НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

"___"_____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювальної мережі

Виконав студент IV курсу, групи ІП-81
(шифр групи)

Бершацька Олександра Олегівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. вик. Олійник Ю.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент Доцент кафедри ІСТ, к.т.н., доцент Ткач М.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2022 Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютеризованих систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

"___"_____2022 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Бершацькій Олександрі Олегівні
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення моніторінгу та класифікації системних логів локальної обчислювалюної мережі

керівник проєкту Олійник Юрій Олександрович ст. вик.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 10 »квітня 2022 р. №1033-с

2. Термін подання студентом проєкту « 19 » червня 2022 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури сховища даних
3) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
4) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використання _____

2) Схема структурна бізнес-процесів _____

3) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2022 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2022	
2	Аналіз існуючих методів розв’язання задачі	16.03.2022	
3	Постановка та формалізація задачі	20.03.2022	
4	Розробка інформаційного забезпечення	22.03.2022	
5	Алгоритмізація задачі	24.03.2022	
6	Обґрунтування вибору використаних технічних засобів	30.03.2022	
7	Розробка програмного забезпечення	03.04.2022	
8	Налагодження програми	30.04.2022	
9	Виконання графічних документів	25.05.2022	
10	Оформлення пояснювальної записки	28.05.2022	
11	Подання ДП на попередній захист	06.06.2022	
12	Подання ДП рецензенту	12.06.2022	
13	Подання ДП на основний захист	18.06.2022	

Студент _____ Олександра БЕРШАЦЬКА

(підпис) (ініціали, прізвище)

Керівник _____ Юрій ОЛІЙНИК

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Робота містить 44 рисунки і 22 таблиці.

Із розвитком комп’ютерних технологій ширшає різноманіття обчислювальних пристроїв, збільшується їх кількість у локальних мережах, необхідних для забезпечення безперебійної роботи підприємств та приватних установ.

Незважаючи на те, що потреба моніторити стан мережі не є новою, за рахунок росту складності мереж постає необхідність покращувати та удосконалювати вже існуючі рішення. Також має сенс роботи їх універсальними аби не було потреби змінювати засоби моніторингу з релізом кожного нового пристрою.

У міру того, як ширшає різноманіття мережі, додаються нові пристрої та вдосконалюються існуючі складнішає процес підтримки робочого стану, запобігання та вирішення проблем, які виникають. Оптимальним є саме передбачення та спроби запобігти критичним ситуація, тож вкрай важливо безперебійно слідкувати за станом системи та кожного пристрою зокрема.

У першому розділі було розглянуто і проаналізовано предметну область, пристрої локальної обчислювальної мережі такі як концентратори, маршрутизатори, комутатори, роутери. Було викладено особливості Syslog, UDP и TCP протоколів, основні відмінності протоколів транспортного рівня та особливості використання. Також було проведено аналіз відомих технічних рішень та програмних продуктів, наявних на поточний час, їхніх особливостей та наявних функцій. Крім того бути висунуті функціональні та нефункціональні вимоги і докладно описані задачі розроблюваного програмного забезпечення.

У другому розділі було описано процеси, які відбуватимуться в ході роботи програми такі як, перехоплення повідомлення, парсинг, перевірка на відповідність правилам, виконання дій. Проведено моделювання та аналіз

програмного забезпечення та представлено у форматі діаграми. Також було описано архітектуру застосунку та обґрунтовано доцільність обраних рішень для реалізації API та сховища даних. Представлено алгоритми кодування та шифрування, використані для забезпечення безпеки даних. Крім того було докладно описано основні методі та функції, які забезпечують виконання задачі програмного забезпечення.

КЛЮЧОВІ СЛОВА: LAN, SYSLOG, UDP, TCP, МЕРЕЖЕВА МОДЕЛЬ OSI, ВЕБЗАСТОСУНОК, WINDOWS SERVICE, .NET 6.0, GRAPHQL, ANGULAR

ABSTRACT

The variety of computing devices is expanding with the development of computer technology, their number in local networks is increasing, which is necessary to ensure the smooth operation of enterprises and private institutions.

Although the need to monitor the state of the network is not new, due to the growing complexity of networks there is a need to improve and refine solutions. It also makes sense to make them universal so that there is no need to change the monitoring tools with the release of each new device.

As the diversity of the network expands, new devices are added and existing ones are improved, making the process of maintaining, preventing, and resolving problems more complex. It is optimal to anticipate and try to prevent critical situations, so it is extremely important to continuously monitor the status of the system and each device.

In the first section the subject area, devices of a local area network such as hubs, routers, switches, routers were considered and analyzed. Features of Syslog, UDP and TCP protocols, the main differences of transport layer protocols and features of use were stated. An analysis of known technical solutions and software products currently available, their features and available functions was also performed. In addition, functional and non-functional requirements should be set and the tasks of the developed software should be described in detail.

