нею. Однією із задач сервера – це збереження інформації. Сервер отримує від клієнта два типи повідомлень. Одне повідомлення надсилається після запуску служби, щоб об’явити новий комп’ютер над, яким буде проводитися моніторинг, інше надсилається після виникнення події при підключенні або відключені USB-пристрою.

Для уникнення можливості SQL-ін’єкцій, було використано підготовлені запити. Замість прямого підставлення значень ми використовуємо псевдо змінні. Загальний алгоритм здійснення запитів виглядає так:

1. Визначаємо запит var sql = “ якийсь запит … (:зміна1, :зміна2 …) ”;
2. Далі створюємо об’єкт типу NpgsqlCommand передавши в конструктор запит та об’єкт з підключенням. var USBcmd = new NpgsqlCommand(sql, con);
3. Встановимо значення псевдо зміним. Для цього використаємо метод USBcmd.Parameters.AddWithValue(“зміна1”, значення);
4. Виконаємо запит. USBcmd.ExecuteNonQuery();
5. Закриємо з’єднання.

4.3. Розробка веб-додатку

За допомогою веб-додатку реалізується можливість зручно переглядати та аналізувати події, які відбулися на корпоративних комп'ютерах. Бекенд додатку базується на мікрофреймворці Flask. Він призначений для швидкого та легкого початку роботи з можливістю масштабування до складних додатків[23]. Мікрофреймворк Flask не вимагає від розробника притримуватися чіткій структурі, як наприклад фреймворк Django, що дозволяє більш гнучко будувати каркас веб-додатку.

Початок роботи додатку починається з файлу __init__.py. Він створює об'єкт додатку, як екземпляр класу Flask. Далі задає параметри налаштування та імпортує модуль в якому розташована логіка додатку.

Важливою частиною додатку є декоратор @app.route('/path'). Він реєструє функцію для обробки URL-адреси. Зареєстрована функція буде обробляти прив'язану до неї URL-адресу та передавати значення назад до браузера. Інтерфейс веб-додатку реалізується за допомогою шаблонів. Шаблони – це файли, які містять статичні дані, а також заповнювачі для відображення динамічних даних. Для візуалізації шаблонів Flask використовує бібліотеку шаблонів Jinja[23]. Писати кожний раз повну структуру HTML було б дуже громіздким рішенням. Щоб уникнути громіздкості, кожен шаблон розширює базовий шаблон та перевизначає конкретні розділи. Давайте поглянемо на загальний принцип. В базовому шаблоні вставляється блок {% block name %} {% endblock %} на місце, що буде розширюватися. Тепер замість повторення коду, що знаходиться базовому шаблоні, потрібно просто розширити його.

```
{% extends "base.html" %}
 {% block name  %}
                <div>
                </div>
{% endblock %}
```

Динамічні компоненти шаблонів розміщуються в фігурних дужках {{ .. }}. Оператори управління, такі як цикли або оператор if, розміщується всередині {% … %}.

Для візуалізації шаблонів використовується функція render_template(). Функція для своєї роботи потребує ім’я шаблону та параметри значення яких використовуються для створення динамічного контенту. Під час кожного звернення до цієї функції, Flask буде шукати шаблони в паці templates.

Щоб отримати дані для аналізу, потрібно організувати роботу за базою даною. Для цього використовується модуль psycopg2, який призначений для взаємодії з PostgreSQL. Принцип підключення та взаємодії з СУБД збігається з тим, який ми використовували при реалізації серверної служби.

Форми є важливою частиною кожного веб-додатку. Flask містить в своєму арсеналі чудовий модуль для створення форм flask_wtf. Кожна форма, яка знаходиться на сайті, представляться у вигляді об’єктів. Здійснення валідації форм відбувається за допомогою функції Required, яка прикріплюєтеся до поля. Нижче загальний вигляд застосування форм flask_wtf.

Спочатку описуються компоненти, які повинна містити форма.

```
class HostAndMac(FlaskForm):
   hostname = StringField("Ім'я юзера")
   mac      = StringField('Mac адреса')
   submit   = SubmitField('Знайти')
```

Далі у функції, яка обробляє URL-адресу, створюється об’єкт форми HostAndMac.

```
@app.route('/home', methods=['GET', 'POST'])
@login_required
def showuserevent():
   ...
   form = HostAndMac()
   ...
   return render_template( 'showuserevent.html',form=form ...)
```

Тепер в шаблон підставимо об’єкт форми.

```
{% extends "menu.html" %}
{% block contant%}
         ...
         ...
   <form method="POST" action="/">
```

```
    {{ form.hidden_tag() }}
        {{ form.hostname() }}
    {{ form.mac()}}
    <button class="button1 btn btn-width" type="submit"> Знайти </button>
  </form>
        ...
        ...
</div>
{% endblock %}
```

Дані з форми пересилаються на сервер за допомого GET або POST методу. Для того, щоб відслідкувати метод GET або POST, використовується глобальний об’єкт request, який дозволяє отримати доступ до даних запиту. За допомогою атрибуту request.method, можна побачити метод запиту. Це дозволить визначити момент, коли потрібно отримувати дані. Також модуль flask_wtf містить метод validate_on_submit(), який повертає true, коли користувач натиснув на кнопку submit. Використовуючи той чи інший спосіб, призведе до однакового результату. Нижче показаний уривок коду, який отримує дані із запиту двома різними способами.

```
...
formAdd = RegisterForm()
    if formAdd.validate_on_submit():
        username = formAdd.username.data
        surname  = formAdd.surname.data
        name     = formAdd.name.data
...
if request.method == 'POST':
        id = request.form['idpage']
```

Так, як додаток містить дані з обмеженим доступом, то виникає необхідність реалізація авторизованого входу. Щоб допустити користувача до перегляду, йому потрібно ввести свій логін та пароль. Після вдалої аутентифікації користувача з ним встановлюються сесія. Встановлення сесії дозволяє не проходити авторизацію при переходах між сторінками. Сесії зберігаються в тимчасовому катологі на сервері, доступ до яких відбувається через session. Уривок коду нижче виконує верифікацію користувача, та після успішного завершення встановлює з ним сесію.

```
form =   LoginForm()
  try:
```

```
    if form.validate_on_submit():
        user = form.username.data
        password = '\\x' + hashlib.sha256(form.password.data.encode('utf-8')).hexdigest()
        print(password)
        userDict = db.CheckUser(user, password)
        if(len(userDict) == 1):
            userDict = userDict[0]
            if(userDict[0] == user):
                if(userDict[1] == password):
                    session['logged_in'] = True
                    session['username']  = user
                    session['surname'] = userDict[2]
                    session['name']    = userDict[3]
                    session['admin']   = userDict[4]
                    return redirect('/')
```

Користувача в системі реєструє адміністратор. Детальна реалізація функції створення нових користувачів та видалення знаходиться в додатку.

Нижче наведена код, який виконує вихід з додатку.

```
@app.route('/logout')
def logout():
    session['logged_in'] = False
    session.pop('logged_in', None)
    session.pop('username',  None)
    session.pop('surname',   None)
    session.pop('name',      None)
    session.pop('admin',     None)
    return redirect('/login')
```

Для перевірки, чи авторизований користувач та його права доступу було створено два декоратори. Декоратор – це функція, яка розширює функціонал іншої функції без зміни її коду. В нашому випадку декоратор буде перевіряти сесію для визначення, чи авторизований користувач та чи має він необхідні права. Якщо перевірка пройшла успішно, то користувач отримує доступ до бажаної сторінки, інакше додаток відкриє сторінку для входу в систему. Далі наведений код цих декораторів.

```
def login_req_user(f):
    @wraps(f)
    def decoratedFunctionCheck(*args, **kwargs):
        if not session.get('logged_in'):
            return  redirect(url_for('.login'))
        return f(*args, **kwargs)
    return decoratedFunctionCheck

def login_req_admin(f):
    @wraps(f)
```

```
def decoratedFunctionCheck(*args, **kwargs):
    if not (session.get('logged_in') and session.get('admin')):
        return  redirect(url_for('.login'))
    return f(*args, **kwargs)
return decoratedFunctionCheck
```

Окрім можливості авторизованого входу, додатку треба вміти відновлювати пароль користувача. Для вирішення цієї задачі був використаний модуль flask_mail. Цей модуль дає можливість відсилати користувачам повідомлення на електрону скриньку. Для відсилання повідомлень було використано почтовий сервіс «Укр.нет». Для взаємодії з програмами почтовий сервіс «Укр. нет» дає змогу налаштувати IMAP. Після активування доступу до зовнішніх програм, веб-додаток зможе використовувати почтову скриньку для розсилання повідомлень.

Нижче наведений список налаштувань, які потрібно задати в файлі __init.py__ для зв’язку з почтовим сервером.

```
app.config['MAIL_SERVER'] = 'smtp.ukr.net'
app.config['MAIL_PORT']  = 465
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
app.config['MAIL_USERNAME'] = 'testboss@ukr.net'
app.config['MAIL_PASSWORD'] = '****************'.
```

Відновлення паролю відбувається за таким принципом. Користувач вводить поштову адресу та відправляє на веб-сервер. Веб-сервер робить пошук, якщо така адреса існує, то розпочинається формування повідомлення. Вершу чергу генерується унікальний токен за допомогою, якого буде згенероване посилання, що дозволить відновити пароль. Для генерування токену застосовується клас URLSafeTimedSerializer. Перше створюється об’єкт за допомого конструктора URLSafeTimedSerializer передавши в нього секретний ключ. Далі за допомогою методу dumps створюється токен, який містить зашифрований Email. Щоб заповнити повідомлення змістом, створимо об’єкт, використавши конструктор Message. В якості параметрів конструктор приймає назву теми повідомлення, почтовий сервер для відправлення повідомлення та валідатор. Посилання генерується за допомого функції url_for() та

прикріплюється до змісту повідомлення. Далі відбувається відправлення. Уривок коду нижче містить розглянутий вище алгоритм.

```
message = "Лист для відновлення паролю надіслано!"
token1 = se.dumps(email, salt='email-confirm')
EmailMsg = Message('Відновлення паролю!', sender = 'testboss@ukr.net', recipients = [email])
Link = url_for('confirm_email', token=token1, _external=True)
EmailMsg.body = 'Посилання для відновленя паролю: {}'.format(Link)
UkrMail = Mail()
UkrMail.send(EmailMsg)
```

Зміна повідомлення відбувається після того, як користувач перейшов за посиланням, створив новий пароль та відправив на сервер. Сервер дозволяє встановити новий пароль, тільки після вдалого отримання Email з токену. Уривок з кодом знаходиться нижче.

```
form = ChangePassword()
  try:
     UserEmail = se.loads(token ,salt='email-confirm', max_age=3600)
  except :
     return '<h1>Не вдалося відновити повідомлення!</h1>'
  if form.validate_on_submit():
    password = form.password.data
    try:
        db = SQLdb()
    except:
        return "Error404"
```

4.4. Перевірка працездатності системи

Для перевірки працездатності системи визначимо план перевірки (табл. 4.1).

Таблиця 4.1 – План перевірки

TC_ID	Вимоги перевірки
TC1	Перевірити спроможність реєструвати службу клієнта та службу сервер в системі за допомогою інсталятора.
TC2	Перевірити спроможність відслідковувати приєднання або від'єднання USB-накопичувачів до ПК, відсилати отримані дані на сервер та зберігати їх до бази даних.
TC3	Перевірити спроможність налаштовувати мережеві параметри, для служби клієнт та служби сервер. Здатність працювати служб на окремих комп'ютерах та можливість переглядати отримані події за допомогою веб-додатку.
TC4	Перевірити здатність авторизованого доступу в систему.
TC5	Перевірити здатність відновити пароль користувача.
TC6	Перевірити надання розмежованого доступу до функціоналу.
TC7	Перевірити здатність видаляти події та вузли.
TC8	Перевірити спроможність здійснювати пошук.
TC9	Перевірити спроможність створювати та видаляти користувачів.

Перевірка TC1

Сценарій перевірки

1. Запускаємо по черзі інсталятори MyUSBServerSetup.msi та MyUSBServiceSetup.msi (рис. 4.1).

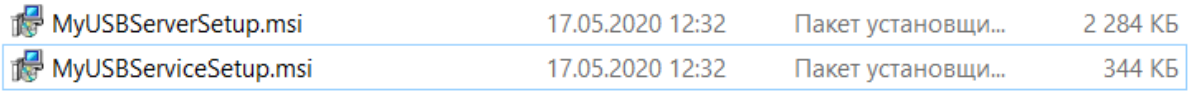


Рисунок 4.1 – Інсталятори служб

Результат

За допомогою переглядача служб знайдено зареєстровані служби (рис. 4.2)

.

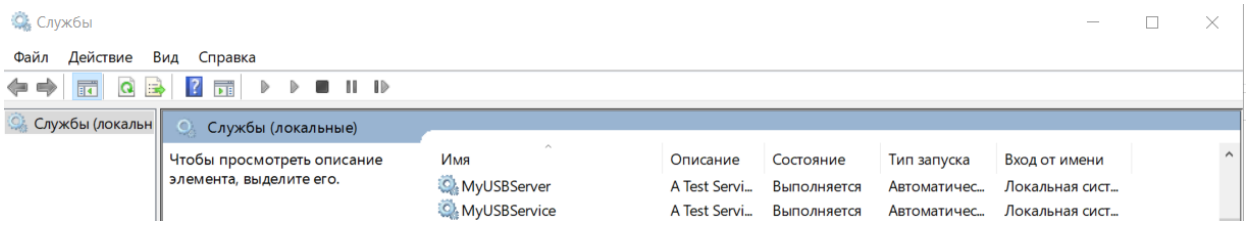


Рисунок 4.2 – Встановлені служби

Перевірка журналів подій показала, що служби були успішно запущені (рис. 4.3 – 4.4).

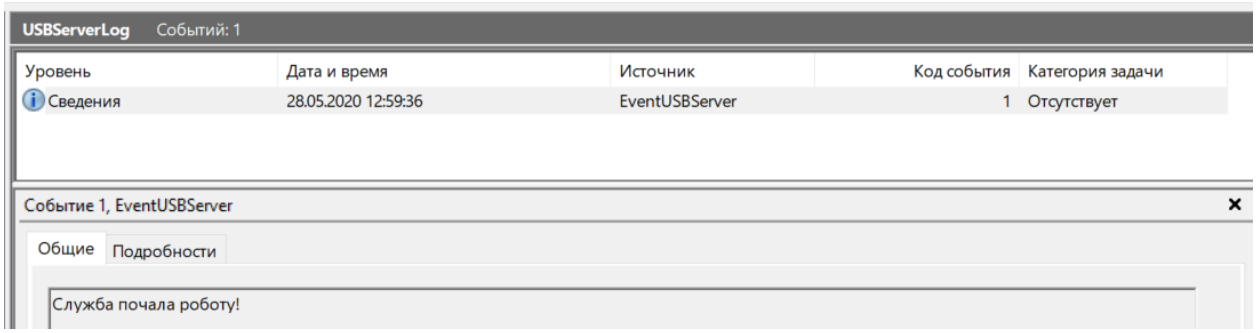


Рисунок 4.3 – Журнал логів серверної служби

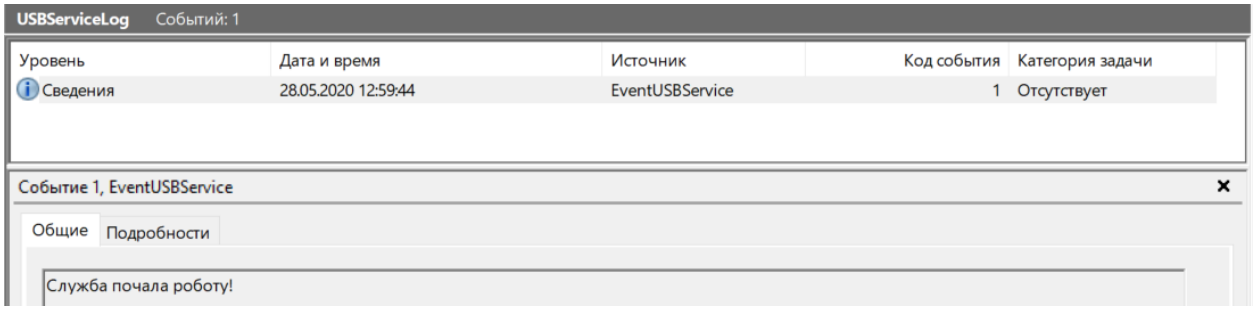


Рисунок 4.4 – Журнал логів клієнтської служби

Перевірка ТС2

Сценарій перевірки

1. Під’єднаємо USB-накопичувач до комп’ютера.
2. В pgAdmin4 вводимо SQL-запит для перевірки надходження події.

select * from usbevent order by eventdate + eventtime DESC.

3. Від’єднаємо USB-накопичувач від комп’ютера.
4. Повторимо дії з пункту 2.

Результат

1. Після під’єднання USB-пристрою дані успішно додані до бази даних. Це означає, що подія під’єднання USB-пристрою була успішно відслідкована, передана на сервер та записана до бази даних (рис. 4.5).

	userpc character varying (255)	id integer	eventdate date	eventtime time without time zone	vid character varying (255)	pid character varying (255)	sn character varying (255)	typeevent character varying (255)
1	LAPTOP-TN2QEGPO	3	2020-05-28	13:34:47	0951	1666	408D5C86C7F9E2618955247E	connected

Рисунок 4.5 – Створений запис в базі даних

2. Після від’єднання USB-пристрою результат збігається з пунктом 1 (рис. 4.6).

	userpc character varying (255)	id integer	eventdate date	eventtime time without time zone	vid character varying (255)	pid character varying (255)	sn character varying (255)	typeevent character varying (255)
1	LAPTOP-TN2QEGPO	3	2020-05-28	14:05:58	0951	1666	408D5C86C7F9E2618955247E	disconnected

Рисунок 4.6 – Створений запис в базі даних

Перевірка ТС3

Сценарій перевірки

1. Встановимо службу клієнт на інший комп’ютер «Big-PC», який знаходиться в локальній мережі.
2. Запустимо утиліту для налаштування служби клієнт та заповнимо поля (рис. 4.7).

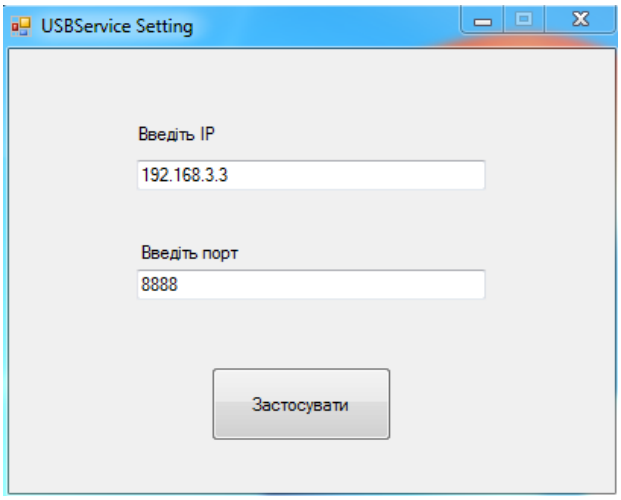


Рисунок 4.7 – Налаштування мережевих параметрів клієнтської служби

3. Натиснемо кнопку «Застосувати» (рис. 4.8).

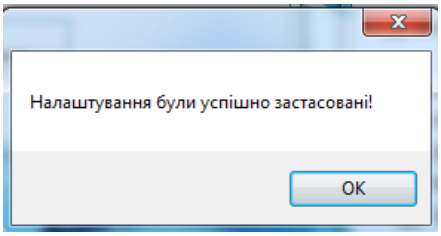


Рисунок 4.8 – Повідомлення про успішне налаштування

4. Запустимо утиліту для налаштування серверної служби та заповнимо поля (рис. 4.9).

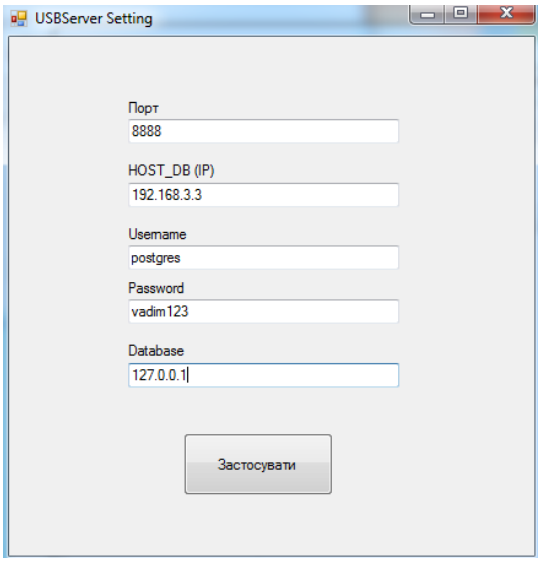


Рисунок 4.9 – Налаштування мережевих параметрів серверної служби

5. Натиснемо кнопку «Застосувати» (рис. 4.10).

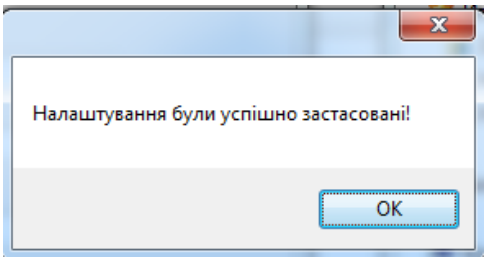


Рисунок 4.10 – Повідомлення про успішне налаштування

6. Під’єднаємо та від’єднаємо USB-пристрій на комп’ютері «Big-PC».

7. Відкриваєм головну сторінку додатку.

8. Переходимо на детальний перегляд подій, які відбулися на комп’ютері «Big-PC».

Результат

Налаштування клієнтської та серверної служби пройшли успішно. Служба клієнт коректно відсилає повідомлення на сервер по мережі. Веб-додаток відображає всю потрібну інформацію.

1. На першому скріншоті відображаються список комп’ютери, які об’єктом моніторингу, їхній IP та мак адреса, загальна кількість подій, остання дата події, останній час події (рис. 4.11).

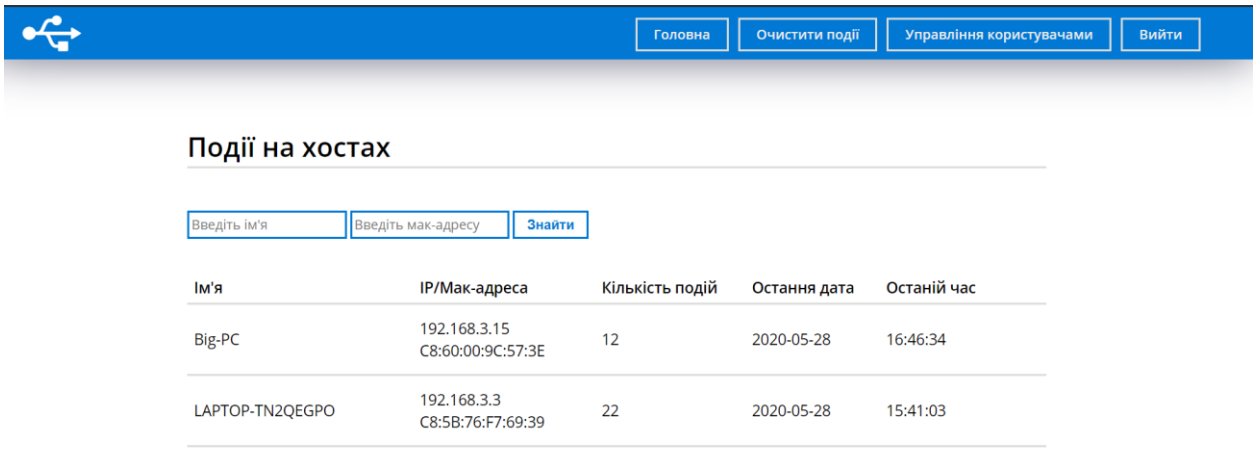


Рисунок 4.11 – Головна сторінка веб-додатку

2. На другому скріншоті відображається повний список подій, які відбулися на комп’ютері «Big-PC» (рис. 4.12).

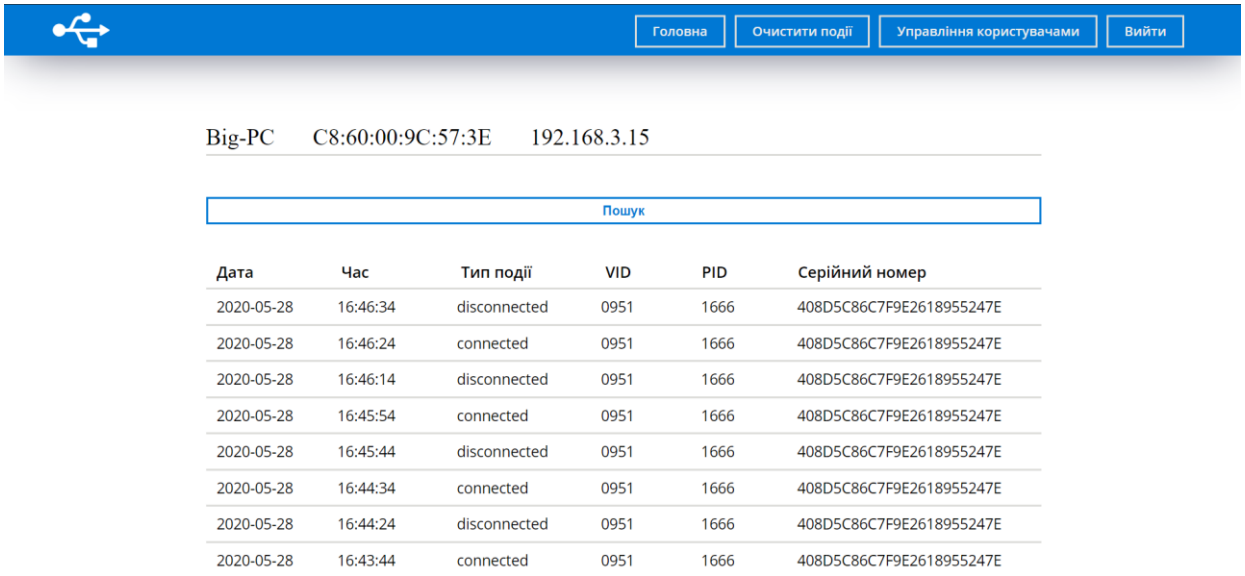


Рисунок 4.12 – Сторінка веб-додатку з подіями на конкретному комп’ютері

Перевірка ТС4

Сценарій перевірки

1. Відкриваємо веб-додаток.

2. У формі авторизації вводимо, логін «coliby123» та пароль «123456v» (рис. 4.13).

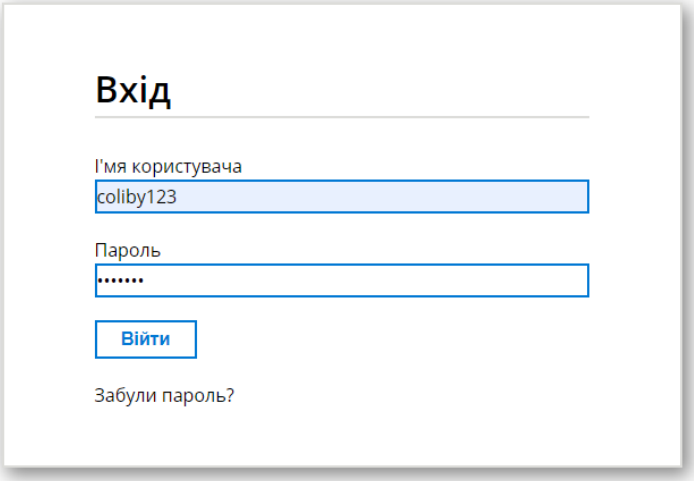


Рисунок 4.13 – Форма входу

3. Натискаємо на кнопку «Війти».

Результат

Авторизація пройшла успішно, користувач отримав доступ до системи (рис. 4.14).

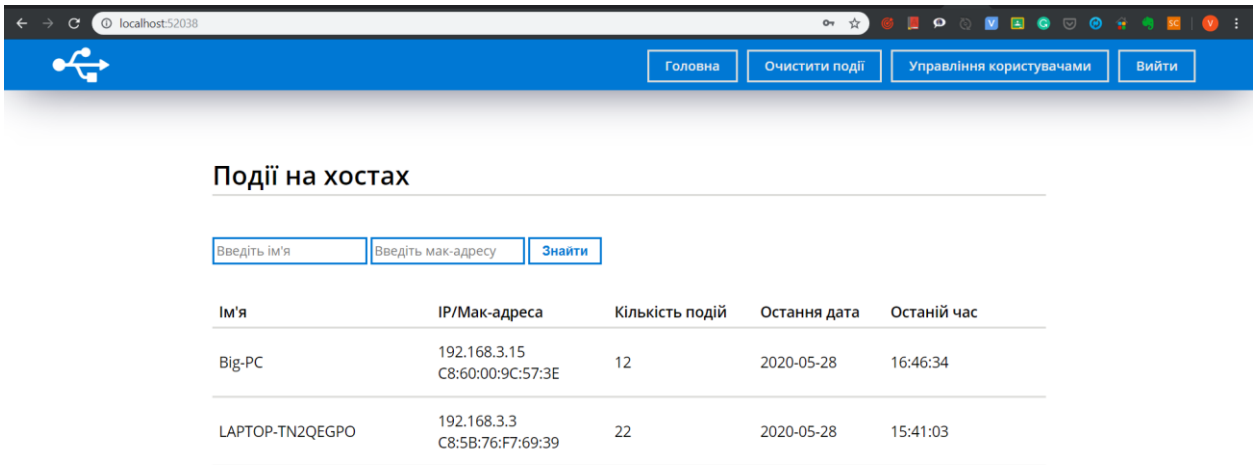


Рисунок 4.14 – Відкрита головна сторінка після авторизації

Перевірка ТС5

Сценарій перевірки

1. На сторінці входу натискаємо на «Забули пароль?».
2. Вводимо email «coliby45@gmail.com» та натиснемо на «Надіслати» (рис. 4.15).

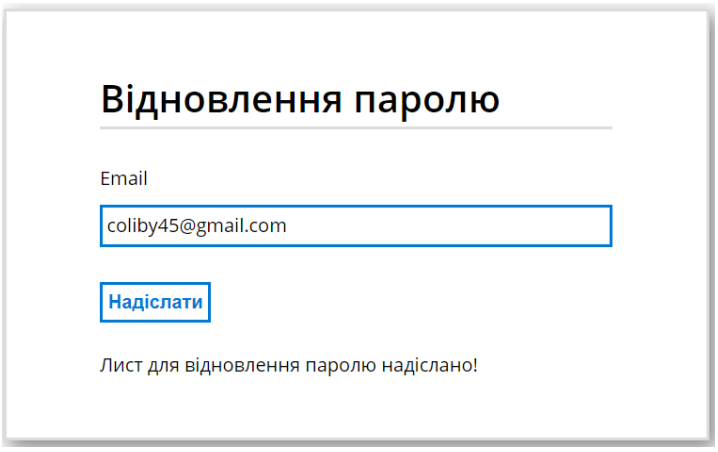


Рисунок 4.15 – Форма відновлення паролю

3. Переходимо за посиланням (рис. 4.16).

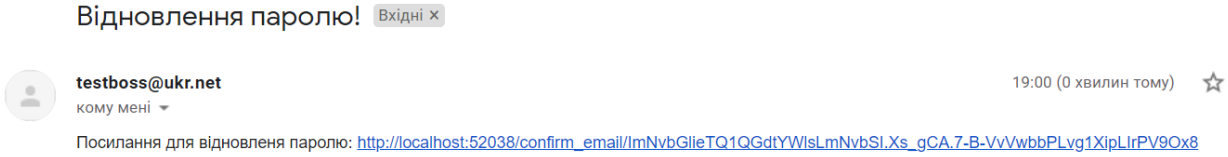


Рисунок 4.16 – Лист, що надійшов на почтову скриньку

4. Вводимо новий пароль «527133O» та натиснемо «Змінити» (рис. 4.17).

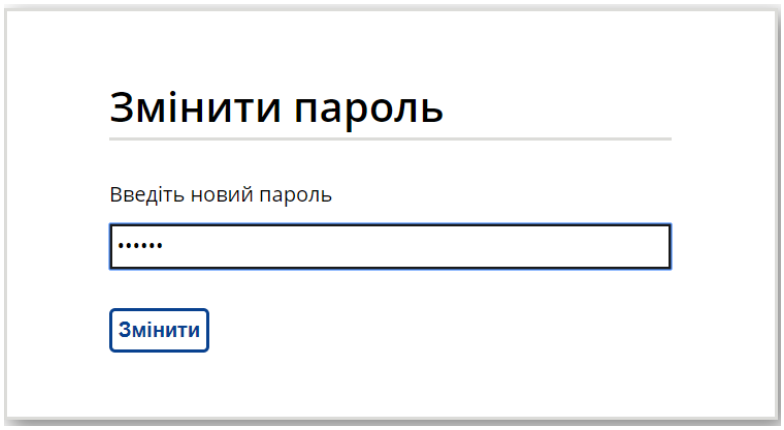


Рисунок 4.17 – Форма зміни паролю

Результат

Користувач потрапив на сторінку входу. Входимо за новим паролем (рис. 4.18).

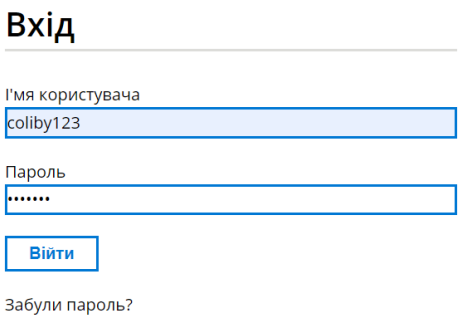


Рисунок 4.18 – Перевірка зміни паролю

Пароль змінений успішно. Користувач потрапив на головну сторінку(рис 4.19).

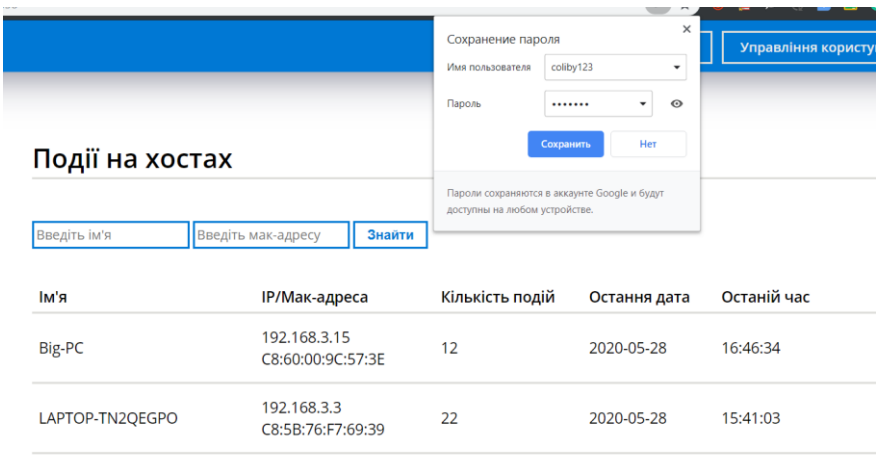


Рисунок 4.19 – Головна сторінка після авторизації за новим паролем

Перевірка ТС6

Сценарій перевірки

1. Входимо в систему від імені користувача зі звичайними правами (рис. 4.20).

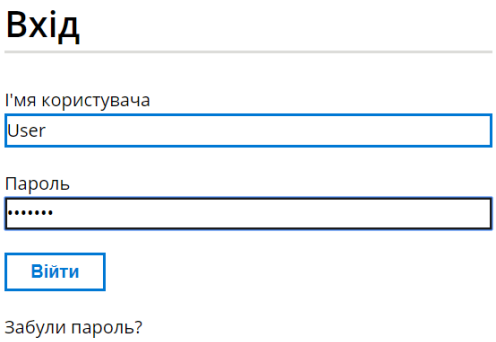


Рисунок 4.20 – Вхід зі звичайними правами

Результат

Вхід до системи пройшов вдало. Користувач має доступ лише до функцій перегляду подій (рис. 4.21).

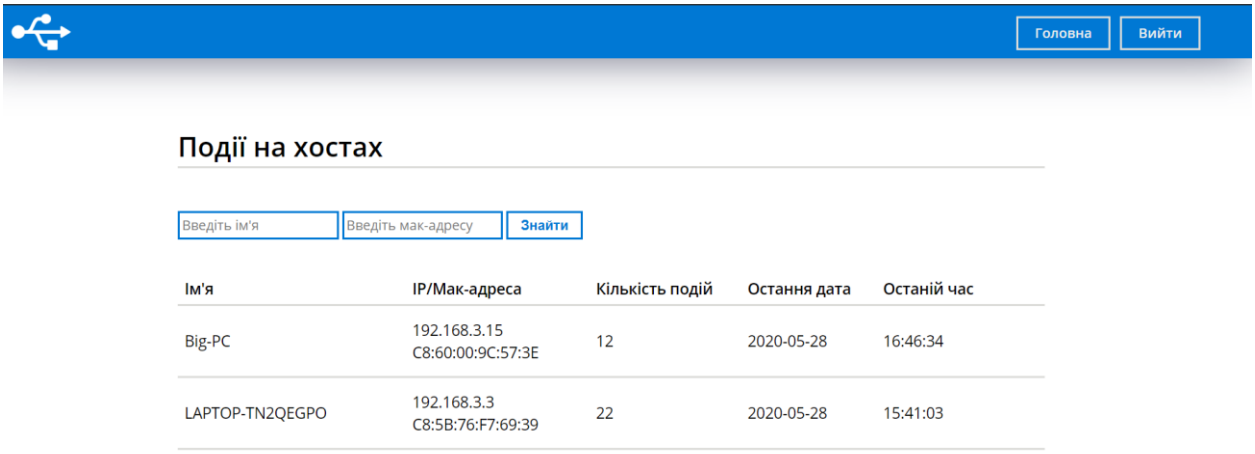


Рисунок 4.21 – Вигляд головної сторінки для користувача зі звичайними правами

Перевірка ТС7

Сценарій перевірки

1. Заходимо на сторінку від користувача з правами Адміністратора.
2. Натиснемо на «Очистити події».
3. Для видалення відмітимо комп’ютер «1111» та натиснемо видалити (рис. 4.22).

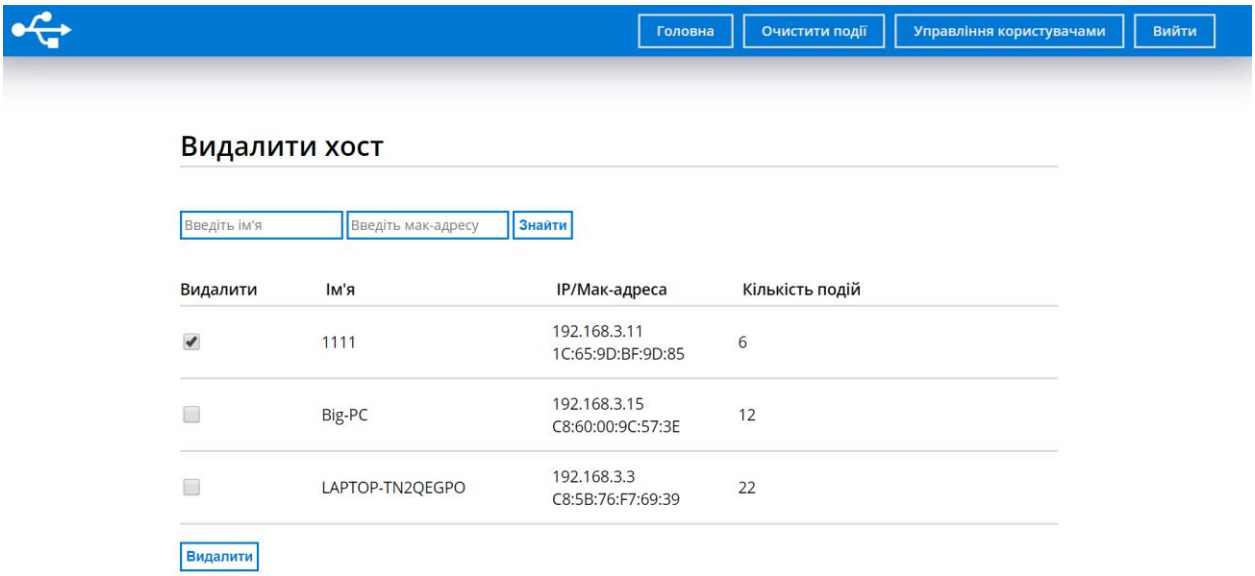


Рисунок 4.22 – Сторінка для видалення всієї інформації на комп’ютерах

Результат

Комп’ютер «1111» був видалений зі системи (рис. 4.23).