The second section described the processes that will take place during the operation of the program such as, interception of messages, parsing, checking compliance with the rules, actions. Software modeling and analysis were performed and presented in diagram format. The architecture of the application was also described and the expediency of the selected solutions for the implementation of the API and data warehouse was substantiated. The encryption and encryption algorithms used to ensure data security are presented. In addition, the main methods and functions that ensure the performance of the software task were described in detail.

KEYWORDS: LAN, LAN, SYSLOG, UDP, TCP, OSI NETWORK MODEL, WEB-APPLICATION, WINDOWS SERVICE, .NET 6.0, GRAPHQL, ANGULAR

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-8101.045490.01.91	Технічне завдання	12	
3	А4	КПІ.ІП-8101.045490.02.81	Пояснювальна записка	65	
4	А4	КПІ.ІП-8101.045490.03.13	Опис програми	26	
5	А4	КПІ.ІП-8101.045490.04.51	Програма та методика тестування	8	
6	А4	КПІ.ІП-8101.045490.05.34	Керівництво користувача	19	
7	А3	КПІ.ІП-8101.045490.06.99	Схема структурна варіантів використань	1	
8	А3	КПІ.ІП-8101.045490.07.99	Схема структурна станів системи	1	
9	А3	КПІ.ІП-8101.045490.08.99	Креслення вигляду екранних форм	1	
10	А4		Звіт Unicheck	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-8101.045490.00.90		
Розроб.		Бершацька О.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Олійник Ю.О.					
Н.контр.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-81	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КЛАСИФІКАЦІЇ ЛОГІВ ЛОКАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ МЕРЕЖІ

Технічне завдання

КПІ.ІП-8104.045490.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юрій ОЛІЙНИК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Олександра БЕРШАЦЬКА

Київ – 2022

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювальної мережі.

Галузь застосування: системне адміністрування.

Наведене технічне завдання поширюється на розробку програмного забезпечення Syslog Server 045490, котра використовується для моніторингу, аналізу та класифікації системних повідомлень у локальній обчислювальній мережі та призначена для використання системними адміністраторами для підтримки мережі та швидкого реагування на проблеми, які можуть виникнути у роботи пристроїв та мережі цілком .

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Syslog Server є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для моніторингу та класифікації системних логів локальної обчислювальної мережі.

Метою розробки є полегшення задачі підтримки обчислювальної мережі у робочому стані, моніторингу пристроїв.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- вхід у систему з використанням власних унікальних облікових даних;
- зміна налаштувань панелі приладів;
- моніторинг результатів роботи програмного забезпечення.

4.1.1.2 Для адміністратора системи:

- вхід у систему з використанням власних облікових даних;
- створення користувачів;
- зміна налаштувань панелі приладів;
- створення правил обробки перехоплених повідомлень;
- створення фільтрів для повідомлень, що надходять;
- створення дій, виконуваних над повідомленнями, які задовольнятимуть фільтрам у межах одного правила;
- налаштування прослуховування повідомлень певних протоколів за певними портами;
- завантаження та менеджмент PowerShell скриптів;
- моніторинг результатів роботи програмного забезпечення.

4.1.2 Розробку серверної частини виконати на платформі .Net6.0. Розробку клієнтської частини виконати на платформі Angular.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення та зберігання чутливої інформації такої, як облікові дані користувачів та

адміністраторів, дані для авторизації SNMP trap’ів 3ї версії, дані про скрипти збережені в системі.

4.2.2 Передбачити захист від некоректних дій користувача шляхом обмеження доступності значного переліку функцій для звичайного користувача.

4.2.3 Забезпечити цілісність інформації в файлів зберігання даних та інформації про налаштування системи

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Вимоги до обслуговування не виставляються

4.3.3 Вимоги до обслуговуючого персоналу не виставляються

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати комп‘ютерах з операційною системою Windows.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору ..... Intel Core i3.

4.4.2.2 Об‘єм ОЗП..... 2048 Мб.

4.4.2.3 Частота центрального процесора …… 2.4 GHz.

4.4.2.4 Версія .NET …… .NET 6.0

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows 8, Windows 8.1 і т.д.)

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: JSON файл, PowerShell файл, graphQL mutation запит з тілом типу C#.

4.5.3 Результати повинні бути представлені в наступному форматі: JSON файл, результіти graphQL мутацій відповідно до переданих параметрів.

4.5.4 Програмне забезпечення повинно бути створено на платформі .NET 6.0 мовою C#, з використанням фреймворку GraphQL для рівню API та фреймворку Angular для написання клієнтської частини.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 66 аркушах формату А4 (без додатків 5.3.2 - 5.3.4).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на 2 аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна варіантів використання.

5.4.2 Схема структурна бізнес-процесів.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	10.03	
2.	Розробка технічного завдання	20.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	24.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03	Схема структурна варіантів використання, схема структурна бізнес-процесів
5.	Програмна реалізація програмного забезпечення	03.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	30.04	Тести, результати тестування
7.	Розробка матеріалів графічної частини проекту	25.05	Графічний матеріал проекту
8.	Розробка матеріалів текстової частини проекту	29.05	Пояснювальна записка
9.	Оформлення технічної документації проекту	04.06	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювальної мережі

КПІ.ІП-8104.045490.02.81

Київ – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LAN	–	Local Area Network – комп'ютерна мережа для обмеженого кола користувачів, що об'єднує комп'ютери в одному приміщенні або в рамках одного підприємства.
Syslog	–	system log – протокол, стандарт надсилання та реєстрації повідомлень про події, які відбуваються у системі; програмне забезпечення для моніторингу подій у мережі
UDP	–	user datagram protocol – один із протоколів в стеку TCP/IP.
TCP	–	transmission control protocol – протокол призначений для керування передаванням даних у комп'ютерних мережах, працює на транспортному рівні моделі OSI
Мережева модель OSI	–	абстрактна мережева модель для комунікацій і розробки мережевих протоколів, представляє рівневий підхід про уявлення організації мережі
Мережа	–	система зв'язку між двома чи більше комп'ютерами.

ВСТУП

Із плином технологічного прогресу різноманіття обчислювальних пристроїв значно збільшується, тож локальні мережі багатьох підприємств стають комплексніше. У міру того, як ширшає різноманіття мережі, додаються нові пристрої та вдосконалюються існуючі складнішає процес підтримки робочого стану, запобігання та вирішення проблем, які виникають. Оптимальним є саме передбачення та спроби запобігти критичним ситуація, тож вкрай важливо безперебійно слідкувати за станом системи та кожного пристрою зокрема.

Для того, щоб якомога швидше локалізувати проблему та віднайти рішення необхідно стабільно моніторити стан мережі, мати уявлення про поточний та минулий стани, а також мати можливість з урахуванням наявних даних передбачити майбутні. Тож рішення для моніторингу стану пристроїв вже давно є нагальною потребою.

Актуальність теми – обчислювальні пристрої використовуються підприємствами, які виконують задачі у різноманітних сферах. Також варіюється складність мережі та її внутрішнє різноманіття. У той же час універсальними є протоколи, з допомогою яких пристрою комунікують між собою. Через уніфікованість системи комунікації доцільним є створення агенту, який матиме можливість моніторити стан мережі та нотифікуватиме користувача – частіше за все системного адміністратора – про підозрілу активність або таку, що вимагає уваги людини.

Мета – збільшити можливості моніторингу системних логів локальної обчислювальної мережі за рахунок покращення інформативності з застосуванням правил та користувацьких фільтрів.

Призначення – моніторинг стану мережі, класифікація та обробка повідомлень пристроїв у системі.

1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Локальна комп'ютерна мережа є об'єднанням певного числа комп'ютерів на відносно невеликій території. В порівнянні з глобальною мережею, локальна мережа зазвичай має більшу швидкість обміну даними, менше географічне покриття та відсутність потреби використовувати запозичену телекомунікаційну лінію зв'язку.

Із розвитком нових досягнень в технологіях мереж на початку 80-х років мережі почали поширюватись. Кожна компанія створювала власні стандарти та обладнання, технічні засоби мереж та програмне забезпечення. Ця конкуренція привела до несумісності створених технологій, та виникненню проблем узгодження обладнання. Необхідно було робити повне оновлення обладнання.

Спочатку були створені стандарти LAN. Ці стандарти були відкритим набором директив для створення мережевих технічних засобів та програмного забезпечення. Апаратне забезпечення різних компаній було узгоджено, що давало можливість будувати надійні і стабільні LAN.

Системний журнал – стандарт надсилання та реєстрації повідомлень про події, що відбуваються в системі, тобто створення журналів подій, що використовується в комп'ютерних мережах, що працюють за протоколом IP. Терміном «syslog» називають як стандартизований мережевий протокол syslog, так і програмне забезпечення (додаток, бібліотеку), яке займається відправкою та отриманням системних повідомлень.

1.2 Змістовний опис і аналіз предметної області

1.2.1 Пристрої локальної мережі

Прості локальні мережі зазвичай складаються з кабелів та одного або кількох комутаторів[1] або повторювачів. Складнощі можуть включати концентратори, мости або маршрутизатори. Концентратор – пристрій фізичного рівня, до якого підключають комп'ютери в мережі за топологією «зірка». Пристрій транслює у всі приєднані сегменти мережі пакети даних які він отримує не знаючи де знаходиться їх кінцевий отримувач. Мост дозволяє станціям будь-якої з мереж звертатись до ресурсів іншої мережі, використовується для збільшення довжини ліній зв'язку або кількості вузлів мережі. Маршрутизатор (або роутер) використовується для поєднання двох чи більше мереж і на підставі інформації про топологію мережі та певних правил приймає рішення про пересилання пакетів мережевого рівня між різними сегментами мережі. Комутатор призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного сегмента. На відміну від концентратора, що поширює трафік від одного приєднаного пристрою до всіх інших, комутатор передає дані лише безпосередньо отримувачу. Це підвищує продуктивність і безпеку мережі, рятуючи інші сегменти мережі від необхідності обробляти дані, які їм не призначалися. Серед комутаторів вирізняють також повторювачі (чи ретранслятори) – мережеве обладнання для підсилення сигналу. Вони призначений для збільшення відстані мережевого з'єднання шляхом повторення електричного сигналу «один на один».