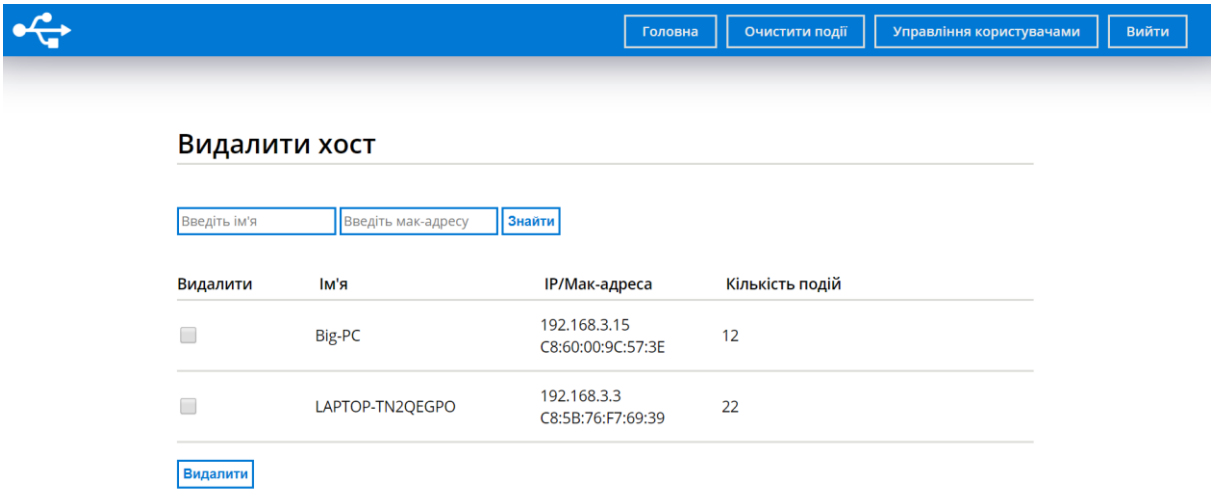


Рисунок 4.23 – Вигляд сторінки після видалення комп’ютера "1111"

Сценарій перевірки

1. Натискаємо на вузол «LAPTOP-TN2QEGPO».
2. Видалимо події, які відбулися з 17.05.2020 по 18.05.2020(рис. 4.24).

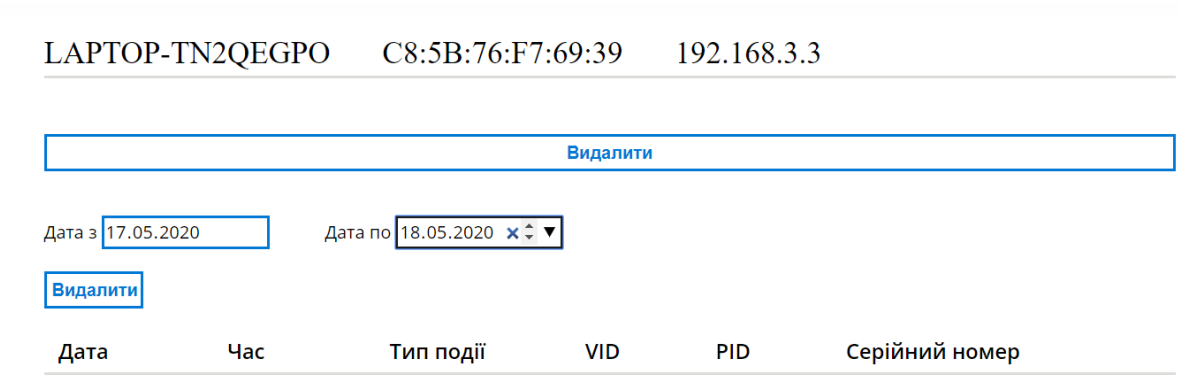


Рисунок 4.24 – Поле вводу для видалення подій на конкретному комп'ютері

3. Натиснемо на кнопку «Видалити»

Результат

Події, які відбулися з 17.05.2020 по 18.05.2020 були видалені (рис. 4.25, 4.26).

2020-05-19	14:02:09	disconnected	1005	B113	07B7150249030052
2020-05-18	12:03:26	connected	1005	B113	07B7150249030052
2020-05-17	19:03:35	disconnected	1005	B113	07B7150249030052
2020-05-17	12:40:48	connected	1005	B113	07B7150249030052
2020-05-17	12:40:28	disconnected	1005	B113	07B7150249030052
2020-05-17	12:24:10	connected	1005	B113	07B7150249030052
2020-05-17	12:12:56	disconnected	1005	B113	07B7150249030052
2020-05-17	12:11:26	connected	1005	B113	07B7150249030052

Рисунок 4.25 – Список подій до видалення

2020-05-26	14:12:06	connected	1005	B113	07B7150249030052
2020-05-26	14:10:35	disconnected	1005	B113	07B7150249030052
2020-05-26	14:10:35	disconnected	1005	B113	07B7150249030052
2020-05-26	14:08:56	connected	1005	B113	07B7150249030052
2020-05-23	19:43:16	disconnected	8564	1000	15T2V7G8I9P0XXKK
2020-05-23	17:38:28	connected	8564	1000	15T2V7G8I9P0XXKK
2020-05-19	14:02:09	disconnected	1005	B113	07B7150249030052

Рисунок 4.26 – Список подій після видалення

Перевірка ТС8

Сценарій перевірки

1. Відкриваємо головну сторінку.
2. Введемо дані для пошуку (рис. 4.27).

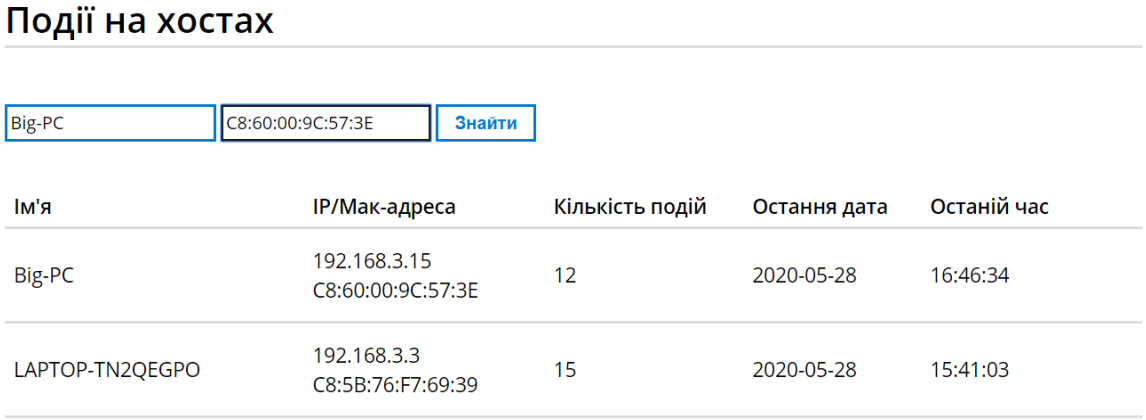


Рисунок 4.27 – Заповнення даних для пошуку комп’ютера

3. Натиснемо «Знайти».

Результат

Система знайшла заданий комп’ютер (рис. 4.28).

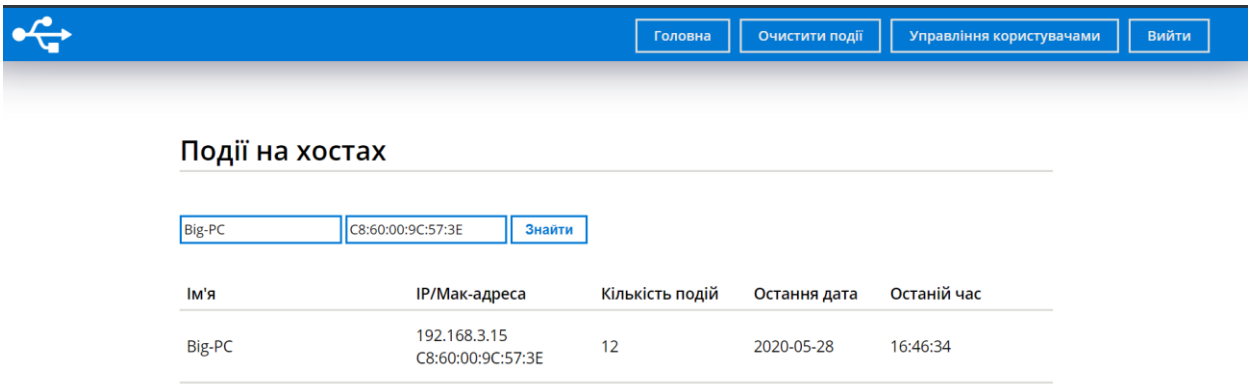


Рисунок 4.28 – Результат пошуку

Сценарій перевірки

1. Натискаємо на комп’ютер «LAPTOP-TN2QEGPO».

2. Вводимо значення для пошуку (рис. 29).

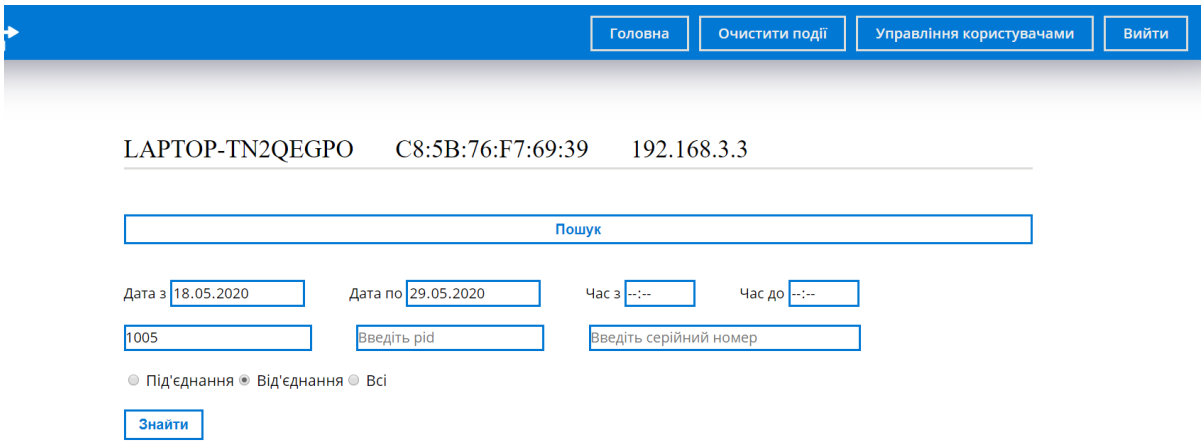


Рисунок 4.29 – Заповнення параметрів для пошуку подій на конкретному комп’ютері

3. Натиснемо «Знайти».

Результат

Система знайшла події, які відповідають заданим параметрам (рис. 4.30).

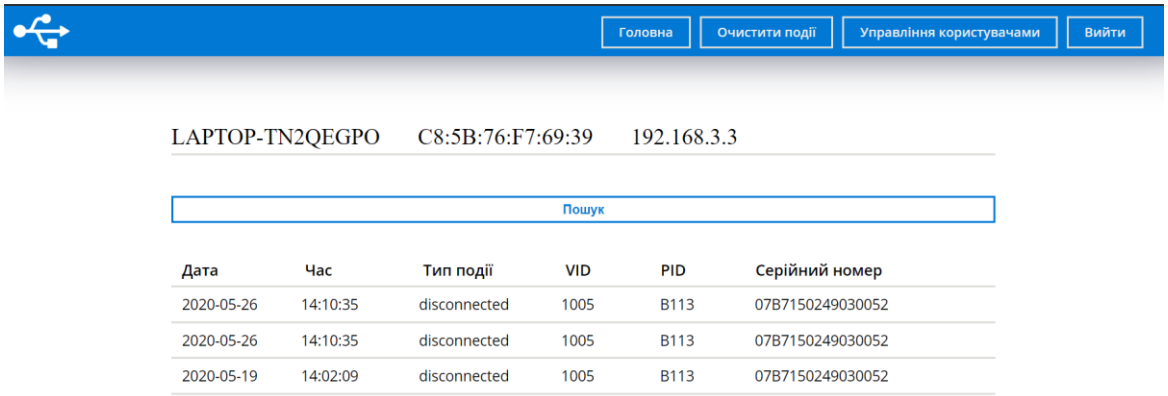


Рисунок 4.30 – Знайдені події

Перевірка ТС9

Сценарій перевірки

1. Натискаємо на «Управління користувачами».
2. Натискаємо на «Додати користувача».

3. Вводимо дані (рис. 4.31).

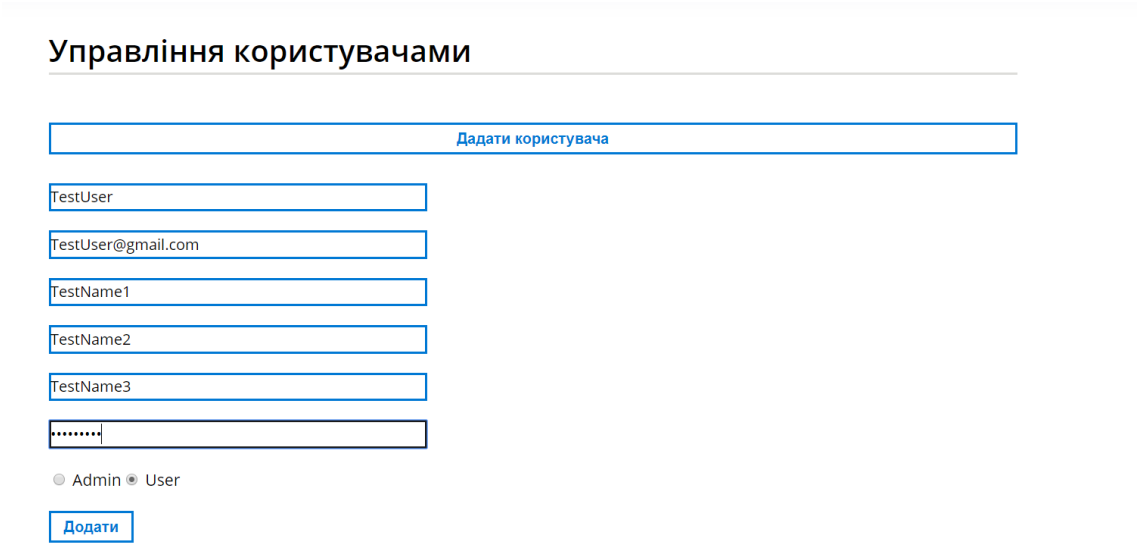


Рисунок 4.31 Заповнення даних для створення користувача

4. Натиснемо «Додати».

Результат

Тестовий користувач був добавлений до системи (рис. 4.32).

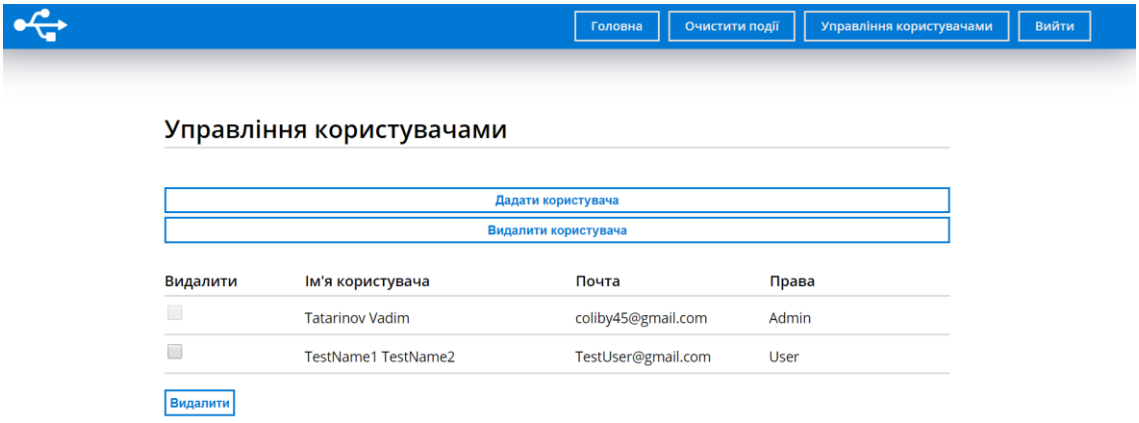


Рисунок 4.32 – Вигляд сторінки після створення користувача

Сценарій перевірки

1. Відмічаємо галочкою «TestName1 TestName2» (рис. 4.33).

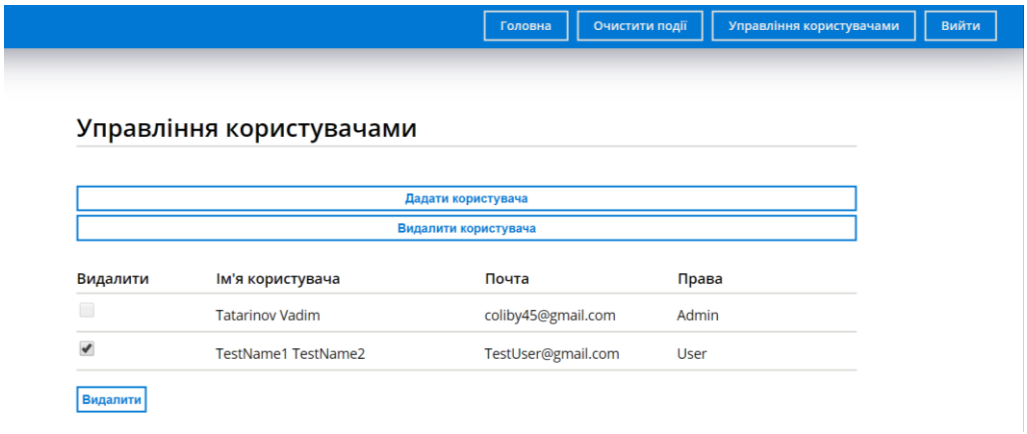


Рисунок 4.33 – Вибір користувача для видалення

2. Натиснемо на «Видалити».

Результат

Користувач був видалений з системи (рис. 4.34).

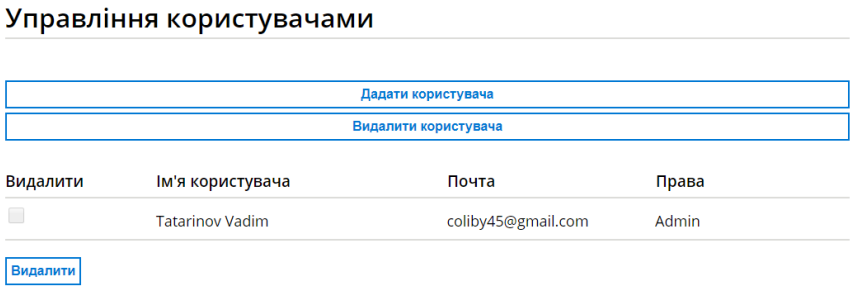


Рисунок 4.34 – Вигляд сторінки після видалення

Висновок

У цій роботі була створена система моніторингу підключень USB-пристроїв для проведення аудиту кібербезпеки. Створений програмний продукт, який відповідає потребам невеликої компанії, що хоче відслідковувати підключення USB-пристроїв для зменшення часу реагування на витік інформації та збільшення шансів встановити особу порушника. Розроблений продукт:

- підтримує операційну систему Windows;
- має змогу відслідковувати підключення USB-пристроїв на комп’ютерах за допомогою встановленої на них клієнтської служби;
- здатен централізовано обробляти події, які сталися на комп’ютерах;
- має веб-додаток, який забезпечує централізований перегляд події;
- має зручний та інтуїтивно зрозумілий інтерфейс. Відображає ім’я, мак-адресу, IP-адресу комп’ютерів на яких сталися події. Дає змогу переглянути список подій, які сталися на конкретному комп’ютері. Кожна подія містить таку інформацію, як PID, VID, серійний номер, тип, час та дату події під’єднання або від’єднання USB-пристрою.
- дає доступ до перегляду тільки після авторизації;
- здатен відновити пароль для зареєстрованого користувача;
- забезпечує розмежований доступ до функцій перегляду, управління користувачами та видалення застарілої інформації.

Під час вирішення поставленої задачі було проведено аналіз наявних рішень, проаналізована документація Windows, визначені технології, які дозволять реалізувати систему за невеликий проміжок часу, проведено проектування системи. Після завершення програмної реалізації всі компоненти системи були ретельно перевірні на працездатність.

Для визначення переліку робі та часу їх виконання, оцінки ризиків та визначити стратегії їх уникнення, було виконано планування робіт.

Система реалізована у вигаді двох Windows служб, утиліт для налаштування параметрів запуску служб та веб-додатку.

Перша служба встановлюється на комп’ютер, над яким буде проводитись спостереження. Для відстеження підключень USB-пристроїв вона використовує вбудований інтерфейс WMI, який призначений для управління та моніторингу операційної системи Windows, та розроблена за допомогою мови програмування C#.

Друга служба виконує задачі централізованого збору подій та збереження їх до бази даних. Реалізована також за допомогою мови C#.

За допомогою утиліт налаштування параметрів, які реалізуються за допомогою мови C#, користувач може налаштовувати параметри, що служба потребує для запуску.

Веб-додаток виконує задачі відображення отриманої інформації. Бекенд веб-додатку було реалізовано за допомогою Python та мікрофреймворку Flask. Інтерфейс реалізовано за допомогою стандартних технологій HTML, CSS та JavaScript.

Щоб зберігати отримані дані, було використано PostgreSQL.

Отже, розроблена система покращить рівень безпеки невеликих компаній, дозволивши адміністраторам з безпеки швидше реагувати на витік інформації та виявляти зловмисників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About WMI [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/wmisdk/about-wmi.

2. About Event Tracing [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing.

3. Determining the Type of Event to Receive [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/wmisdk/determining-the-type-of-event-to-receive.

4. WMI for Detection and Response [Електронний ресурс] // United States. Department of Homeland Security. – 2016. – Режим доступу до ресурсу: https://www.hsdl.org/?abstract&did=795007.

5. WqlEventQuery Constructors [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/system.management.wqleventquery.-ctor?view=dotnet-plat-ext-3.1#System_Management_WqlEventQuery__ctor.

6. ManagementEventWatcher Class [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/system.management.managementeventwatcher?view=dotnet-plat-ext-3.1.

7. Michael U. CANARY FOR USB PORTS [Електронний ресурс] / Uttmark Michael. – 2017. – Режим доступу до ресурсу: https://hackaday.com/2017/04/04/canary-for-usb-ports/.

8. USBDeview [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.softpedia.com/get/System/System-Info/USBDeview.shtml.

9. CleverControl Review [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://reviews.financesonline.com/p/clevercontrol/.

10. USB Security Analyzer Detects and Responds to Potential Threats [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.solarwinds.com/security-event-manager/use-cases/usb-security-analyzer.

11. PRODUCT REVIEW: AlienVault USM Anywhere [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.cybersecurity-insiders.com/product-review-alienvault-usm-anywhere/.

12. Tutorial: Create a Windows service app [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer.

13. Pro and Cons of C# Programming Language [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.pros-cons.net/pro-and-cons-of-csharp-programming-language/.

14. Faisal P. Create A Windows Service In C# [Електронний ресурс] / Pathan Faisal. – 2020. – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/create-windows-services-in-c-sharp/.

15. Pros & Cons of Using Python For Web Development [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://thepythonguru.com/pros-cons-of-using-python-for-web-development/.

16. Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://logz.io/blog/relational-database-comparison/.

17. Web Design 101: How HTML, CSS, and JavaScript Work [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://blog.hubspot.com/marketing/web-design-html-css-javascript.

18. UML Use Case Diagrams [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.uml-diagrams.org/use-case-diagrams.html.

19. Activity Diagrams [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.uml-diagrams.org/activity-diagrams.html.

20. Deployment Diagrams Overview [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.uml-diagrams.org/deployment-diagrams-overview.html.

21. What is a Data Flow Diagram [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.lucidchart.com/pages/data-flow-diagram.

22. What is Entity Relationship Diagram (ERD)? [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/.

23. Flask Tutorial [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.tutorialspoint.com/flask/index.htm.

24. C# Tutorials [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docsmicrosoft.com/en-us/dotnet/csharp/tutorials/.

Додаток А

Технічне завдання

1 Призначення та мета системи

1.1 Призначення системи

Система призначення для відслідковування подій підключення USB-пристроїв для проведення аудиту кібербезпеки.

1.2 Мета створення системи

Забезпечити можливість слідкувати за підключеннями USB-пристроїв до корпоративних комп’ютерів.

1.3 Цільова аудиторія

Невеликі компанії, які потребують забезпечити контроль за USB-пристроями та не мають змогу придбати дорогі системи з непотрібними для вирішення цієї задачі функціями.

2 Вимоги до системи

2.1 Загальні вимоги

2.1.1 Вимоги до структури системи

Система має мати таку структуру:

- клієнтська служба, яка встановлюється на комп’ютер над, яким потрібно здійснювати моніторинг;
- серверна служба;
- веб-додаток;

Вимоги до персоналу

Адміністратор повинен вміти встановлювати та налаштовувати базу даних PostgreSQL, мати базові знання з мережевих технологій для налаштування взаємозв’язку серверної та клієнтської служби, вміти розгорнути веб-додаток за допомогою веб-сервера. Для подальшого користування спеціальні навички не потрібні.

2.1.2 Вимоги до прав доступу

Інформація про події надається тільки після авторизації.

Користувачів системи розділяють на дві групи:

1. Звичайний користувач
2. Адміністратор

Звичайний користувач має доступ лише до перегляду подій. Адміністратор може переглядати події, видаляти застарілу інформацію та створювати або видаляти користувачів.

2.2 Вимоги до компонентів системи

2.2.1 Вимоги до клієнтської служби

2.2.1.1 Функціональні вимоги до клієнтської служби

Клієнтська служба має вміти:

- Відслідковувати підключення USB-пристрою до комп’ютера;
- Надсилати події на сервер.

2.2.1.2 Вимоги до реалізації клієнтської служби

Клієнтська служба мусить реалізовуватися за допомогою мови програмування C#. Налаштування параметрів для запуску відбувається за допомогою утиліти.

2.2.1.3 Технічні та програмні вимоги для роботи клієнтської служби

Для своєї роботи клієнтська служба потребує:

- Операційну систему Windows 7/8/10;

- .NET Framework 4.5 - 4.8 та вище.

2.2.2 Вимоги до серверної служби

2.2.2.1 Функціональні вимоги до серверної служби

Серверна служба має вміти:

- Приймати повідомлення від клієнтської служби;
- Перевірити формат повідомлення;
- Записати дані до бази даних.

2.2.2.2 Вимоги до реалізації серверної служби

Серверна служба мусить реалізовуватися за допомогою мови програмування C#. Налаштування параметрів для запуску відбувається за допомогою утиліти.

2.2.2.3 Технічні та програмні вимоги для роботи серверної служби

Для своєї роботи серверна служба потребує:

- Операційну систему Windows 7/8/10;
- .NET Framework 4.5 - 4.8 та вище.

2.2.3 Вимоги до веб-додатку

2.2.3.1 Функціональні до веб-додатку

Веб-додаток має вміти:

- відображати події, які відбувались на комп’ютерах. Відображати інформацію про USB-пристрій таку, як PID, VID, серійний номер, час події, дату події;
- відображати загальну інформацію про комп’ютер на якому сталась подія таку, як фізична адреса комп’ютера, IP-адреса, ім’я комп’ютера;
- надавати доступ користувачам тільки після авторизації;
- відновлювати пароль;
- надавати доступ до різних функції додатку відповідно до прав доступу користувача;

- створювати та видаляти користувачів.

2.2.3.2 Вимоги до реалізації серверної служби

Інтерфейс веб-додатку має реалізовуватися за допомогою HTML, CSS та JavaScript. Серверна частина сайту має бути реалізовано за допомогою мови Python та фреймверку Flask.

2.2.3.3 Технічні та програмні вимоги для роботи серверної служби

Інтерфейс правильно відображається за допомогою браузерів Google Chrome 83.0.4 – та вище, Firefox 76.0.1 – та вище. Для роботи серверної частини потрібен інтерпретатор Python 3.8.3 та будь-який веб-сервер.

3. Програмна документація

- Діаграми UML;
- Технічне завдання;
- Програмний код системи;

4. Етапи розробки

- визначення та оформлення завдання до роботи;
- планування робіт;
- здійснити аналіз предметної області;
- дослідити документацію windows;
- налаштувати середовище для розробки програмного коду;
- розробити службу клієнт;
- розробити серверну службу;
- спроектувати структуру бази даних;
- розробити веб-додаток;
- протестувати систему;
- розробити інструкцію користувача;
- оформити звіт;
- презентація та захист;

Додаток Б

Планування робіт

Мета проекту: створити систему моніторингу підключень USB-пристроїв для проведення аудиту кібербезпеки. Де адміністратор безпеки зможе централізовано переглядати пристрої, які були підключені до комп'ютері з встановленою службою для спостереження, за допомогою зручного та інтуїтивно зрозумілого веб-інтерфейсу. Деталізуємо проект методом SMART (табл. Б.1).

Таблиця Б.1 – SMART деталізація

Specific (конкретна)	Створити систему моніторингу підключення USB пристроїв для аудиту кібербезпеки.
Measurable (вимірювана)	Результатом роботи проекту є програмний продукт, що забезпечує моніторинг підключень USB-пристроїв.
Achievable (досяжна)	Реалізація системи моніторингу підключень USB пристроїв відбуваються за допомогою: C# , Python, HTML, CSS, JavaScript та база даних PostgreSQL .
ReLevant (реалістична)	Технічні засоби для реалізації є безкоштовними для завантажування та використання, що дає можливість реалізувати буть якій проект за допомогою їх. Розробник володіє необхідними знаннями для використання вище згаданих технологій.
Time – framed (Обмежена у часі)	Проект обмежений в часі та має бути виконаний відповідно до строків здачі бакалаврської роботи.

Планування змісту структури робіт відбувається за допомого інструменту WBS(Work Break Structure), що представляє роботи у вигляді ієрархічної структури, яка досягається методом послідовної декомпозиції (рис. Б.1).

Після розробки діаграми WBS наступним кроком є створення OBS(Organization structure), яка описує встановлену організаційну структуру для планування проектів та управління ресурсами (рис. Б.2).

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Опис ролі
Розробник	Татарінов В.Р	Виконує програмну реалізацію.
Проектувальник	Татарінов В.Р	Виконує проектування структур бази даних. Розроблює діаграму … ,та проектування інтерфейсу програми.
Тестувальник	Татарінов В.Р	Тестує працездатність компонентів системи.
Консультант	Кальченко В.В	Формує завдання, та узгоджує план робіт.

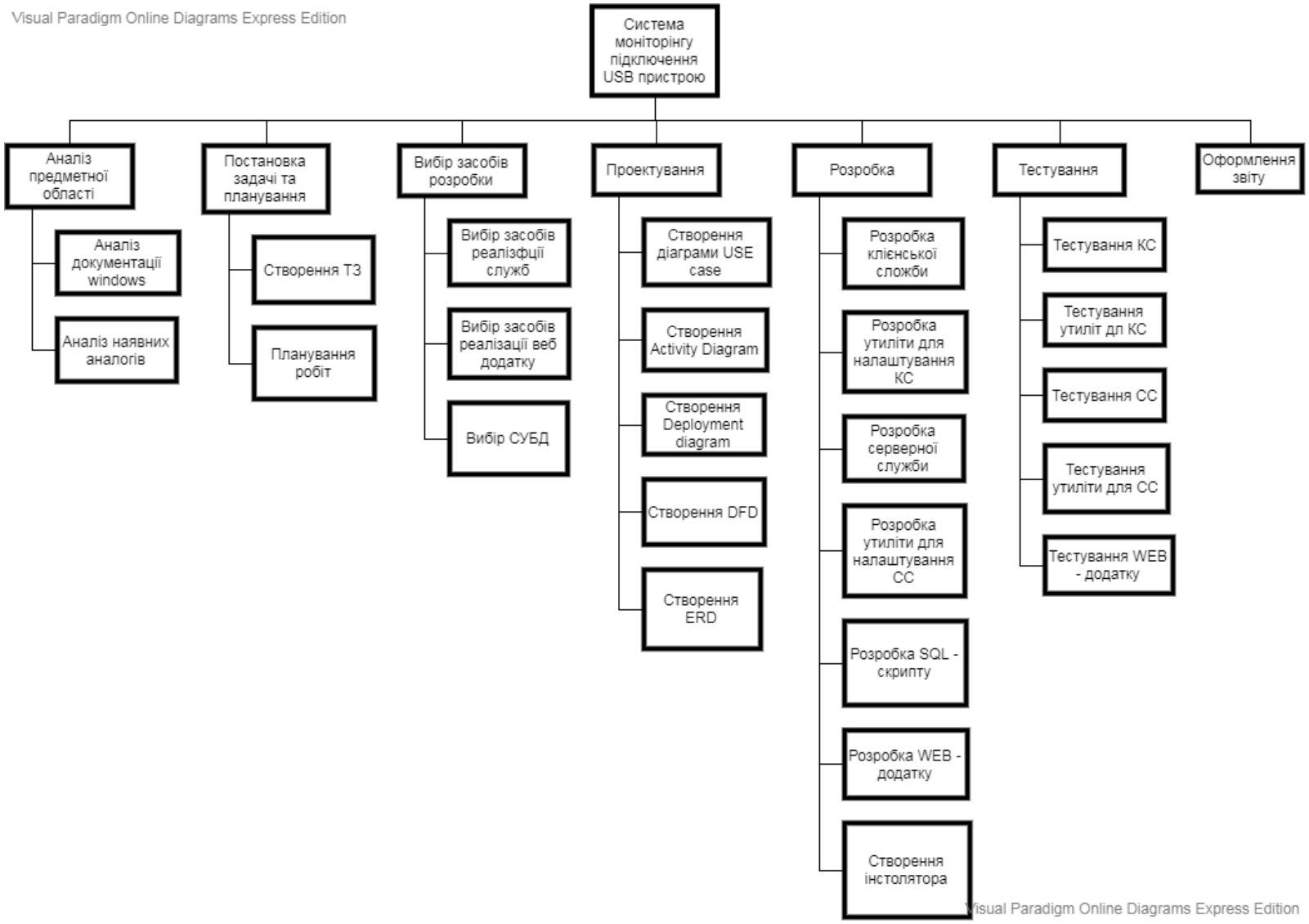
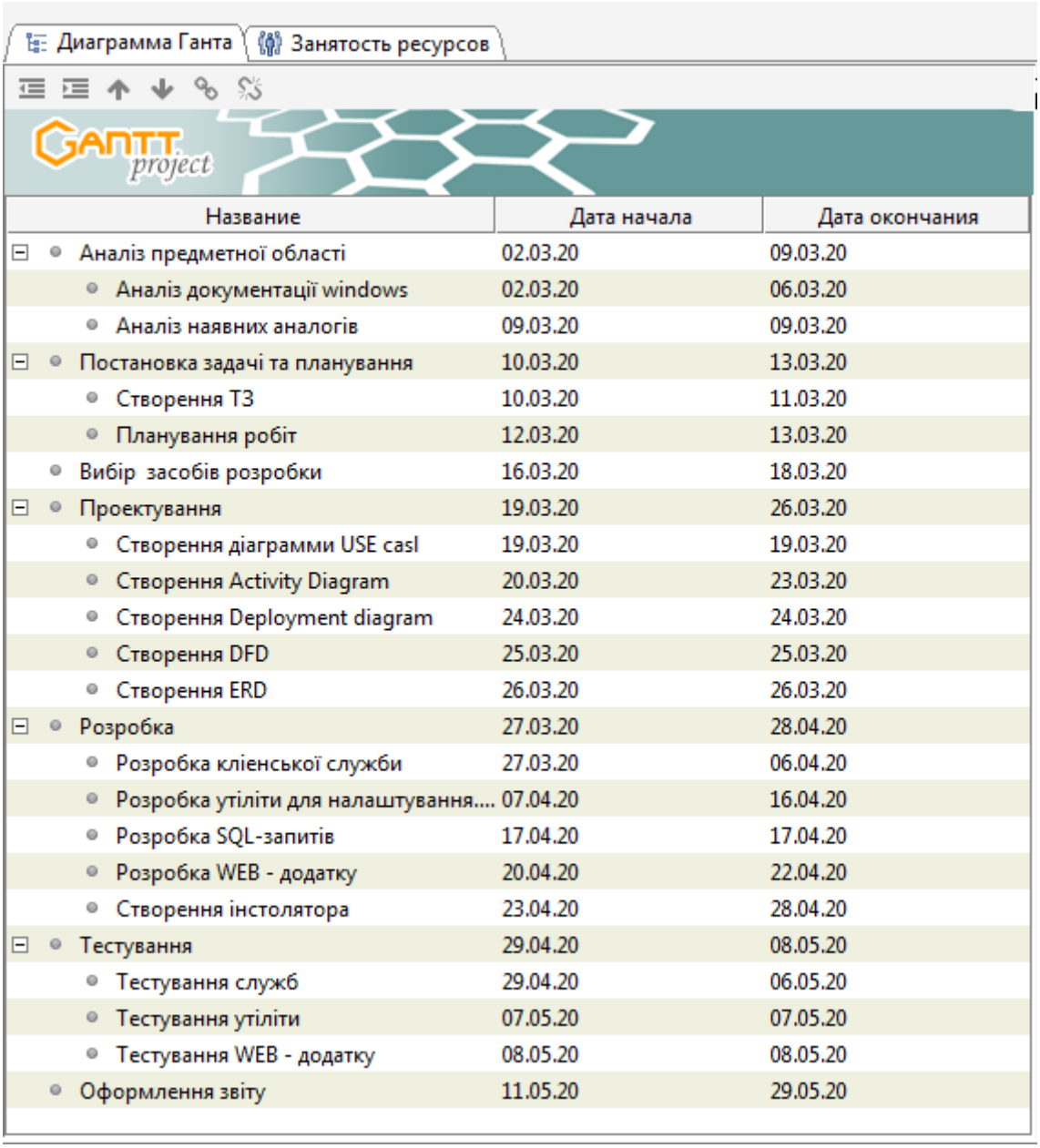


Рисунок Б.1 – WBS. Структура робіт

Діаграма Ґанта. Вона використовується для планування проектів будь-яких розмірів та показує, яку роботу потрібно виконати в визначений день. Також допомагає представити дати початку та кінця проекту за допомогою простого графіку. Нижче побудована діаграма Ґанта для планування робіт проекту (рис. Б.3, Б.4, Б.5, Б.6).