1.2.2 Syslog протокол

У обчислювальній техніці syslog[2] є стандартом реєстрації повідомлень у мережі. Стандартом передбачається, що джерела формують текстові повідомлення про події, що відбуваються в них, і передають їх на обробку серверу Syslog (“syslog daemon” чи “syslog server”), використовуючи один з

мережевих протоколів сімейства IP – UDP або TCP. Формування повідомлень про події та їх передача відбувається за певними правилами, які називають протоколом Syslog. Як правило, повідомлення має розмір до 1024 байт і надсилається у відкритому вигляді. Протокол Syslog має дві специфікації. RFC 3164[3] – простіший варіант протоколу, яких є застарілим проте, проте досі використовується деякими пристроями, тож підтримується окремими мережами. RFC 5424[4] – актуальний на поточний момент часу стандарт протоколу, який було опубліковано у 2009 році. Згідно цього протоколу повідомлення складається з заголовку, структурованих даних та безпосередньо повідомлення. Заголовок складається з інформації, яка описує властивості повідомлення: пріоритету, версії протоколу, мітки часу, назви хоста, назви додатку, які повідомлення надіслав, ідентифікаційного номеру процесу та ідентифікатора повідомлення.

1.2.3 UDP та TCP протоколи

У комп'ютерних мережах протокол UDP[5] – це стандарт зв'язку, який використовується переважно для встановлення стійких до втрат з'єднань з малою затримкою між програмами в Інтернеті. UDP прискорює передачу, дозволяючи передавати дані до того, як сторона, що приймає, надасть згоду. В результаті UDP вигідний для термінових комунікацій, включаючи передачу голосу, пошук в системі доменних імен і відтворення відео або аудіо.

UDP є альтернативою протоколу управління передачею TCP[6]. Обидва протоколи працюють поверх IP і іноді називаються UDP/IP або TCP/IP[7]. Проте між ними існують важливі відмінності. UDP забезпечує зв'язок між процесами, а TCP підтримує зв'язок між хостами. RFC 768 [8] вичерпно описує стандарт протоколу UDP.

TCP відправляє окремі пакети та вважається надійним транспортним середовищем. UDP відправляє повідомлення, які називають дейтаграмами. UDP не дає жодних гарантій, що дані будуть доставлені, і не пропонує

спеціальних функцій для повторної передачі втрачених або пошкоджених повідомлень, що в свою чергу забезпечує TCP. Протокол TCP докладно описаний у стандарті RFC 768.

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Після аналізу існуючих технічних рішень можна помітити, що більшість відомих проектів є комерційними. Вони пропонують комплексні продукти, які можуть бути зручно налаштовані під конкретні потреби локальної мережі та часто не вимагають на це багато часу, бо є достатньо дружніми до користувача. Проте слід відзначити, що напрацювання з базовим функціоналом можна знайти і з відкритим кодом, але в такому разі розраховувати на підтримку від розробників не доводиться. Такий варіант може задовольнити базові потреби маленької компанії або окремого розробника. Поширена реалізація Syslog серверу – десктопний додаток для налаштування процесів моніторингу та обробки та веб-частиною для перегляду результатів роботи додатку у режимі read-only. Також часто можна побачити лише додаток. Він включає в себе можливість як налаштовувати систему, так і наглядно бачити результати роботи, а також систематизувати уявлення про події у мережі за допомогою різноманітних графіків.

1.3.2 Аналіз відомих програмних продуктів

Існує кілька типів інструментів, які можна використовувати для моніторингу та централізації системних журналів, але деякі з них більше підходять для налаштувань підприємства або великого бізнесу. Можете переглядати свої події за допомогою засобу перегляду подій у Windows, але якщо використовується інша операційна система або у мережі більше одного пристрою, необхідно використовувати сервер системного журналу, щоб централізувати процес моніторингу.

Провівши аналіз наявних програмних рішень, можна виділити деякі з них, які більш популярні серед користувачів через доступний функціонал та зручність.

Kiwi Syslog Server[9] – простий у конфігурації інструмент керування системним журналом. Він прослуховує логи та повідомлення Syslog від мережевих пристроїв, таких як маршрутизатори, хости, комутатори та інші пристрої з підтримкою Syslog.

Додаток можна запустити у режимі Windows Service та отримувати повідомлення, що надходять з систем Linux, UNIX та Windows. Також можна використовувати механізм автоматичної відповіді, наприклад, сповіщення електронною поштою, запуск виконання скриптів або програм, логування у файлі або базу даних ODBC чи переслати повідомлення людині, яка відповідає за вирішення конкретної проблеми. Крім цього користувачу доступна можливість конфігурувати фільтри, аби виділити лише потрібні повідомлення, наприклад, за IP адресою чи пріоритетом, та дії, які можуть бути виконані над повідомленням.

У Kiwi Syslog наявна веб-версія для перегляду повідомлень, які надходять, у режимі read-only. Це дозволяє користувачам крім безпосередньо адміністратора слідкувати за станом локальної мережі.

EventLog Analyzer[10] – надає інструменти для кореляції журналів подій у реальному часі, щоб мати можливість швидко визначити, коли виникла проблема, чи виглядають логи підозрілими (або схожими на загрозу безпеці). Також можете створити правила для сповіщень, аби отримувати сповіщення про те, що є найважливішим для організації з точки зору часу роботи служби або проблем із безпекою. Додаток також включає попередньо визначені правила для журналів подій безпеки.

Для цілей відповідності EventLog Analyzer також містить шаблони звітів для багатьох різних нормативних актів, тому, якщо вам потрібно створити звіт про події, які відбулися, ви можете зробити це легко. Архівування даних

журналу також допомагає вам виконувати вимоги щодо збереження історичних даних. У поєднанні з можливостями аудиту ви можете перевіряти дані журналу з ваших периметральних пристроїв і стежити за маршрутизаторами, комутаторами, входами та виходами користувачів, а також визначати, чи проходить через вашу мережу шкідливий або шкідливий трафік.

EventLog Analyzer орієнтований на безпеку, а не лише на моніторинг та керування з метою усунення несправностей. Він включає інструменти пошуку та інструменти аналізу, але це краще підходить для ситуацій, коли необхідно відстежувати логи в першу чергу з метою безпеки або потенційних порушень.

Nagios Log Server[11] – корелює події журналу на серверах в режимі реального часу, що дозволяє швидко вирішувати проблеми, перш ніж вони вплинуть на кінцевих користувачів. Завдяки доступному та простому у використанні API можна також можете інтегрувати Nagios Log Server із сторонніми рішеннями та зовнішніми додатками, якщо це необхідно.

Як і EventLog Analyzer, Nagios Log Server розроблено спеціально для аудиту безпеки та мережі, тому можна швидко отримувати сповіщення про шкідливі або підозрілі події. Він також включає можливість виконувати сценарії або надсилати електронні листи у відповідь на проблему. Додаток також має багатокористувацькі можливості, тож вся команда може мати доступ, а також будь-який інший персонал, якому потрібна інформація про моніторинг.

Графічний інтерфейс додатку може бути змінений, тож можна конфігурувати панель інструментів з точки зору макета та дизайну, щоб відповідати кожному користувачеві та його перевагам.

Paessler PRTG Network Monitor[12] – інструмент, який можна використовувати як сервер системного журналу. PRTG працює в основному з датчиками, які встановлюються у мережі, а потім використовуються як пристрої моніторингу в певних точках.

Одним із способів роботи PRTG як сервера системного журналу є використання датчика приймача. Цей датчик отримує та аналізує повідомлення системного журналу, включаючи те, скільки їх було отримано кожну секунду, скільки описано як тип повідомлення «попередження» або «помилка», а також чи перекидаються пакети на порт системного журналу. Однак не можна використовувати велику кількість цих типів датчиків, інакше вони будуть мати великий вплив на продуктивність системи. Якщо мережа дуже велика та включає багато пристроїв, використання такої кількості датчиків може призвести до уповільнення роботи.

Можна налаштувати датчик Syslog Receiver як централізований датчик для моніторингу всіх повідомлень, що надходять через мережу, або налаштувати кожного відправника як пристрій, а потім застосувати датчик приймача Syslog до кожного з них. Централізований датчик зменшує кількість потрібних датчиків, але створює більше навантаження на процесор. Підхід «сенсор на пристрій» зменшує навантаження на центральний процесор, але може значно збільшити кількість потрібних датчиків, залежно від розміру мережі.

PRTG також діє як загальний мережевий монітор для мережі з моніторингом стану пристрою, керуванням продуктивністю мережі та іншими корисними інструментами.