Название	Дата начала	Дата окончания
Аналіз предметної області	02.03.20	09.03.20
Аналіз документації windows	02.03.20	06.03.20
Аналіз наявних аналогів	09.03.20	09.03.20
Постановка задачі та планування	10.03.20	13.03.20
Створення ТЗ	10.03.20	11.03.20
Планування робіт	12.03.20	13.03.20
Вибір засобів розробки	16.03.20	18.03.20
Проектування	19.03.20	26.03.20
Створення діаграмми USE casl	19.03.20	19.03.20
Створення Activity Diagram	20.03.20	23.03.20
Створення Deployment diagram	24.03.20	24.03.20
Створення DFD	25.03.20	25.03.20
Створення ERD	26.03.20	26.03.20
Розробка	27.03.20	28.04.20
Розробка кліенської служби	27.03.20	06.04.20
Розробка утіліти для налаштування....	07.04.20	16.04.20
Розробка SQL-запитів	17.04.20	17.04.20
Розробка WEB - додатку	20.04.20	22.04.20
Створення інстолятора	23.04.20	28.04.20
Тестування	29.04.20	08.05.20
Тестування служб	29.04.20	06.05.20
Тестування утіліти	07.05.20	07.05.20
Тестування WEB - додатку	08.05.20	08.05.20
Оформлення звіту	11.05.20	29.05.20

Рисунок Б.3 – Список робіт

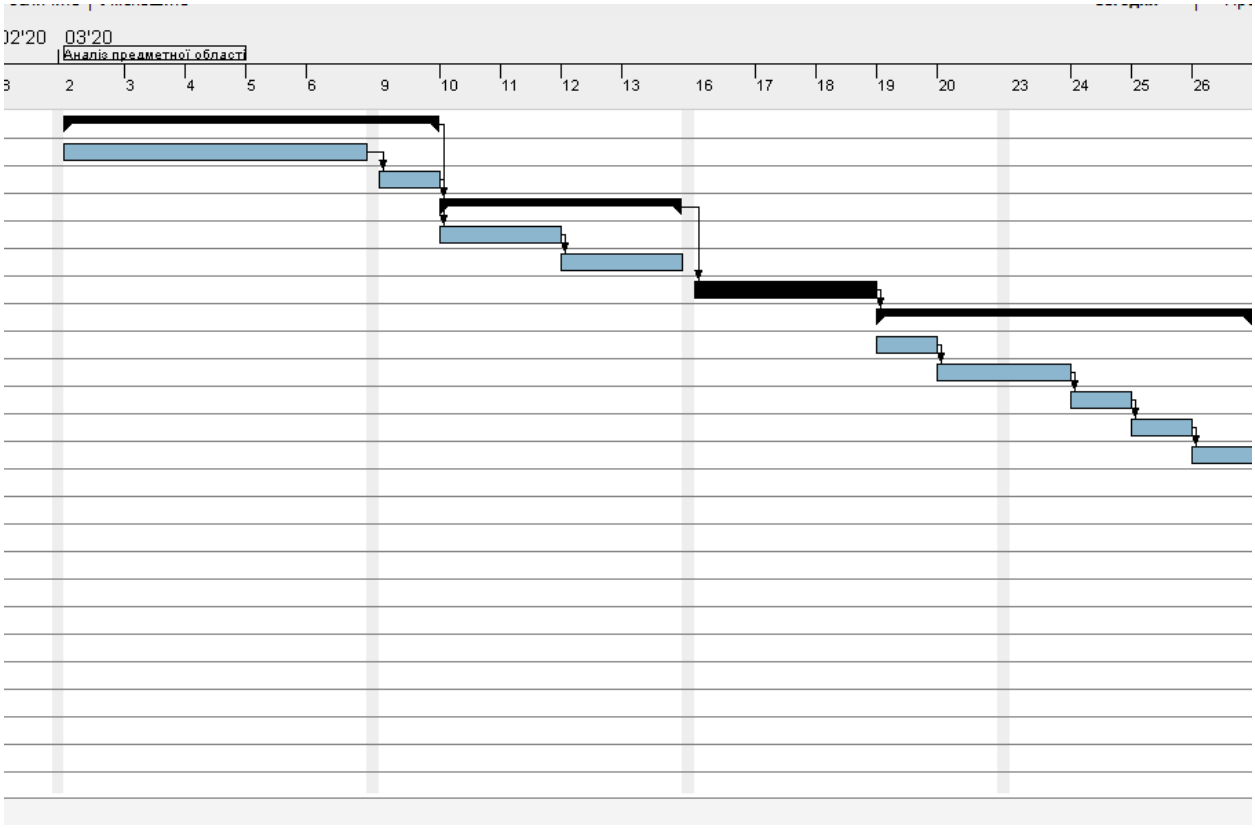


Рисунок Б.4 – Діаграма Ґанта

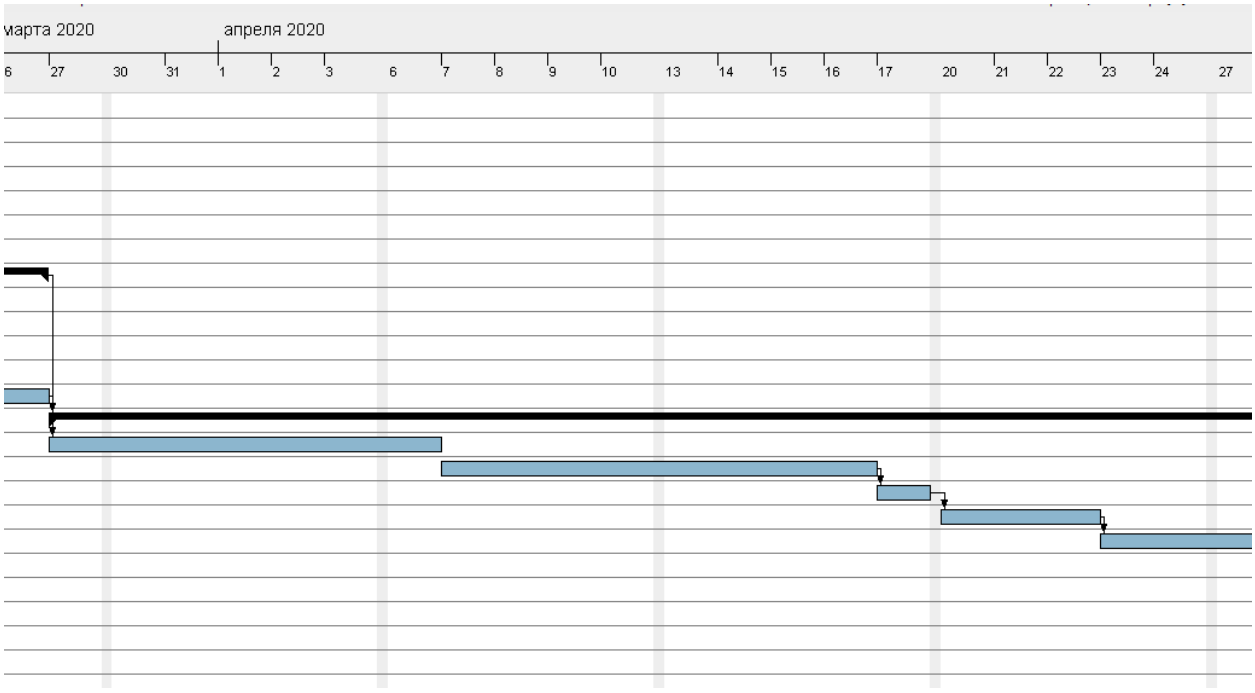


Рисунок Б.5 – Діаграма Ґанта

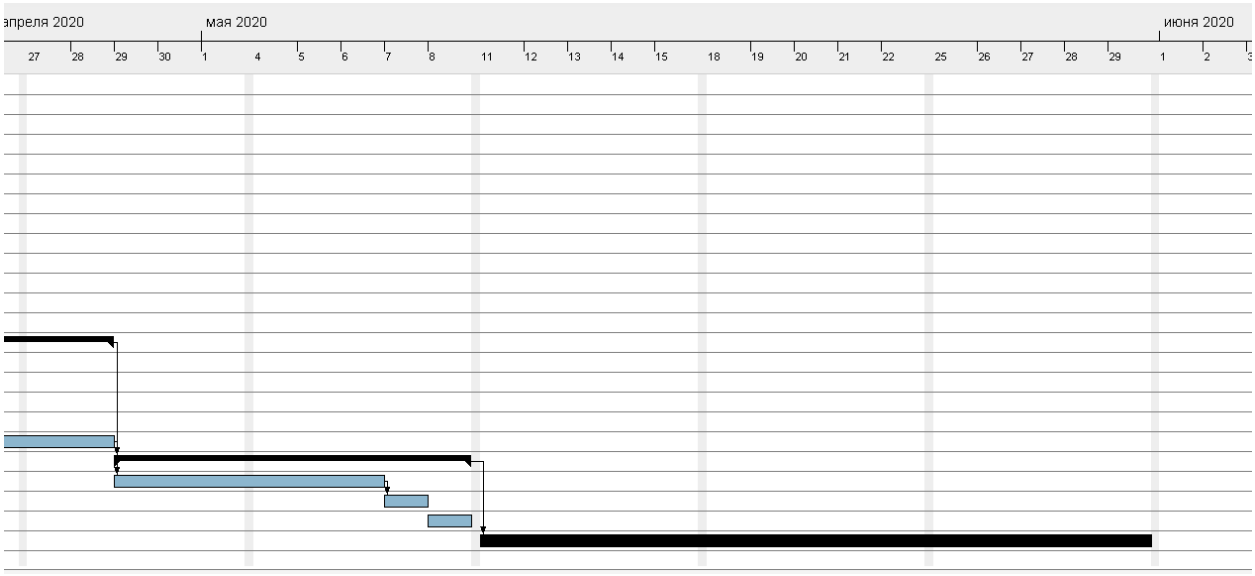


Рисунок Б.5 – Діаграма Ґанта

Аналіз ризиків. Проведемо оцінку ризиків для визначення ступеня важливості ризику, які можуть виникнути під час виконання проекту, що дозволить ефективно протидіяти більш суттєвим ризикам.

Визначимо показники за якими будем оцінювати ризики (табл. Б.3, Б.4).

Таблиця Б.3 – Ймовірність виникнення ризику

Позначення	Ймовірність ризику
М	Мінімальна
Н	Низька
С	Середня
ПВ	Помірно висока
В	Висока

Таблиця Б.4 – Величина втрат

Позначення	Ступінь втрат
М	Мінімальна
Н	Низька
С	Середня
ПВ	Помірно висока
В	Висока

За допомогою визначних показників була побудована матриця визначення ступеню ризиків (табл. Б.5).

Таблиця Б.5 – Матриця визначення ступеню ризиків

Йомові./Втрати	М	Н	С	ПВ	В
М	М	М	Н	Н	С
Н	М	Н	Н	С	ПВ
С	Н	Н	С	ПВ	В
ПВ	Н	С	ПВ	В	В
В	С	ПВ	В	В	В

На основі вище згаданих даних була побудована таблиця визначення ступеня ризику (табл. Б.6).

Таблиця Б.6 – Визначення ступеня ризиків

№	Перелік ризиків проекту	Ймовірність ризику	Величина втрат	Рівень ризиків
1	Неякісне планування	С	ПВ	ПВ
2	Недотримання плану робіт	Н	ПВ	С
3	Суперечливість у вимогах	Н	С	Н

Продовження таблиці Б.6

№	Перелік ризиків проекту	Ймовірність ризику	Величина втрат	Рівень ризиків
4	Нездатність впоратись з поставленою задачею	С	В	В
5	Втрата або пошкодження апаратного забезпечення	Н	В	ПВ
6	Критичне пошкодження файлів	Н	В	ПВ
7	Неправильне або неповне тестування	В	С	В

Нижче визначений план запобігання та реагування на проектні ризики.

Таблиця Б.7 Запобігання та реакція на ризики

Проектні ризики	План запобігання ризику	План реагування на ризик
Неякісне планування	Чітко визначити список робіт. Детально оцінити та спланувати час на виконання кожної роботи.	Уразі виявлення недоліків в плануванні, треба виконати повторне планування, ураховуючи попередній досвід.
Недотримання плану робіт	Чітко визначити календарний план. Проведення звітності за виконані роботи відповідно термінам.	Обговорення можливостей внесення поправок в терміни з замовником.

Продовження таблиці Б.7

Проектні ризики	План запобігання ризику	План реагування на ризик
Суперечливість у вимогах	Детально обговорити вимоги до продукту із замовником. Зробити детальний опис технічного завдання.	Уточнення вимог та технічного завдання.
Нездатність впоратись з поставленою задачею	Проаналізувати вимоги до проекту та визначити ступінь кваліфікації працівників.	Підвищити ступінь кваліфікації працівників для вирішення поставленої задачі.
Втрата або пошкодження апаратного забезпечення	Провести огляд апаратного забезпечення перед початком робіт.	Замінити або відремонтувати пошкоджену частину апаратного забезпечення.
Критичне пошкодження файлів	Використовувати перевірене програмне забезпечення. Робити резервне збереження файлів.	Зробити перезапуск системи. Виконати відкат системи.
Неправильне або неповне тестування	Виконати повне тестування усіх компонентів системи.	Зробити повторне тестування.

Додаток В Програмний код

Код клієнтської служби

Service1.cs

```
using Microsoft.Win32;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Management;
using System.ServiceProcess;
using System.Text.RegularExpressions;
using USBServices;


namespace USBService
{

    public partial class Service1 : ServiceBase
    {
        private ManagementEventWatcher USBEventWatcher;
        private const string ipPattern = @"(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";
        private const string portPattern = @"([0-9]{1,4}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-
4][0-9]{2}|655[0-2][0-9]|6553[0-5])";
        private string ip = "127.0.0.1";
        private string port = "8889";
        private EventLog eventLog1;

        public Service1()
        {
            InitializeComponent();
            WqlEventQuery queryToGetUSBObj = new WqlEventQuery(
                 "SELECT * FROM __InstanceOperationEvent WITHIN 10 WHERE TargetInstance ISA
'Win32_USBHub'");
            USBEventWatcher = new ManagementEventWatcher(queryToGetUSBObj);
            USBEventWatcher.EventArrived += HandleEventUSB;
        }

        private void HandleEventUSB(object sender, EventArrivedEventArgs e)
        {

            ManagementBaseObject targetObj =
(ManagementBaseObject)e.NewEvent.Properties["TargetInstance"].Value;
            string chackClass = (string)e.NewEvent.Properties["__CLASS"].Value;
            Dictionary<string, string> USBDataAndHostInfo = null;

            if (chackClass == "__InstanceDeletionEvent" || chackClass ==
"__InstanceCreationEvent")
            {
                CollectInfo collectData = new CollectInfo(eventLog1, targetObj, chackClass);
                USBDataAndHostInfo = collectData.CollectData();
                string output = JsonConvert.SerializeObject(USBDataAndHostInfo);
                NetConnection TCPConnection = new NetConnection();
                TCPConnection.SendData(Convert.ToString(output), ip, port, eventLog1);
            }
        }

        public void OnDebug()
        {
            OnStart(null);
```

```
        }

        protected override void OnStart(string[] args)
        {
            eventLog1 = new System.Diagnostics.EventLog();
            if (!System.Diagnostics.EventLog.SourceExists("EventUSBService"))
            {
                System.Diagnostics.EventLog.CreateEventSource(
                    "EventUSBService", "USBServiceLog");
            }
            eventLog1.Source = "EventUSBService";
            eventLog1.Log = "USBServiceLog";

            try
            {
                RegistryKey localMachineKey = Registry.LocalMachine;
                RegistryKey USBMonService =
localMachineKey.OpenSubKey(@"SYSTEM\CurrentControlSet\Services\MyUSBService", true);
                string ImagePath = USBMonService.GetValue("ImagePath").ToString();
                string[] SplitMainPath = ImagePath.Split(new char[] { '^' });
                if (SplitMainPath.Length > 1)
                {
                    string[] IpAndPort = SplitMainPath[1].Split(new char[] { ' ' });
                    if (Regex.IsMatch(IpAndPort[1], ipPattern, RegexOptions.IgnoreCase))
                    {
                        ip = IpAndPort[1];
                    }
                    if (Regex.IsMatch(IpAndPort[2], portPattern, RegexOptions.IgnoreCase))
                    {
                        port = IpAndPort[2];
                    }
                }
            }
            catch(Exception ex)
            {
               eventLog1.WriteEntry("Не вдалося встановити початкові IP та Port за допомогою
реєстру : " + ex, EventLogEntryType.Error, 10);
            }
            Dictionary<string, string> HostInfo = null;
            CollectInfo collectData = new CollectInfo(eventLog1);
            HostInfo = collectData.LaunchCollectData();
            string output = JsonConvert.SerializeObject(HostInfo);
            NetConnection TCPConnection = new NetConnection();
            TCPConnection.SendData(Convert.ToString(output), ip, port, eventLog1);
            eventLog1.WriteEntry("Служба почала роботу!", EventLogEntryType.Information, 1);
            USBEventWatcher.Start();
        }

        protected override void OnStop()
        {
            eventLog1.WriteEntry("Служба припинила роботу!", EventLogEntryType.Information,
2);
            USBEventWatcher.Stop();

        }
    }
}
```

CollectInfo.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Management;
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Text.RegularExpressions;

namespace USBService
{
    class CollectInfo
    {
        private string pattern =
@"(?<VID>(?<=VID_).*(?=&))|(?<PID>(?<=PID_).*[^\\](?=\\))|(?<SN>(?<=PID_.*\\)[^\\].*)";
        private EventLog eventLog1;
        private ManagementBaseObject targetObj;
        private string chackClass;
        public CollectInfo(EventLog eventLog1, ManagementBaseObject targetObj, string
chackClass) {
            this.eventLog1  = eventLog1;
            this.targetObj  = targetObj;
            this.chackClass = chackClass;
        }

        public CollectInfo(EventLog eventLog1)
        {
            this.eventLog1 = eventLog1;
        }

        private IPAddress GetCurrentIPAddress()
        {
            using (Socket SetNewSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, 0))
            {
                SetNewSocket.Connect("8.8.8.8", 65530);
                IPEndPoint IpendPoint = SetNewSocket.LocalEndPoint as IPEndPoint;
                return IpendPoint.Address;
            }
        }

        private string GetMacUseingIP(IPAddress ip)
        {
            byte[] macAddrByIP = new byte[6];
            int length = macAddrByIP.Length;
            SendARP(ip.GetHashCode(), 0, macAddrByIP, ref length);
            return BitConverter.ToString(macAddrByIP, 0, 6).Replace("-",":");
        }

        [DllImport("iphlpapi.dll", ExactSpelling = true)]
        public static extern int SendARP(int DeIP, int SoIP, [Out] byte[] pcMACAddr, ref int
pPhyAddrLen);
        private void GetCurrentIPandMac(ref Dictionary<string, string> SaveData)
        {
            IPAddress ip    = GetCurrentIPAddress();
            string    mac   = GetMacUseingIP(ip);
            SaveData.Add("ip" , ip.ToString());
            SaveData.Add("mac", mac);
        }

        private void getUSBinfo(ref Dictionary<string, string> SaveData)
        {
            string USBeventType = null;
            switch (chackClass)
```

```
            {
                case "__InstanceDeletionEvent":
                    USBeventType = "disconnected";
                    break;
                case "__InstanceCreationEvent":
                    USBeventType = "connected";
                    break;
            }
            Regex regex = new Regex(pattern);
            MatchCollection PidVidNumber =
regex.Matches(Convert.ToString(targetObj["PNPDeviceID"]));

            SaveData.Add("date", DateTime.Now.ToString("dd.MM.yyyy"));
            SaveData.Add("time", DateTime.Now.ToString("T"));
            SaveData.Add("USBeventType", USBeventType);
            SaveData.Add("VID", PidVidNumber[0].Value);
            SaveData.Add("PID", PidVidNumber[1].Value);
            SaveData.Add("SN", PidVidNumber[2].Value);

        }

        private void getHostinfo(ref Dictionary<string, string> SaveData)
        {
            IPGlobalProperties objPCProp = IPGlobalProperties.GetIPGlobalProperties();
            NetworkInterface[] NetIntf = NetworkInterface.GetAllNetworkInterfaces();
            PhysicalAddress macAddress = NetIntf[0].GetPhysicalAddress();
            SaveData.Add("macAddress", Convert.ToString(macAddress));
            SaveData.Add("hostName", objPCProp.HostName);

        }

        public Dictionary<string, string> CollectData()
        {
            Dictionary<string, string> SaveData = new Dictionary<string, string>();

            try
            {
                getUSBinfo (ref SaveData);
                getHostinfo(ref SaveData);
                GetCurrentIPandMac(ref SaveData);
                SaveData.Add("massageType", "1");
            }
            catch (Exception ex)
            {
                eventLog1.WriteEntry("Виникла помилка при отримані даних про USB або хосту!
: " + ex, EventLogEntryType.Error, 11);
            }

            return SaveData;
        }

        public Dictionary<string, string> LaunchCollectData() {
            Dictionary<string, string> SaveData = new Dictionary<string, string>();
            try
            {
                getHostinfo(ref SaveData);
                GetCurrentIPandMac(ref SaveData);
                SaveData.Add("massageType", "2");
            }
            catch (Exception ex)
            {
                eventLog1.WriteEntry("Виникла помилка при отримані даних про USB або хосту!
: " + ex, EventLogEntryType.Error, 11);
```

```
            }
            return SaveData;
        }
    }
}
```

NetConnection.cs

```
using System;
using System.Diagnostics;
using System.Net.Sockets;
using System.Text;

namespace USBServices
{

    class NetConnection
    {

        TcpClient SetConClient = null;

        public void SendData(string message, string ip, string port, EventLog eventLog1)
        {

            try
            {
                SetConClient = new TcpClient(ip, Convert.ToInt32(port));
                NetworkStream mystream = SetConClient.GetStream();
                byte[] data = Encoding.Unicode.GetBytes(message);
                mystream.Write(data, 0, data.Length);

            }
            catch (Exception ex)
            {
                eventLog1.WriteEntry("Мережева помилка : " + ex, EventLogEntryType.Error,
12);
            }
            finally
            {
                if (SetConClient != null)
                    SetConClient.Close();
            }

        }
    }
}
```

Код серверної служби

Service1.cs

```
using Microsoft.Win32;
using System;
using System.Diagnostics;
using System.ServiceProcess;
using System.Text.RegularExpressions;

namespace MyUSBServer
{
    public partial class Service1 : ServiceBase
    {
        int port = 8889;
        string Username = "postgres";
        string Host = "127.0.0.1";
        string Password = "vadim123";
        string DB = "USBdb";
        private EventLog eventLog1;
        string ipPattern = @"(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-
5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";
        string portPattern = @"([0-9]{1,4}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-
9]{2}|655[0-2][0-9]|6553[0-5])";
        TCP_listener listener;

        public Service1()
        {
            InitializeComponent();
        }
        public void OnDebug()
        {
            OnStart(null);
        }

        protected override void OnStart(string[] args)
        {

            eventLog1 = new System.Diagnostics.EventLog();
            if (!System.Diagnostics.EventLog.SourceExists("EventUSBServer"))
            {
                System.Diagnostics.EventLog.CreateEventSource(
                    "EventUSBServer", "USBServerLog");
            }
            eventLog1.Source = "EventUSBServer";
            eventLog1.Log = "USBServerLog";

            try
            {
                RegistryKey localMachineKey = Registry.LocalMachine;
                RegistryKey USBMonService =
localMachineKey.OpenSubKey(@"SYSTEM\CurrentControlSet\Services\MyUSBServer", true);
                string ImagePath = USBMonService.GetValue("ImagePath").ToString();
                string[] SplitMainPath = ImagePath.Split(new char[] { '^' });
                if (SplitMainPath.Length > 1)
                {
                    string[] StartData = SplitMainPath[1].Split(new char[] { ' ' });

                    if (Regex.IsMatch(StartData[1], portPattern, RegexOptions.IgnoreCase))
                    {
```