Таблиця 1.1 – Порівняльна характеристика відомих програмних рішень

	Kiwi Syslog Server	**EventLog Analyzer**	**Nagios Log Server**	**Paessler Network Monitor**
UDP	+	+	+	+
TCP	+	+	+	+

Продовження таблиці 1.1

The highest TLS version	1.3	1.2	1.2	1.3
SNMP traps support	+	+	-	-
Supported Database Outputs	MySQL Oracle Microsoft SQL Microsoft Access	MySQL PostgreSQL Oracle	Sybase Firebird Ingres Oracle	MySQL PostgreSQL SQLite Microsoft SQL
Message filtering	+	+	+	+
RFC standard supported	3164, 5424	3164	3164	3164
Web-застосунок	+	+	-	+

Як видно з таблиці, усі перелічені рішення підтримують базові протоколи передачі повідомлень TCP та UDP, проте різниться версія підтримуваного TLS протокола для повідомлень SecureTCP[8]. На поточний момент часу остання версія TLS – 1.3. Також не вся рішення підтримують перехват SNMP даних та актуальну версію RFC стандарту для Syslog протоколу.

1.4 Аналіз вимог до програмного забезпечення

Програмне рішення для моніторингу мережі повинно виконувати наступні функції:

- прослуховування визначених мережевих портів;
- обробка повідомлень, надісланих протоколами UDP та TCP;
- парсинг повідомлення відповідно до стандарту RFC5424;
- створення правил, які складаються з набору фільтрів та дій;
- створення користувацьких фільтрів;
- фільтрування надісланих повідомлень відповідно існуючим фільтрам;
- створення та конфігурація дій;
- класифікація повідомлень;
- виконання дій у разі відповідності існуючим фільтрам у межах одного правила.

1.4.1 Розроблення функціональних вимог

Функціональні вимоги до програмного забезпечення для моніторингу, обробки та класифікації повідомлень у локальній обчислювальній мережі наведені у таблиці 1.2. Діаграму варіантів використання програмного забезпечення наведено у графічному матеріалі до дипломного проєкту.

Таблиця 1.2 – Функціональні вимоги

Актор	Варіант використання	Функціональна вимога	Пріоритет
Користувач	Створення правила	Можливість створити правило, яке складатиметься з набору фільтрів та дії	Високий
	Створення фільтрів у правилі	У вікні створення правила обрати додавання фільтру та заповнити поля відповідно до типу обраного фільтру. Доступні фільтри за IP-адресою або її частиною, за протоколом, за значеннями пріоритету, походження та рівню.	Високий
	Фільтр «IP-address»	Фільтрація за IP-адресою або її частиною	Середній
	Фільтр за протоколом	Фільтрація за протоколом, яким було надіслано повідомлення	Середній

Продовження таблиці 1.2

	Фільтр за значенням пріоритету	Фільтрація за значенням пріоритетності	Середній
	Фільтр за походженням повідомлення		Середній
	Фільтр за рівнем терміновості		Середній
	Створення дій у правилі	У вікні створення правила обрати додавання дії та заповнити поля відповідно до типу обраної дії.	Високий
	Вивід повідомлення на дисплей		Середній
	Дія логування повідомлення у базу даних		Середній
	Дія надсилання email-повідомлення		Середній

Продовження таблиці 1.2

	Аналіз результатів фільтрації та виконання дій	Після надходження повідомлення користувач може впевнитися в коректності роботи програми за допомогою засобів візуалізації	Високий

1.4.2 Розроблення нефункціональних вимог

Вимоги до апаратного та програмного забезпечення середи, де буде встановлено застосунок, наведені у таблиці 1.3.

Таблиця 1.3 – Вимоги до системи

Операційна система	• Windows Server 2019 • Windows Server 2016 • Windows Server 2012 and 2012 R2 • Windows 10 • Windows 8.1 • Windows 8
ЦП	Не менш ніж 2.4 GHz
Пам’ять	2 GB
Версія .NET	.NET 6.0

Для запису повідомлень до бази даних необхідно мати встановлену одну з перелічених баз даних на вибір користувача.

Таблиця 1.4 – Бази даних

SQL Server	• Microsoft SQL Server 2019 • Microsoft SQL Server 2017 • Microsoft SQL Server 2016 • Microsoft SQL Server 2014 • Microsoft SQL Server 2012
Інші бази даних	• Microsoft Access • MySQL • Oracle

1.5 Постановка задачі

Робота має на меті створення продукту для полегшення задачі підтримки локальних мереж різних масштабів за допомогою програмного забезпечення моніторингу, обробки, класифікації повідомлень та виконання дій над отриманими повідомленнями, які були заздалегідь налаштовані користувачем. Тобто можна виділити наступні задачі:

- прослуховування мережевих портів;
- парсинг отриманих повідомлень відповідно до стандарту протоколу;
- створення правил обробки повідомлень;
- класифікація повідомлень на основі створених правил;
- конфігурація фільтрів для повідомлень, які надходять;
- конфігурація дій, які будуть виконані у разі проходження повідомленням фільтрів у межах правила;
- можливість наглядно спостерігати за результатами роботи застосунку за допомогою дашборду.

Детальний опис кожної із задач наведено нижче.

До задачі прослуховування мережевих портів входить перехоплення повідомлення надісланого певним протоколом на певний порт. Наприклад,

портом за замовчуванням для протоколу UDP є 514, а для TCP - 1468. Тобто, якщо користувач за власним бажанням не змінить властивості мережі за замовчуванням, то кожне повідомлення надіслане у мережі за допомогою протоколу UDP на 514-й порт повинно бути перехоплене додатком. Теж саме для протоколу TCP – кожне повідомлення надіслане у мережі протоколом TCP на 1468-й порт повинно бути перехоплене додатком. У разі зміни портів у конфігурації мережі чи пристроїв, які надсилають повідомлення необхідно змінити налаштування додатку аби прослуховувати належні порти.

Задача парсингу повідомлень має на меті перетворення отриманого масиву байтів у дані повідомлення. Окрім самого тексту повідомлення згідно останнього стандарту Syslog-повідомлення RFC 5424 необхідно отримати пріоритет, версію протоколу, мітку часу, ім’я хоста, який надіслав повідомлення, назву додатку, який створив повідомлення, ідентифікатор процесу та повідомлення та безпосередньо текст. Ці властивості, їх місце і призначення фіксовані та описані у стандарті, тож, якщо отримане повідомлення не відповідає цим нормам, масив байтів сприймається конвертується у строку та сприймається як текст повідомлення. У разі коли наявні лише деякі з необхідних властивостей, вони ідентифікуються коректно, а решта залишаються пустими або заповнюються значеннями за замовчування.

Аби не потонути у потоці мережевих повідомлень, їх потрібно фільтрувати за певними ознаками. Тож правило – це набір фільтрів та дій. Фільтрація може відбуватися за певною межею пріоритету чи IP-адресом або проміжком IP-адресів. Тож якщо перехоплене повідомлення надійшло від хосту з вказаною у фільтрі певного правила IP-адресою, то будуть виконані дії, створені у межах цього правила. Правил може бути декілька. Фільтрів у межах одного правила може більше одного або не бути зовсім. Тоді дії цього правила виконуватимуться над кожним перехопленим повідомленням. Створювати правило без дії не має сенсу, бо в такому разі система жодним чином не сповістить користувача про перехоплене повідомлення. Серед варіантів дій

показати повідомлення на дашборді, записати повідомлення у файл або базу даних.

Висновки до розділу

У першому розділі було описано і проаналізовано предметну область, а саме пристрої локальної обчислювальної мережі такі як концентратори, маршрутизатори, комутатори. Було викладено особливості Syslog, UDP и TCP протоколів, основні відмінності протоколів транспортного рівня та особливості використання. Також було проведено аналіз відомих технічних рішень та програмних продуктів, наявних на поточний час, їхніх особливостей та наявних функцій.

Крім того бути описані функціональні та нефункціональні вимоги і докладно описані задачі розроблюваного програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Перш ніж розпочинати роботу над реалізацією застосунку необхідно визначити архітектуру програмного забезпечення і змоделювати внутрішні процеси.

Спочатку користувачеві необхідно налаштувати Syslog сервер – зазначити, які протоколи транспортного рівня прослуховувати та за якими портами. Тож першим кроком у процесах роботи додатку буде налаштування параметрів роботи серверу. Наступним кроком є створення правил обробки повідомлень, які складаються з фільтрів та дій. Далі можна перевірити роботу серверу та вірність конфігурування, надіславши тестове повідомлення за допомогою Syslog генератору.

Діаграму бізнес-процесів у розробленому застосунку наведено у графічному матеріалі до дипломного проєкту.

2.2 Архітектура програмного забезпечення

Реалізація back-end частини додатку було виконано на платформі .Net 6.0, яка дозволяє створювати програмне забезпечення під різні операційні системи, мовою C#. Обрано n-layer архітектуру[13], як більш оптимальну для виконуваних задач.