```
                    port = Convert.ToInt32(StartData[1]);
                }

                Username = StartData[2];

                if (Regex.IsMatch(StartData[3], ipPattern, RegexOptions.IgnoreCase))
                {
                    Host = StartData[3];
                }

                Password = StartData[4];
                DB = StartData[5];
            }
        }
        catch (Exception ex)
        {
            eventLog1.WriteEntry("Не вдалося встановити початкові дані." + ex,
EventLogEntryType.Error, 10);
        }

        try
        {
            listener = new TCP_listener(Username, Host, Password, DB, port, eventLog1);
            listener.Start();
        }
        catch(Exception ex)
        {
            eventLog1.WriteEntry("Не вдалося почати прослуховувати порт : " + ex,
EventLogEntryType.Error, 11);
        }
        eventLog1.WriteEntry("Служба почала роботу!", EventLogEntryType.Information, 1);
    }

    protected override void OnStop()
    {
        eventLog1.WriteEntry("Служба припинила роботу!", EventLogEntryType.Information,
2);
        listener.Stop();
    }
}
}
```

TCP_listener.cs

```
using System;
using System.Diagnostics;
using System.Net.Sockets;
using System.Threading;
using System.Threading.Tasks;

namespace MyUSBServer
{
    class TCP_listener
    {
        Thread myListenerThread;
        TcpListener listener;
        private int port;
        private string Username;
        private string Host;
        private string Password;
        private string DB;
        private EventLog eventLog1;
```

```
        private bool _Alive = true;
        CancellationTokenSource cancellationToken;
        public TCP_listener(string Username, string Host, string Password,
                                        string DB, int port, EventLog eventLog1)
        {
            this.port = port;
            this.Username = Username;
            this.Host = Host;
            this.Password = Password;
            this.DB = DB;
            this.eventLog1 = eventLog1;
        }
        public void Start() {
            listener = TcpListener.Create(port);
            listener.Start();
            myListenerThread = new Thread(new ThreadStart(this.listenerProcessAsync));
            myListenerThread.Start();
        }
        public void Stop() {

            this._Alive = false;
            cancellationToken.Token.Register(() => listener.Stop());
            cancellationToken.Cancel();

        }
        private async void listenerProcessAsync()
        {
            cancellationToken = new CancellationTokenSource();
            try
            {
                while (_Alive)
                {
                    TcpClient client = await Task.Run(() => listener.AcceptTcpClientAsync(),
cancellationToken.Token);
                    ClientObj clientObjWorker = new ClientObj(client, eventLog1, Username,
Host, Password, DB);
                    Thread clientThreadWorker = new Thread(new
ThreadStart(clientObjWorker.Process));
                    clientThreadWorker.Start();
                }
            }
            catch (Exception ex)
            {
                eventLog1.WriteEntry("Виникла помилка при прослуховуванні порту : " + ex,
EventLogEntryType.Error, 12);
                if (listener != null)
                    listener.Stop();
            }
        }


    }
}
```

ClientObj.cs

```
using Newtonsoft.Json.Linq;
using Newtonsoft.Json.Schema;
using Npgsql;
using System;
using System.Diagnostics;
using System.Net.Sockets;
```

```
using System.Text;

namespace MyUSBServer
{
    public class ClientObj
    {
        public TcpClient clientUSB;
        private string Username;
        private string Host;
        private string Password;
        private string DB;
        private EventLog eventLog1;
        public ClientObj(TcpClient tcpClient, EventLog eventLog1,
            string Username, string Host, string Password, string DB)
        {
            clientUSB = tcpClient;
            this.eventLog1 = eventLog1;
            this.Username = Username;
            this.Host = Host;
            this.Password = Password;
            this.DB = DB;
        }

        [Obsolete]
        public void Process()
        {
            NetworkStream streamUSB = null;
            DB_Worker db = null;
            try
            {
                streamUSB = clientUSB.GetStream();
                byte[] dataUSB = new byte[64];

                StringBuilder builderUSB = new StringBuilder();

                int bytes = 0;
                do
                {
                    bytes = streamUSB.Read(dataUSB, 0, dataUSB.Length);
                    builderUSB.Append(Encoding.Unicode.GetString(dataUSB, 0, bytes));
                }
                while (streamUSB.DataAvailable);
                string message = builderUSB.ToString();
                string schemaJson1 = @"{ 'date' : 'string',
                                         'time':'string',
                                         'USBeventType':'string',
                                         'VID':'string',
                                         'PID':'string',
                                         'SN':'string',
                                         'macAddress':'string',
                                         'hostName':'string',
                                         'ip' :'string',
                                         'mac': 'string',
                                         'massageType': 'string'}";

                string schemaJson2 = @"{
                                         'macAddress':'string',
                                         'hostName':'string',
                                         'ip' :'string',
                                         'mac': 'string',
                                         'massageType': 'string'}";

                JsonSchema schema1 = JsonSchema.Parse(schemaJson1);
```

```
                JsonSchema schema2 = JsonSchema.Parse(schemaJson2);
                JObject usbEventInfo = JObject.Parse(message);
                bool valid1 = usbEventInfo.IsValid(schema1);
                bool valid2 = usbEventInfo.IsValid(schema2);
                if (valid1 || valid2)
                {

                    if ((string)usbEventInfo["massageType"] == "1")
                    {
                        db = new DB_Worker(eventLog1, Username, Host, Password, DB);
                        db.Open();
                        db.insert_usb_event(usbEventInfo);
                    }
                    else if ((string)usbEventInfo["massageType"] == "2")
                    {
                        db = new DB_Worker(eventLog1, Username, Host, Password, DB);
                        db.Open();
                        db.insert_start_info(usbEventInfo);
                    }
                }
                else
                {
                    eventLog1.WriteEntry("Не вірний формат повідомлення! ",
EventLogEntryType.Error, 13);
                }
            }
            catch (NpgsqlException ex)
            {
                eventLog1.WriteEntry("Не вдалося записати дані до бази даних : " + ex,
EventLogEntryType.Error, 14);
            }
            catch (Exception ex)
            {
                eventLog1.WriteEntry("Помилка : " + ex, EventLogEntryType.Error, 15);
            }
            finally
            {
                if (streamUSB != null)
                    streamUSB.Close();
                if (clientUSB != null)
                    clientUSB.Close();
                if(db != null)
                    if (db.con != null)
                        db.Close();
            }
        }
    }
}
```

DB_Worker.cs

```
using Newtonsoft.Json.Linq;
using Npgsql;
using System;
using System.Diagnostics;

namespace MyUSBServer
{
    class DB_Worker
    {
        private string Username;
        private string Host;
```

```
        private string Password;
        private string DB;
        private EventLog eventLog1;
        public DB_Worker(EventLog eventLog1, string Username, string Host, string Password,
string DB)
        {
            this.eventLog1 = eventLog1;
            this.Username = Username;
            this.Host = Host;
            this.Password = Password;
            this.DB = DB;
        }
        public NpgsqlConnection MyDBCon { private set; get; }
        public void Open()
        {
            var cs = String.Format("Host={0};Username={1};Password={2};Database={3}", Host,
Username, Password, DB);
            MyDBCon = new NpgsqlConnection(cs);
            MyDBCon.Open();
        }

        public void Close()
        {
            MyDBCon.Close();
        }

        public void insert_usb_event(JObject usbEventInfo)
        {
            var sqlUSB = "select * from insert_usb_event(:userpc, :dmac, :mac, :ip,
:eventdate, :eventtime, :VID, :PID, :SN, :typeevent)";
            var USBcmd = new NpgsqlCommand(sqlUSB, MyDBCon);
            USBcmd.Parameters.AddWithValue("userpc", (string)usbEventInfo["hostName"]);
            USBcmd.Parameters.AddWithValue("dmac", (string)usbEventInfo["macAddress"]);
            USBcmd.Parameters.AddWithValue("mac", (string)usbEventInfo["mac"]);
            USBcmd.Parameters.AddWithValue("ip", (string)usbEventInfo["ip"]);
            USBcmd.Parameters.AddWithValue("eventdate", (string)usbEventInfo["date"]);
            USBcmd.Parameters.AddWithValue("eventtime", (string)usbEventInfo["time"]);
            USBcmd.Parameters.AddWithValue("VID", (string)usbEventInfo["VID"]);
            USBcmd.Parameters.AddWithValue("PID", (string)usbEventInfo["PID"]);
            USBcmd.Parameters.AddWithValue("SN", (string)usbEventInfo["SN"]);
            USBcmd.Parameters.AddWithValue("typeevent",
(string)usbEventInfo["USBeventType"]);
            USBcmd.ExecuteNonQuery();
        }
        public void insert_start_info(JObject usbEventInfo)
        {
            var sqlUSB = "select * from update_net_info(:userpc, :dmac, :mac, :ip)";
            var UserCmd = new NpgsqlCommand(sqlUSB, MyDBCon);
            UserCmd.Parameters.AddWithValue("userpc", (string)usbEventInfo["hostName"]);
            UserCmd.Parameters.AddWithValue("dmac"  , (string)usbEventInfo["macAddress"]);
            UserCmd.Parameters.AddWithValue("mac"   , (string)usbEventInfo["mac"]);
            UserCmd.Parameters.AddWithValue("ip"    , (string)usbEventInfo["ip"]);
            UserCmd.ExecuteNonQuery();
        }
    }
}
```

Код утиліт для налаштування параметрів

Утиліта для налаштування клієнтської служби

```
using Microsoft.Win32;
using System;
using System.Security;
using System.ServiceProcess;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace SettingTools
{
    public partial class Form1 : Form
    {
        string ipPattern = @"(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-
5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";
        string portPattern = @"([0-9]{1,4}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-
9]{2}|655[0-2][0-9]|6553[0-5])";
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string ip = textIp.Text;
            string port = textPort.Text;
            bool startOrNot = true;

            if(ip == "")
            {
                MessageBox.Show("Введіть ip");
                startOrNot = false;
            }
            else if (!Regex.IsMatch(ip, ipPattern, RegexOptions.IgnoreCase))
            {
                MessageBox.Show("Невірний формат IP");
                startOrNot = false;
            }

            if (port == "")
            {
                MessageBox.Show("Введіть port");
                startOrNot = false;
            }
            else if (!Regex.IsMatch(port, portPattern, RegexOptions.IgnoreCase))
            {
                MessageBox.Show("Невірний формат Port");
                startOrNot = false;
            }

            if (startOrNot)
            {
                try
                {
                    RegistryKey localMachineKey = Registry.LocalMachine;
                    RegistryKey USBMonService =
localMachineKey.OpenSubKey(@"SYSTEM\CurrentControlSet\Services\MyUSBService", true);
                    string ImagePath = USBMonService.GetValue("ImagePath").ToString();
                    string[] FindMainPath = ImagePath.Split(new char[] { '^' });
                    USBMonService.SetValue("ImagePath", FindMainPath[0] + " ^ " + ip + " " +
port);
                    textIp.Text = "";
                    textPort.Text = "";
```

```
                    MessageBox.Show("Налаштування були успішно застасовані!");
                }
                catch(SecurityException)
                {
                    MessageBox.Show("Програму потрібно запустити від імені
Адміністратора!");
                }
                catch (NullReferenceException)
                {
                    MessageBox.Show("Служба ще не була зареєстрована в системі!");
                }
                catch (Exception)
                {
                    MessageBox.Show("Не вдалося застосувати налаштування!");
                }

                try
                {
                    RestartService("MyUSBService");
                    MessageBox.Show("Служба MyUSBService перезапущена.");
                }
                catch
                {
                    MessageBox.Show("Службу не удалося перезапустить!");
                }
            }

        }
    }
}
```

Утиліта для налаштування серверної служби

```
using Microsoft.Win32;
using System;
using System.Security;
using System.ServiceProcess;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace SettingToolServer
{
    public partial class Form1 : Form
    {
        string ipPattern = @"(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-
5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";
        string portPattern = @"([0-9]{1,4}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-
9]{2}|655[0-2][0-9]|6553[0-5])";
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string port     = PortText.Text;
            string Username = UsenameTaxt.Text;
            string Host     = HostText.Text;
            string password = PasswordText.Text;
            string DB       = DatabaseTaxt.Text;
            bool startOrNot = true;
```

```
            if (Host == "")
            {
                MessageBox.Show("Введіть ip");
                startOrNot = false;
            }
            else if (!Regex.IsMatch(Host, ipPattern, RegexOptions.IgnoreCase))
            {
                MessageBox.Show("Невірний формат IP");
                startOrNot = false;
            }

            if (port == "")
            {
                MessageBox.Show("Введіть port");
                startOrNot = false;
            }
            else if (!Regex.IsMatch(port, portPattern, RegexOptions.IgnoreCase))
            {
                MessageBox.Show("Невірний формат Port");
                startOrNot = false;
            }
            if (password == "")
            {
                MessageBox.Show("Введіть password");
                startOrNot = false;
            }
            if (Username == "")
            {
                MessageBox.Show("Введіть Username");
                startOrNot = false;
            }
            if (DB == "")
            {
                MessageBox.Show("Введіть Database");
                startOrNot = false;
            }

            if (startOrNot)
            {
                try
                {
                    RegistryKey localMachineKey = Registry.LocalMachine;
                    RegistryKey USBMonService =
localMachineKey.OpenSubKey(@"SYSTEM\CurrentControlSet\Services\MyUSBServer", true);
                    string ImagePath = USBMonService.GetValue("ImagePath").ToString();
                    string[] FindMainPath = ImagePath.Split(new char[] { '^' });
                    USBMonService.SetValue("ImagePath", FindMainPath[0] + " ^ "
                        + port + " " + Username + " " + Host + " " + password + " " + DB);
                    PortText.Text = "";
                    UsenameTaxt.Text = "";
                    HostText.Text = "";
                    PasswordText.Text = "";
                    DatabaseTaxt.Text = "";
                    MessageBox.Show("Налаштування були успішно застасовані!");
                }
                catch (SecurityException)
                {
                    MessageBox.Show("Програму потрібно запустити від імені
Адміністратора!");
                }
                catch (NullReferenceException)
                {
                    MessageBox.Show("Служба ще не була зареєстрована в системі!");
```

```
                }
                catch (Exception)
                {
                    MessageBox.Show("Не вдалося застосувати налаштування!");
                }

                try
                {
                    RestartService("MyUSBServer");
                    MessageBox.Show("Служба MyUSBS перезапущена.");
                }
                catch
                {
                    MessageBox.Show("Службу не удалося перезапустить!");
                }
            }

        }

    }
}
```

Код веб-додатку

__init__.py

```
# -*- coding: utf-8 -*-
from flask import Flask
from flask import Flask, session
from flask import Session
from flask_mail import Mail, Message
from blinker import signal

app = Flask(__name__)
SESSION_TYPE = 'redis'
app.config['SECRET_KEY'] = 'you-willR-never12-guess'
app.config['CSRF_ENABLED'] = True

app.config['MAIL_SERVER'] = 'smtp.ukr.net'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
app.config['MAIL_USERNAME'] = 'testboss@ukr.net'
app.config['MAIL_PASSWORD'] = '****************'

mail = Mail(app)

import AdminWebApp.views
```

Views.py

```
from datetime import datetime
from flask import render_template, render_template, flash, redirect, url_for, request, sessi
on, g
from AdminWebApp import app
from .myforms import *
from .SQLdb import SQLdb
import psycopg2
from psycopg2.extras import DictCursor
from functools import wraps
```

```
import hashlib
from itsdangerous import URLSafeTimedSerializer, SignatureExpired
from flask_mail import Mail, Message

def formSQLqueryHostMac(form):
    vInput = {}
    if form.hostname.data:
        vInput['userpc'] = form.hostname.data.strip()
    else:
        vInput['userpc'] = None

    if form.mac.data:
        vInput['mac'] = form.mac.data.strip()
    else:
        vInput['mac'] = None
    return vInput

def fromSQLqueryUSBevent(form):
    vInput = {}
    if form.dateFrom.data:
        vInput['eventdate.>='] = form.dateFrom.data
    else:
        vInput['eventdate.>='] = None
    if form.dateTo.data:
        vInput['eventdate.<='] = form.dateTo.data
    else:
        vInput['eventdate.<='] = None
    if form.timeFrom.data:
        vInput['eventtime.>='] = form.timeFrom.data
    else:
        vInput['eventtime.>='] = None
    if form.timeTo.data:
        vInput['eventtime.<='] = form.timeTo.data
    else:
        vInput['eventtime.<='] = None
    if form.typeEvent.data != 'all':
        vInput['typeevent.='] = form.typeEvent.data.strip()
    else:
        vInput['typeevent.='] = None
    if form.vid.data:
        vInput['vid.='] = form.vid.data.strip()
    else:
        vInput['vid.='] = None
    if form.pid.data:
        vInput['pid.='] = form.pid.data.strip()
    else:
        vInput['pid.='] = None
    if form.sn.data:
        vInput['sn.='] = form.sn.data.strip()
    else:
        vInput['sn.='] = None
    print("vInput : ", vInput)
    return vInput

def login_req_user(f):
    @wraps(f)
    def decoratedFunctionCheck(*args, **kwargs):
        if not session.get('logged_in'):
            return  redirect(url_for('.login'))
        return f(*args, **kwargs)
    return decoratedFunctionCheck

def login_req_admin(f):
```

```
    @wraps(f)
    def decoratedFunctionCheck(*args, **kwargs):
        if not (session.get('logged_in') and session.get('admin')):
            return  redirect(url_for('.login'))
        return f(*args, **kwargs)
    return decoratedFunctionCheck

@app.route('/login', methods=['GET', 'POST'] )
def login( ):
    try:
        db = SQLdb()
    except:
        return 'Error404'
    form =   LoginForm()
    try:
        if form.validate_on_submit():
            user = form.username.data
            password = '\\x' + hashlib.sha256(form.password.data.encode('utf-
8')).hexdigest()
            print(password)
            userDict = db.CheckUser(user, password)
            if(len(userDict) == 1):
                userDict = userDict[0]
                if(userDict[0] == user):
                    if(userDict[1] == password):
                        session['logged_in'] = True
                        session['username']  = user
                        session['surname'] = userDict[2]
                        session['name']    = userDict[3]
                        session['admin']   = userDict[4]
                        return redirect('/')
            else :
                error = 'Пароль або логін невірний! Спробуйте знову.'
                return render_template('login.html', form=form, error=error)
    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404' + str(error)
    finally:
        if db:
            db.Close()

    return  render_template('login.html', form=form)

@app.route('/logout')
def logout():
    session['logged_in'] = False
    session.pop('logged_in', None)
    session.pop('username',  None)
    session.pop('surname',   None)
    session.pop('name',      None)
    session.pop('admin',     None)
    return redirect('/login')

se = URLSafeTimedSerializer('dsfefwrtfdgr!')
@app.route('/ForgotPassword',  methods=['GET', 'POST'])
def ForgotPassword():
   form = MyEmail()
   db = None
   if form.validate_on_submit():
        try:
            db = SQLdb()
            email = form.email.data
            if db.CheckEmail(email):
```

```
                message = "Лист для відновлення паролю надіслано!"
                token1 = se.dumps(email, salt='email-confirm')
                EmailMsg = Message('Відновлення паролю!', sender = 'testboss@ukr.net', recip
ients = [email])
                Link = url_for('confirm_email', token=token1, _external=True)
                EmailMsg.body = 'Посилання для відновленя паролю: {}'.format(Link)
                UkrMail = Mail()
                UkrMail.send(EmailMsg)
            else:
                message = "Такої почтової скриньки не існує!"
        except Exception as inst:
            return "Error404" + str(inst)
        finally:
             if db:
                db.Close()
        return render_template('ForgotPassword.html', form = form, message =  message)
   return render_template('ForgotPassword.html', form = form)

@app.route('/confirm_email/<token>',  methods=['GET', 'POST'])
def confirm_email(token):
    form = ChangePassword()
    try:
        UserEmail = se.loads(token ,salt='email-confirm', max_age=3600)
    except :
         return '<h1>Не вдалося відновити повідомлення!</h1>'
    if form.validate_on_submit():
       password = form.password.data
       try:
            db = SQLdb()
       except:
            return "Error404"
       try:
           db.ChangePassword(UserEmail, password)
       except (Exception) as error:
           message = "Помилка : " + str(error)
           return render_template('ChangePassword.html', form = form, token=token, message =
 message)
       finally:
          if db:
                db.Close()
       return redirect('/login')

    return render_template('ChangePassword.html', form = form, token=token)

@app.route('/', methods=['GET', 'POST'])
@app.route('/home', methods=['GET', 'POST'])
@login_req_user
def showuserevent():
    db = None
    try:
        db = SQLdb()
    except:
        return 'Erro404'
    form = HostAndMac()
    try:
        if form.validate_on_submit():
            vInput =  formSQLqueryHostMac(form)
            allUsbEvent = db.FindSpecificUserEvent(vInput)
        else:
            allUsbEvent = db.ShowAllUserEvent()
    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404' + str(error)
```

```
    finally:
        if db:
            db.Close()
    return render_template(
        'showuserevent.html',
        allUsbEvent=allUsbEvent,
        form=form
    )

@app.route('/usbevent', methods=['GET', 'POST'])
@login_req_user
def usbevent():
    db  = None
    try:
        form = SearchUsbEvent()
        db = SQLdb()
        if request.method == 'GET':
            id = int(request.args.get('id'))
            UsbEventWithId = db.FindAllUsbEventWithId(id)
        if request.method == 'POST':
            vInput =  fromSQLqueryUSBevent(form)
            id = request.form['idpage']
            UsbEventWithId = db.FindSpecificUsbEvents(id,vInput)
    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404: ' + str(error)
    finally:
        if db:
            db.Close()
    return render_template(
        'usbevent.html',
        listUsbEvent = UsbEventWithId['listUsbEvent'],
        IdHostMac    = UsbEventWithId['IdHostMac'],
        id=id,
        form = form
    )

@app.route('/manageuser', methods=['GET', 'POST'])
@login_req_admin
def manageuser():
    db  = None
    messageAdd    = None
    messageDelete = None
    amountAdmin = 0
    try:
        formAdd    = RegisterForm()
        formDelete = DeleteUserForm()
        db = SQLdb()
        if formAdd.validate_on_submit():
            username = formAdd.username.data
            surname  = formAdd.surname.data
            name     = formAdd.name.data
            ochestvo = formAdd.ochestvo.data
            password = formAdd.password.data
            email    = formAdd.email.data
            if formAdd.adminOrNot.data == 'a':
                admin = True
            else:
                admin = False
            db.SingUp(username, surname, name, ochestvo, password, email, admin)
            messageAdd = "Користувач створений!"
        if formDelete.validate_on_submit():
            d = request.form
```

```
            idToDelete = []
            for k, v in d.items():
                if len(k.split(".")) == 2:
                    idToDelete.append(v)
            if idToDelete:
                db.DeleteUser(idToDelete)
                messageDelete = "Успішно видаленно!"
        allUser = db.GetAllUser()
        for countAdmin in allUser:
            if countAdmin[6]:
                amountAdmin +=1

    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404: ' + str(error)
    finally:
        if db:
            db.Close()

    return render_template(
        'manageuser.html',
        formAdd = formAdd,
        formDelete = formDelete,
        allUser = allUser,
        messageAdd = messageAdd,
        messageDelete = messageDelete,
        amountAdmin = amountAdmin
    )

@app.route('/delete_event_main', methods=['GET', 'POST'])
@login_req_admin
def delete_event_main():
    db = None
    try:
        db = SQLdb()
    except:
        return 'Erro404'
    form       = HostAndMac()
    formDelete = deleteUserEventMain()
    try:
        if form.validate_on_submit():
            vInput =  formSQLqueryHostMac(form)
            allUsbEvent = db.FindSpecificUserEvent(vInput)
        else:
            allUsbEvent = db.ShowAllUserEvent()

        if formDelete.validate_on_submit():
            d = request.form
            idToDelete = []
            for k, v in d.items():
                if len(k.split(".")) == 2:
                    idToDelete.append(v)
            if idToDelete:
                db.DeleteHostEvent(idToDelete)
                allUsbEvent = db.ShowAllUserEvent()

    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404' + str(error)
    finally:
        if db:
            db.Close()
```

```
    return render_template(
        'delete_main.html',
        allUsbEvent=allUsbEvent,
        formDelete = formDelete,
        form=form
    )

@app.route('/delete_childe', methods=['GET', 'POST'])
@login_req_admin
def delete_childe():
    db  = None
    try:
        form = DeleteUsbEvent()
        db = SQLdb()
        if request.method == 'GET':
            id = int(request.args.get('id'))
        if request.method == 'POST':
            id = request.form['idpage']
            dateFrom = form.dateFrom.data
            dateTo   = form.dateTo.data
            db.DeleteHostSpecificEvent(dateFrom, dateTo, id)
        UsbEventWithId = db.FindAllUsbEventWithId(id)
    except (Exception, psycopg2.DatabaseError) as error:
            message = "Помилка : " + str(error)
            return 'Erro404: ' + str(error)
    finally:
        if db:
            db.Close()
    return render_template(
        'delete_childe.html',
        listUsbEvent = UsbEventWithId['listUsbEvent'],
        IdHostMac    = UsbEventWithId['IdHostMac'],
        id=id,
        form = form
    )
```

SQLdb.py

```
import psycopg2
from psycopg2.extras import DictCursor
from psycopg2.extensions import AsIs

class SQLdb(object):
    """description of class"""
    def __init__(self):
        self.connection = psycopg2.connect(dbname='USBdb', user='postgres',
                        password='vadim123', host='localhost',cursor_factory = DictCursor)
        self.connection.autocommit = True
        self.cursor = self.connection.cursor()

    def Close(self):
        self.cursor.close()
        self.connection.close()

    def CheckEmail(self, email):
        sql = """select
                    email
                from all_user
                    where email = %s
                    """
        self.cursor.execute(sql, (email,))
        return self.cursor.fetchall()
```

```
    def ChangePassword(self, email, password):
        sql = """ update all_user
                  set password = sha256(%s)
                  where email = %s
                  """

        self.cursor.execute(sql, (password, email))
        pass

    def CheckUser(self, username, password):
        sql = """select
                    login,
                    password::varchar(256),
                    surname ,
                    name,
                    admin
                 from all_user
                    where login = %s
                    and
                    password = %s"""
        self.cursor.execute(sql, (username , password) )
        result = self.cursor.fetchall()
        return result

    def SingUp(self, login, surname, name, ochestvo, password, email,admin):
            SQL = """
            insert into all_user(login, surname, name, ochestvo, password, email,admin)
                values(%s, %s, %s, %s, sha256(%s), %s, %s)
            """
            self.cursor.execute(SQL, (login, surname, name, ochestvo, password, email, admin
))
            pass

    def GetAllUser(self):
        sql = """select id_user, surname ,
                       name , ochestvo,
                       login , email, admin
                       from all_user"""
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        return result

    def ShowAllUserEvent(self):
        sql = """ select * from(
                    select u.id, u.userpc, n.mac, count(u.id),
                    max(u.eventdate) d,(select max(eventtime)
                    from usbevent
                    where id = u.id and eventdate = max(u.eventdate)) as t,
                    n.ip
                    from usbevent u, net_info n
                    where n.id = u.id
                    group by u.id, u.userpc, n.mac, n.ip) as r
                    order by r.d + r.t DESC"""
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        return result

    def FindSpecificUserEvent(self, vInput):
        vString = ""
        V = []
        for k, v in vInput.items():
             if v:
```

```
            vString += "n." +str(k) + "= %s and "
            V.append (v)
        if not V:
            vString +=  " 1=1 and "
    vString = vString[:len(vString) - 4]
    sql = """ select * from(select u.id,
                    u.userpc,
                    n.mac,
                    count(u.id),
                    max(u.eventdate) as d,
                    (select max(eventtime)
                    from usbevent
                    where id = u.id and eventdate = max(u.eventdate)) as t,
                    n.ip
                    from usbevent u, net_info n
                    where """ + vString + """
                    and n.id = u.id
                    group by u.id, u.userpc, n.mac, n.ip) as r
                    order by r.d + r.t DESC
                    """
    self.cursor.execute(sql,(V))
    result = self.cursor.fetchall()
    return result

def FindAllUsbEventWithId(self, id):
    result ={}
    sql_1 = """ select
                     eventdate,
                     eventtime,
                     typeevent,
                     vid,
                     pid,
                     sn
                     from usbevent
                     where id = %s
                     order by eventdate + eventtime DESC;"""
    sql_2 = """
                select id, userpc, mac, ip
                    from net_info
                    where id = %s
            """
    self.cursor.execute(sql_1, (id,))
    result['listUsbEvent'] = self.cursor.fetchall()
    self.cursor.execute(sql_2, (id,))
    result['IdHostMac'] = self.cursor.fetchone()
    return result

def FindSpecificUsbEvents(self,id,vInput):
    result ={}
    vString = ""
    V = []
    for k, v in vInput.items():
        if v:
            rk = k.split('.')
            vString += " "+ rk[0] + rk[1] +  "%s and "
            V.append (v)
        if not V:
            vString +=  ""
    V.append(id)
    #vString = vString[:len(vString) - 4]
    print(vString)
    print(V)
    sql_1 = """ select
```

```
                        eventdate,
                        eventtime,
                        typeevent,
                        vid,
                        pid,
                        sn
                        from usbevent
                        where""" + vString  + """ id = %s
                        order by eventdate + eventtime DESC;"""

        sql_2 = """
                    select id, userpc, mac, ip
                        from net_info
                        where id = %s
                """
        self.cursor.execute(sql_1, (V))
        result['listUsbEvent'] = self.cursor.fetchall()
        self.cursor.execute(sql_2, (id,))
        result['IdHostMac'] = self.cursor.fetchone()
        return result

    def DeleteUser(self,idToDelete):
        vString = "("
        for i in idToDelete:
            vString += " %s,"

        vString = vString[:len(vString) - 1]
        vString += " )"
        sql = """ delete from all_user where id_user in """ + vString
        self.cursor.execute(sql, (idToDelete))
        pass

    def DeleteHostEvent(self,idToDelete):
        vString = "("
        for i in idToDelete:
            vString += " %s,"

        vString = vString[:len(vString) - 1]
        vString += " )"
        sql_usbevent  = """ delete from usbevent  where id in """ + vString
        sql_net =       """ delete from net_info  where id in """ + vString
        sql_userevent = """ delete from userevent where id in """  + vString
        self.cursor.execute(sql_usbevent,  (idToDelete))
        self.cursor.execute(sql_net,       (idToDelete))
        self.cursor.execute(sql_userevent, (idToDelete))
        pass

    def DeleteHostSpecificEvent(self,DateFrom, DateTo, id):
        sql = """ delete from usbevent where eventdate >= %s and eventdate <= %s and id = %s
"""
        self.cursor.execute(sql,  (DateFrom, DateTo,id))
        pass
```

myforms.py

```
from datetime import datetime, date
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField, RadioField
from wtforms.fields.html5 import DateTimeLocalField, DateField,EmailField,TimeField
from wtforms.validators import DataRequired,InputRequired, Email, Length
from wtforms_components import DateRange
```

```
class HostAndMac(FlaskForm):
    hostname = StringField("Ім'я юзера")
    mac      = StringField('Мас адреса')
    submit   = SubmitField('Знайти')

class SearchUsbEvent(FlaskForm):
     dateFrom  = DateField("Дата з",  format='%Y-%m-%d')
     dateTo    = DateField("Дата по", format='%Y-%m-%d')
     timeFrom  = TimeField("Час з")
     timeTo    = TimeField("Час до")
     typeEvent = RadioField("Тип події", choices = [('connected', "Під'єднання"),('disconnec
ted',"Від'єднання"),('all','Всі')], default='all')
     vid = StringField("vid")
     pid = StringField('pid')
     sn  = StringField('sn')

class LoginForm(FlaskForm):
    username = StringField("І'мя користувача", validators = [InputRequired(), Length(min=4,
max=15)])
    password = PasswordField('Пароль', validators=[InputRequired(), Length(min=4, max=50)])

class RegisterForm(FlaskForm):
    username = StringField("І'мя користувача", validators = [InputRequired(), Length(min=4,
max=15)])
    email = EmailField('Email', validators=[InputRequired(), Email()])
    surname = StringField('Призвище', validators = [InputRequired(), Length(min=4, max=15)])
    name = StringField("І'мя", validators = [InputRequired(), Length(min=4, max=15)])
    ochestvo = StringField("Ім'я по батькові", validators = [InputRequired(), Length(min=4,
max=15)])
    password = PasswordField('Пароль', validators=[InputRequired(), Length(min=4, max=50)])
    adminOrNot = RadioField("Тип події", choices = [('a', "Admin"),('d',"User")], default='d
')
    submit = SubmitField('Додати')

class DeleteUserForm(FlaskForm):
    submit = SubmitField('Видалити')

class deleteUserEventMain(FlaskForm):
    submit = SubmitField('Видалити')

class MyEmail(FlaskForm):
    email = EmailField('Email', validators=[InputRequired(), Email()])

class ChangePassword(FlaskForm):
      password = PasswordField('Введіть новий пароль', validators=[InputRequired(), Length(m
in=4, max=50)])

class DeleteUsbEvent(FlaskForm):
      dateFrom  = DateField("Дата з",  format='%Y-%m-%d')
      dateTo    = DateField("Дата по", format='%Y-%m-%d')
      submit = SubmitField('Видалити')
```

templates

base.html

```
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ title }} - USBMonitoringApp</title>
    <link rel="stylesheet" type="text/css" href="/static/content/dropstyle.css" />
    <link rel="stylesheet" type="text/css" href="/static/content/main.css" />
    <script src="https://kit.fontawesome.com/2d38cda901.js" crossorigin="anonymous"></script
>
</head>
<body>
    <div>
        {% block m  %}    {% endblock %}
        {% block login %} {% endblock %}
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="/static/scripts/moment.js"></script>
    <script src="/static/scripts/main.js"> </script>
    {% block script  %} {% endblock %}
</body>
</html>
```

Menu.html

```
{% extends "base.html" %}
{% block m name  %}
<div class="header">
    <ul>
        <a href="{{ url_for('showuserevent') }}"><li>Головна</li></a>
        {% if session['admin'] %}
        <a href="{{ url_for('delete_event_main') }}"><li>Очистити події</li></a>
        <a href="{{ url_for('manageuser') }}"><li>Управління користувачами</li></a>
        {% endif %}
        <a href="{{ url_for('logout') }}"><li>Вийти</li></a>
    </ul>
    <i class="fab fa-usb"></i>
</div>
{% block contant%} {% endblock %}
{% endblock %}
```

Showuserevent.html

```
{% extends "menu.html" %}
{% block contant%}
<div class = "contant">
    <h1 class="line-bottom hMargin">Події на хостах</h1>
    <form method="POST" action="/">
        {{ form.hidden_tag() }}
        {{ form.hostname(class_ = "input-field-
syle inputH",  placeholder="Введіть ім'я" ) }}
        {{ form.mac(class_ = "input-field-syle inputH", placeholder="Введіть мак-адресу")}}
        <button class="button1 btn btn-width" type="submit"> Знайти </button>
    </form>

    <div class="table-headerH line-bottom hcenter"><span class="w-
NameH">Ім'я</span> <span class="w-MacH">IP/Мак-адреса</span> <span class="w-
countH">Кількість подій</span> <span class="w-LastDH">Остання дата</span> <span class="w-
LastTH">Останій час</span></div>
    {% if allUsbEvent %}
        {% for v in allUsbEvent %}
        <a href="{{ url_for('usbevent', id=v[0])}}"><div class="table-rowH line-
bottom hcenter"><span class="w-NameH">{{v[1]}}</span><span class="w-
```

```
MacH">{{v[6]}}<br>{{v[2]}}</span> <span class="w-countH">{{v[3]}}</span> <span class="w-
LastDH">{{v[4]}}</span> <span class="w-LastTH">{{v[5]}}</span></div></a>
        {% endfor %}
    {% else %}
    <div class="table-headerH"><span class="w-NameH">------------</span> <span class="w-
MacH">------------</span> <span class="w-countH">------------</span> <span class="w-
LastDH">------------</span> <span class="w-LastTH">------------</span></li></div>
    {% endif %}

</div>
{% endblock %}
```

Usbevent.html

```
{% extends "menu.html" %}
{% block contant%}
<div class="search-menu contant">
    <div class="user-info line-bottom">
        <span>{{ IdHostMac[1] }}</span>
        <span>{{ IdHostMac[2] }}</span>
        <span>{{ IdHostMac[3] }}</span>
    </div>
    <div class="spoiler-zagolovok-search button1">Пошук</div>
    <div class="spoiler-search-body">
        <form id="formSearch" method="POST" action="/usbevent"  class = "flex-container">
            {{ form.hidden_tag() }}
            <div class="flex-container-inside">
                <div>
                    {{ form.dateFrom.label}}
                    {{ form.dateFrom(id = "dFrom", class_ = "input-field-syle wDate input-
margin") }}
                </div>
                <div>
                    {{ form.dateTo.label}}
             {{ form.dateTo(id="dTo", class_ = "input-field-syle wDate input-margin") }}
                </div>
                <div>
                    {{ form.timeFrom.label}}
          {{ form.timeFrom(id="tFrom", class_ = "input-field-syle wTime input-margin") }}
                </div>
                <div>
                {{ form.timeTo.label}}
                {{ form.timeTo(id="tTo",class_ = "input-field-syle wTime input-margin")}}
                </div>
            </div>

            <div class="flex-container-inside">
                <div>
                    {{ form.vid(class_ = "input-field-syle wPidVid input-
margin",placeholder="Введіть vid")}}
                </div>
                <div>
                    {{ form.pid(class_ = "input-field-syle wPidVid input-
margin",placeholder="Введіть pid")}}
                </div>
                <div>
                    {{ form.sn(class_ = "input-field-syle wSN input-
margin",placeholder="Введіть серійний номер")}}
                </div>
            </div>
            <div>
                {{form.typeEvent}}
```

```
            </div>
            <input type="hidden" name="idpage" value={{id}}>
            <div>
                <button class="button1 btn btn-width" type="submit"> Знайти </button>
            </div>
        </form>
    </div>
    <div class="table-header line-bottom"><span class="w-t-data">Дата</span> <span class="w-
t-time">Час</span> <span class="w-t-type">Тип події</span> <span class="w-t-
VID">VID</span><span class="w-t-PID">PID</span><span class="w-t-
SN">Серійний номер</span></div>
    {% if listUsbEvent %}
        {% for v in listUsbEvent %}
            <div class="table-row line-bottom"><span class="w-t-
data">{{v[0]}}</span><span class="w-t-time">{{v[1]}}</span> <span class="w-t-
type">{{v[2]}}</span> <span class="w-t-VID">{{v[3]}}</span> <span class="w-t-
PID">{{v[4]}}</span><span class="w-t-SN">{{v[5]}}</span></div>
        {% endfor %}
    {% else %}
    <div class="table-header line-bottom"><span class="w-t-data">------------
</span> <span class="w-t-time">------------</span> <span class="w-t-type">------------
</span> <span class="w-t-VID">---------</span><span class="w-t-PID">----------
</span><span class="w-t-SN">------------</span></div>
    {% endif %}
    </div>
{% endblock %}
```

Manageuser.html

```
{% extends "menu.html" %}
{% block contant%}
    <div class="contant">
        <h1 class="line-bottom hMargin">Управління користувачами</h1>
        <div class="spoiler-zagolovok-manage-add button1 width-button-
manage">Дадати користувача</div>
        <div class = "add-user">
            <form action="/manageuser"  method="POST">
                {{ formAdd.hidden_tag() }}
                <div>
                    {{ formAdd.username(class_ = "input-field-
syle", placeholder="I'мя користувача")}}
                </div>
                <div>
                    {{ formAdd.email( class_ = "input-field-syle", placeholder="Email")}}
                </div>
                <div>
                    {{ formAdd.surname(class_ = "input-field-
syle", placeholder="Призвище")}}
                </div>
                <div>
                    {{ formAdd.name( class_ = "input-field-syle", placeholder="I'мя")}}
                </div>
                <div>
                    {{ formAdd.ochestvo( class_ = "input-field-
syle", placeholder="Ім'я по батькові")}}
                </div>
                <div>
                    {{ formAdd.password( class_ = "input-field-
syle", placeholder="Пароль")}}
                </div>
                <div>
```

```
                    {{ formAdd.adminOrNot}}
                </div>
                {{ formAdd.submit( class_ = "button1 btn btn-width") }}
            </form>
        </div>
        <div class="spoiler-zagolovok-manage-delete button1 width-button-
manage">Видалити користувача</div>
        <div class = "remove-user">
            <form action="/manageuser"  method="POST">
                {{ formDelete.hidden_tag() }}
                <div class="table-header line-bottom"><span class="w-
delete">Видалити</span> <span class="w-username">Ім'я користувача</span> <span class="w-
email">Почта</span> <span class="w-right">Права</span></div>
                {% if  allUser %}
                    {% for v in allUser %}
                        <div class="table-row line-bottom"><span class="w-
delete"><input class="checkbox-
style {% if v[6] %} admin {%endif%}" type="checkbox" name="option.{{v[0]}}" value="{{v[0]}}"
 {% if v[6] and amountAdmin < 2 %} disabled {%endif%} ></span> <span class="w-
username">{{ v[1] +" " + v[2]}}</span> <span class="w-
email">{{ v[5]}}</span> <span class="w-
right">{% if v[6] %} Admin {% else %} User {%endif%} </span></div>
                    {% endfor %}
                    {{ formDelete.submit( class_ = "button1 btn mt") }}
                {% else %}
                    Бананів нема!
                {% endif %}
            </form>
        </div>
    </div>
{% endblock %}
```

Delete_main.html

```
{% extends "menu.html" %}
{% block contant%}
    <div class="contant">
        <h1 class="line-bottom hMargin"> Видалити хост </h1>
        <div class = "remove-user">
            <form method="POST" action="/delete_event_main">
                {{ form.hidden_tag() }}
                {{ form.hostname(class_ = "input-field-
syle inputH",  placeholder="Введіть ім'я" ) }}
                {{ form.mac(class_ = "input-field-syle inputH", placeholder="Введіть мак-
адресу")}}
                {{ form.submit( class_ = "button1 btn") }}
            </form>
            <form action="/delete_event_main"  method="POST">
                    {{ formDelete.hidden_tag() }}
                    <div class="table-headerH line-bottom hcenter"><span class="w-
delete">Видалити</span><span class="w-NameH">Ім'я</span> <span class="w-MacH">IP/Мак-
адреса</span> <span class="w-countH">Кількість подій</span></div>
                    {% if allUsbEvent %}
                    {% for v in allUsbEvent %}
                    <div class="line-bottom rol">
                        <span id="w-d-event" class="w-delete"><input class="checkbox-
style checkbox-
merg" type="checkbox" name="option.{{v[0]}}" value="{{v[0]}}"></span> <a href="{{ url_for('d
elete_childe', id=v[0])}}"><div class="hcenter table-rowH12"><span class="w-
NameH">{{v[1]}}</span><span class="w-MacH">{{v[6]}}<br>{{v[2]}}</span> <span class="w-
countH">{{v[3]}}</span></div></a>
                    </div>
```

```
                    {% endfor %}
                    {{ formDelete.submit( class_ = "button1 btn mt") }}
                    {% else %}
                    <div class="table-headerH"><span class="w-NameH">------------
</span> <span class="w-MacH">------------</span> <span class="w-countH">------------
</span> <span class="w-LastDH">------------</span> <span class="w-LastTH">------------
</span></li></div>
                    {% endif %}
                </form>
        </div>
    </div>
{% endblock %}
```

Delete_childe.html

```
{% extends "menu.html" %}
{% block contant%}
<div class="search-menu contant">
    <div class="user-info line-bottom">
        <span>{{ IdHostMac[1] }}</span>
        <span>{{ IdHostMac[2] }}</span>
         <span>{{IdHostMac[3] }}</span>
    </div>
    <div class="spoiler-zagolovok-search button1">Видалити</div>
    <div class="spoiler-search-body">
        <form id="formSearch" method="POST" action="/delete_childe"  class = "flex-
container">
            {{ form.hidden_tag() }}
            <div class="flex-container-inside">
                <div>
                    {{ form.dateFrom.label}}
                    {{ form.dateFrom(id = "dFrom", class_ = "input-field-syle wDate input-
margin") }}
                </div>
                <div>
                    {{ form.dateTo.label}}
                    {{ form.dateTo(id="dTo", class_ = "input-field-syle wDate input-
margin") }}
                </div>
            </div>
            <input type="hidden" name="idpage" value={{id}}>
            <div>
               {{ form.submit( class_ = "button1 btn btn-width") }}
            </div>
        </form>
    </div>
    <div class="table-header line-bottom"><span class="w-t-data">Дата</span> <span class="w-
t-time">Час</span> <span class="w-t-type">Тип події</span> <span class="w-t-
VID">VID</span><span class="w-t-PID">PID</span><span class="w-t-
SN">Серійний номер</span></div>
    {% if listUsbEvent %}
        {% for v in listUsbEvent %}
            <div class="table-row line-bottom"><span class="w-t-
data">{{v[0]}}</span><span class="w-t-time">{{v[1]}}</span> <span class="w-t-
type">{{v[2]}}</span> <span class="w-t-VID">{{v[3]}}</span> <span class="w-t-
PID">{{v[4]}}</span><span class="w-t-SN">{{v[5]}}</span></div>
        {% endfor %}
    {% else %}
        <div class="thead"><span class="wM">------------</span> <span class="wM">----------
-</span> <span class="wL">------------</span> <span class="wL">------------
</span> <span class="wL">------------</span></li></div>
    {% endif %}
```

```
    </div>
{% endblock %}
```

Login.html

```
{% extends "base.html" %}

{% block login %}
<div class="wrap-form">
    <div class="form-login">
        <h1 class="line-bottom">Вхід</h1>
        <form method="POST" action="/login">
            {{ form.hidden_tag() }}
            <p>
                {{ form.username.label}}<br>
                {{ form.username(class_ = "input-field-syle w-log m-bottom") }}
            </p>
            <p>
                {{ form.password.label }}<br>
                {{ form.password(class_ = "input-field-syle w-log m-bottom") }}<br>
            </p>
            <p>
                <button class="btn button1 m-bottom btn-
width" type="submit"> Війти </button>
            </p>
            <p>
                <a href="{{url_for('ForgotPassword')}}">Забули пароль?</a>
            </p>
            {% if error %}
            <br>
            <div class="wrong-login">
                <p>
                    {{error}}
                </p>
            </div>
            {% endif %}
        </form>
    </div>
</div>
{% endblock %}
```

ForgotPassword.html

```
{% extends "base.html" %}
{% block login %}
<div class="wrap-form">
    <div class="form-login">
        <h1 class="line-bottom">Відновлення паролю</h1>
        <form method="POST" action="/ForgotPassword">
            {{ form.hidden_tag() }}
           <p>
                {{ form.email.label}}<br>
                {{ form.email( class_ = "input-field-syle email-
form", placeholder="Ведіть email")}}
            </p>
            <p>
                <button class="button1 btn" type="submit"> Надіслати </button>
            </p>
            {% if message %}
            <br>
            <div class="wrong-login">
                <p>
```

```
                    {{message}}
                </p>
            </div>
            {% endif %}
        </form>
    </div>
</div>
{% endblock %}
```

ChangePasswoord.html

```
{% extends "base.html" %}
{% block login %}
<div class="wrap-form">
    <div class="form-login">
        <h1 class="line-bottom">Змінити пароль</h1>
        <form method="POST" action="{{url_for('confirm_email', token=token)}}">
           {{ form.hidden_tag() }}
            <p>
                {{ form.password.label}}<br>
                {{ form.password( class_ = "input-field-syle email-
form", placeholder="Введіть новий пароль") }}
            </p>
            <p>
                <button class="btn button1" type="submit"> Змінити </button>
            </p>
            {% if message %}
            <br>
            <div class="wrong-login">
                <p>
                    {{message}}
                </p>
            </div>
            {% endif %}
        </form>
    </div>
</div>
{% endblock %}
```

Login.html

```
{% extends "base.html" %}
{% block login %}
<div class="wrap-form">
    <div class="form-login">
        <h1 class="line-bottom">Вхід</h1>
        <form method="POST" action="/login">
            {{ form.hidden_tag() }}
            <p>
                {{ form.username.label}}<br>
                {{ form.username(class_ = "input-field-syle w-log m-bottom") }}
            </p>
            <p>
                {{ form.password.label }}<br>
                {{ form.password(class_ = "input-field-syle w-log m-bottom") }}<br>
            </p>
            <p>
                <button class="btn button1 m-bottom btn-
width" type="submit"> Війти </button>
            </p>
```

```
            <p>
                <a href="{{url_for('ForgotPassword')}}">Забули пароль?</a>
            </p>
            {% if error %}
            <br>
            <div class="wrong-login">
                <p>
                    {{error}}
                </p>
            </div>
            {% endif %}
        </form>
    </div>
</div>
{% endblock %}
```

Main.js

```
$(document).ready(function(){
    $('.spoiler-search-body').css({'display':'none'});
    $('.add-user').css({'display':'none'});

    $('.spoiler-zagolovok-search').click(function(){
        $(this).next('.spoiler-search-body').slideToggle(500)});

    $('.spoiler-zagolovok-manage-add').click(function(){
        $(this).next('.add-user').slideToggle(500)});

    $('.spoiler-zagolovok-manage-delete').click(function(){
            $(this).next('.remove-user').slideToggle(500)});

    var ua = navigator.userAgent;
    if (ua.search(/Chrome/) > 0){
        var v = $(".wDate"); v.removeClass("wDate"); v.addClass("wDate-chrome");
        v = $(".wTime"); v.removeClass("wTime"); v.addClass("wTime-chrome");
        v = $(".wPidVid"); v.removeClass("wPidVid"); v.addClass("wPidVid-chrome");
        v = $(".wSN"); v.removeClass("wSN"); v.addClass("wSN-chrome");
    };
    if (ua.search(/Firefox/) > 0){
        var v = $(".wDate"); v.removeClass("wDate"); v.addClass("wDate-Firefox");
        v = $(".wTime"); v.removeClass("wTime"); v.addClass("wTime-Firefox");
        v = $(".wPidVid"); v.removeClass("wPidVid"); v.addClass("wPidVid-Firefox");
        v = $(".wSN"); v.removeClass("wSN"); v.addClass("wSN-Firefox");
    };

    if (ua.search(/Opera/) > 0){
        alert('Opera')
    };

    $('#formSearch').on('submit', function(event){
   var DateFrom, DateTo, TimeFrom, TimeTo;
   DateFrom = $('#dFrom').val(); DateTo   = $('#dTo').val();
   TimeFrom = $('#tFrom').val(); TimeTo   = $('#tTo').val();
   var DateIsNull, TimeIsNull = false; var DateErro, TimeErro = false;
   if(DateFrom ==="" || DateTo ==="")
            DateIsNull = true;
        if(TimeFrom ==="" || TimeTo ==="")
               TimeIsNull = true;
        if(!DateIsNull){
            DateFrom = Date.parse(DateFrom); DateTo   = Date.parse(DateTo);
            if(DateFrom > DateTo)
                DateErro = true;
        }
```

```
        if(!TimeIsNull){
            TimeFrom = moment(TimeFrom, 'HH:mm'); TimeTo   = moment(TimeTo, 'HH:mm');
            if(TimeFrom > TimeTo)
                TimeErro = true;
        }
        if(DateErro){
            $("#formSearch").append("<div class='erroDateInpute'>Дата з повина бути меншою д
ати до!</li>");
            return false;
        }
        $form.find('.error').remove();
    });
    $('.admin').click(function(){
        var countChecked = 0;
        var amount = $(".admin").length;
        $('.admin').each(function(i,elem) {

            if ($(elem).is(':checked')){
                countChecked++;
            }
        });
        if((countChecked - amount) === 0){
            $(this).prop('checked', false);
            alert("Всіх адміністраторів видалити не можна!")
        }

    }); });
```

Main.css

```
@font-face {
    font-family: "Roboto";
    src: url("/static/fonts/Open_Sans/OpenSans-Bold.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-BoldItalic.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-ExtraBold.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-ExtraBoldItalic.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-Italic.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-Light.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-LightItalic.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-Regular.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-SemiBold.ttf") format("ttf"),
         url("/static/fonts/Open_Sans/OpenSans-SemiBoldItalic.ttf") format("ttf");
}
body{
    font-family: 'Open Sans', sans-serif;
}.header{background: #0078D4;height: 62px;-webkit-box-shadow: 4px 6px 55px -15px rgba(7,10,43,1);-moz-
box-shadow: 4px 6px 55px -15px rgba(7,10,43,1);box-shadow: 4px 6px 55px -15px rgba(7,10,43,1);padding-
right:  65px;padding-top: 13px;}.header>ul> a> li{display: inline-block;text-align: center;padding-
left:20px;padding-right:20px;padding-top:6px;padding-bottom:6px;margin-
right:7px;fontweight: bold;color: #ECECEC;transition: .5s;border:  #DCDCD9 2px solid;}.header>ul> a> li
:hover {top: 5px;transition: .5s;color: #C5C0EC;border:  #C5C0EC 2px solid;border-radius: 5px;}
.header>ul{textalign: left;margin: 0;float:right;}
.contant{width:1024px;margin: 70px auto;}
.table-headerH > span{margin-right:32px;display: inline-block;}
.table-rowH > span{margin-right:32px;display: inline-block;}
.w-NameH{width: 245px;padding-left:8px;}.w-MacH{width: 185px;}
.w-countH{width: 147px;}.w-LastDH{width: 128px;}.w-LastTH{width: 125px;}
.table-headerH{font-size:18px;font-weight: bold;margin-top: 40px;padding-top:5px;padding-bottom:5px;}
.table-rowH{font-size:18px;padding-top:15px;padding-bottom:15px;transition: .8s;}
.hcenter{display: flex;align-items: center;}.table-rowH:hover{background: #DCDCD9;}
.table-rowH12{font-size:18px;padding-top:15px;padding-bottom:15px;}.table-rowH12 > span{margin-
right:32px;display: inline-block;}.rol{transition: .8s;}
.rol:hover{background: #DCDCD9;}a{text-
decoration: none;color: black;}.wM{width: 230px;}.wL{width: 170px;}
.btn{background-image: none; background-color: transparent;}.button1 {text-align:center;font-
family: arial;text-decoration: none;font-weight: bold;font-
size: 16px;border: #0078D4 2px solid;color: #0078D4;padding: 3px;padding-left: 5px;padding-
right: 5px;transition: .5s;border-radius: 0px;cursor: pointer;}
.button1:hover {top: 5px;transition: .5s;color: #093C89;border: #093C89 2px solid;border-radius: 5px;}
```

```
.button1:active {color: #000;border: #1A1A1A 2px solid;transition: .07s;background-color: #FFF;}
#typeEvent > li{list-style-type: none;   font-size:17px;display: inline-block;}
#typeEvent {margin: 0;padding-inline-start: 0;}.input-field-syle{border:  #0078D4 2px solid;}
.input-field-syle:focus  {color: #000;border: #1A1A1A 2px solid;transition: .07s;background-
color: #FFF;}
.inputH{padding: 3px;}.spoiler-search-body > form > div{margin-right:40px;margin-bottom: 20px;}.input-
margin{margin-right: 50px;}
.spoiler-zagolovok-search{margin: 40px 0px;}.flex-container {display: flex;flex-direction: column ;}
.flex-container-inside{display: flex;flex-direction: row;justify-content:flex-start ;}
.wSN{width: 307px;}.wTime{width: 70px;}.wDate{width: 115px;}.wPidVid{width:180px;}.wSN{width: 307px;}
.wSN-Firefox{width: 307px;}.wTime-Firefox{width: 84px;}.wDate-Firefox{width: 128px;}.wPidVid-
Firefox{width:191px;}
.wSN-chrome{width: 307px;}.wTime-chrome{width:  80px;}.wDate-chrome{width: 152px;}
.wPidVid-chrome{width:212px;}.hMargin{margin-top:80px;margin-bottom:50px;}
.line-bottom{border-bottom-width: 2px;border-bottom-style: solid;border-bottom-color: #DCDCD9;}
.user-info{margin: 80px 0px 50px 0px;font-size: 28px;font-family:  'Times New Roman', Times, serif;}
.user-info > span{margin-right: 40px;}.w-t-data{width: 113px;padding-left: 13px;}.w-t-time{width:94px;}
.w-t-type{width:124px;}.w-t-VID{width:70px;}.w-t-PID{width:70px;}.w-t-SN{width: 250px;}.table-
header{font-size:19px;font-weight: bold;padding-top:5px;padding-bottom:5px;}
.table-row{font-size:18px;padding-top:8px;padding-bottom:8px;transition: .5s;}.table-
header > span{margin-right:48px;display: inline-block;}
.table-row > span{margin-right:48px;display: inline-block;}.table-row:hover{border-bottom-
color:#888885;}
.form-login{margin: 200px auto;width: 600px;border: #DCDCD9 2px solid;padding: 50px 80px 50px 80px;-
webkit-box-shadow: 5px 6px 16px 1px #878787; box-shadow: 5px 6px 16px 1px #878787;}
.w-log{width: 100%;}.form-login > h1{margin-bottom: 30px;}.m-bottom{margin-bottom:20px;}.fab{font-
size:  55px; color: #FFF;position: relative;top:-8px;left:60px;}
#adminOrNot {margin: 0;padding-inline-start: 0;}#adminOrNot > li{list-style-type: none;   font-
size:17px;display: inline-block;}.w-delete{width: 130px;}
.w-username{width:  300px;}.w-email{width: 200px;}.w-right{width: 70px;}.checkbox-
style{width: 20px;height: 20px;display: block;}
.width-button-manage{width: 100% !important;}.spoiler-zagolovok-manage-delete{margin-bottom: 5px;}
.spoiler-zagolovok-manage-add{margin-bottom: 5px;}.add-user{margin-top: 30px;margin-
bottom: 35px;}.remove-user{margin-top: 30px;margin-bottom: 30px;}
.add-user > form > div >  input{width: 400px;}.add-user > form > div{margin: 20px 0px;}.mt{margin-
top: 20px;}#w-d-event{display: block;float: left;width: 157px; }
.erroDateInpute{font-size: 20px;color:red;}.email-form{margin-top:10px;margin-
bottom: 30px;width:100%;padding:4px;}.btn-width{width: 90px;}.checkbox-merg{margin-top: 31px;}
```

Функції створені в базі даних

```
CREATE or REPLACE FUNCTION insert_usb_event (_userpc varchar, _dmac varchar, _mac v
archar, _ip varchar, _eventdate date,
_eventtime time, _VID varchar, _PID varchar, _SN varchar, _typeevent varchar)
returns int as
$$
declare
userid int;
begin
    IF( select count(*) from UserEvent where userpc = _userpc and dmac = _dmac) = 0
    THEN
       insert INTO UserEvent(userpc, dmac)
            values(_userpc, _dmac);
       select id into userid from userevent u
            where u.userpc = _userpc and u.dmac = _dmac;
       insert INTO USBEvent(id, userpc, eventdate , eventtime, VID, PID, SN, typeev
ent)
            values(userid, _userpc, _eventdate  , _eventtime, _VID, _PID, _SN, _typ
eevent);
       insert INTO net_info(id, userpc,ip, mac)
            values(userid, _userpc, _ip, _mac);
       if found THEN
            return 1;
       else return 0;
```

```
       end if;
     ELSE
        select id into userid from userevent u
            where u.userpc = _userpc and u.dmac = _dmac;
        insert INTO USBEvent(id, userpc, eventdate , eventtime, VID, PID, SN, typee
vent)
            values(userid, _userpc, _eventdate  , _eventtime, _VID, _PID, _SN, _typ
eevent);
        UPDATE net_info
        SET mac = _mac, ip = _ip
        WHERE id = userid;
        if found THEN
            return 1;
        else return 0;
        end if;
    END IF;
end
$$
language plpgsql;


--FUNCTIONS
CREATE or REPLACE FUNCTION update_net_info(_userpc varchar, _dmac varchar, _mac var
char, _ip varchar)
returns int as
$$
declare
userid int;
begin
    IF( select count(*) from UserEvent where userpc = _userpc and dmac = _dmac) = 0
    THEN
       insert INTO UserEvent(userpc, dmac)
            values(_userpc, _dmac);
       select id into userid from userevent u
            where u.userpc = _userpc and u.dmac = _dmac;
       insert INTO net_info(id, userpc,ip, mac)
            values(userid, _userpc, _ip, _mac);
       if found THEN
            return 1;
       else return 0;
       end if;
     ELSE
        select id into userid from userevent u
            where u.userpc = _userpc and u.dmac = _dmac;

        UPDATE net_info
        SET mac = _mac, ip = _ip
        WHERE id = userid;
        if found THEN
            return 1;
        else return 0;
        end if;
    END IF;
end
$$
language plpgsql;
```