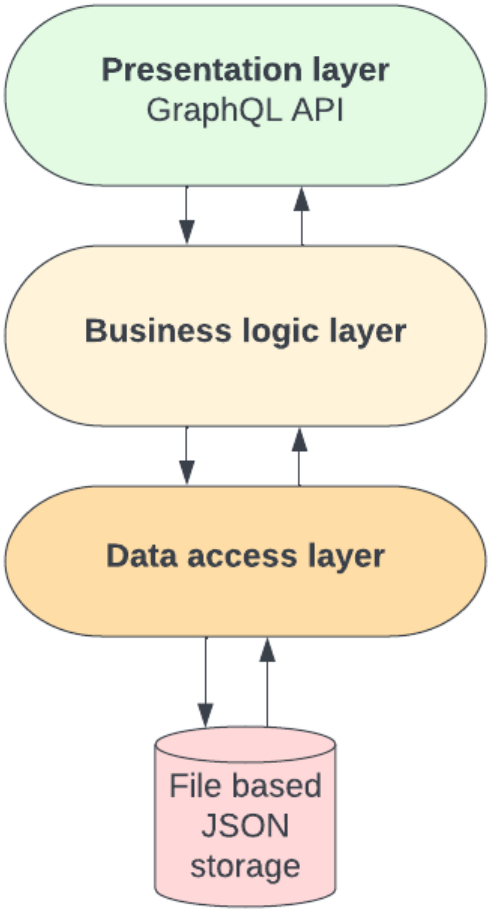


Рисунок 2.2 – Схема архітектури

Для реалізації **рівня презентації** було обрано graphQL[14], мову запитів і керування даними з відкритим вихідним кодом для API, а також середовище виконання для виконання запитів із наявними даними. Перевага graphQL API перед REST насамперед у тому, що graphQL надає можливість вказати, які саме дані необхідно отримати після виконання конкретного запиту, замість того, аби після вже виконання результат фільтрувати. Таке рішення робить оптимальнішою, систему, якій необхідна гнучкість API запитів, та скорочує час їх виконання.

На **рівні бізнес логіки** реалізовано безпосередньо функціонал програмного забезпечення, наприклад функції парсингу та прослуховування, а також обробки повідомлень, перевірка, чи відповідає повідомлення наявним фільтрам та виконання заданих дій. Крім того сам процес створення правил,

фільтрів, дій та процес конфігурування додатку реалізовано саме на рівні бізнес логіки.

Рівень доступу до даних відповідає за організацію доступу до даних додатку, що, очевидно, з назви. Основна інформація, які зберігатимуться на машині користувача – налаштування продукту. Тож було прийняте рішення зберігати ці дані у файлах формату JSON аби спростити сам процес зберігання та реалізації процесу збереження даних. Крім того цей підхід дозволяє зменшити об'єм простору, необхідного для даних, дозволяє не встановлювати сервер бази даних та спрощує процес маніпуляцій.

На рівні бізнес логіки реалізовано процедуру прослуховування мережевих портів. Тож коли повідомлення було перехоплено, воно подається у систему в форматі даних ISyslogPacket. Деталі реалізації типу наведені у таблиці 2.1.

Таблиця 2.1 – Опис сутності ISyslogPacket

Назва поля	**Опис**	**Тип**
RawPacket	Набір байтів повідомлення	byte []
Message	Текст повідомлення	string
SyslogProtocol	Протокол	SyslogProtocol
Severity	Значення терміновості	int
Facility	Класифікація	int

Тип даних, отриманий з байтів перехопленого повідомлення подається на вхід парсеру. Масив байтів перетворюється на строку та за допомогою регулярного виразу отримується вся інформація, наявна у повідомленні. В результаті парсингу утворюється екземпляр класу ILogEntryOrchestratorData. Специфікацію типу даних ILogEntryOrchestratorData наведено у таблиці 2.2.

Таблиця 2.2 – Опис сутності ILogEntryOrchestratorData

Назва поля	**Опис**	**Тип**
IpAddress	Адрес джерела повідомлення	string

Продовження таблиці 2.2

RawMessage	Байти перехопленого повідомлення	byte[]
RawMessageText	Повідомлення цілком у форматі строки	string
InputSourceType	Тип протоколу, яким було надіслано повідомлення	InputSourceType
LocalAddress	Адреса серверу	string
LocalPort	Порт, на який прийшло повідомлення	int
RFCHeader	Заловок повідомлення відповідно до RFC 5424	string
RFCHostname	Назва хоста, який надіслав повідомлення	string
RFCDate	Дата, коли повідомлення було надіслано	string
RFCPIDTag	Идентифікатор процесу	string

Після вдалого виконання парсингу повідомлення подається у менеджер правил, де одне за одним перебираються кожне з існуючих у системі правил. Спершу необхідно перевірити, чи відповідає повідомлення усім фільтрам, наявним у правилі. Якщо наявна невідповідність хоча б одному фільтру, виконання поточного правила припиняється та опрацювання наступного. Коли повідомлення відповідає усім фільтрам, розпочинається виконання дій, створених у межах того ж правила. Вихідні дані роботи залежать від типів виконуваних дій. Наразі система надає можливості відобразити повідомлення на панелі статистики, виконати логування у файл або до журналу подій Windows, надіслати email повідомлення, переслати перехоплене повідомлення на інший хоста або надіслати нове syslog повідомлення, виконати PowerShell скрипт.

Під час виконання дій до процедури PerformAction надходять дані типу ILogEntryOrchestratorData з інформацією про повідомлення та IActionInfo. Специфікація типу IActionInfo наведено у таблиці 2.3.

Таблиця 2.3 – Опис сутності IActionInfo

Назва поля	**Опис**	**Тип**
Name	Назва дії	string
ActionSystemType	Системний тип дії	Type
ActionIdentifier	Ідентифікатор дії	ActionIdentifier

Інтерфейс, наведений у таблиці 2.3, імплементується під час реалізації кожною з дії в залежностей від особливостей дії та інформації, потрібної під час її виконання. Прикладом реалізації є тип RunScriptActionInfo. Опис сутності RunScriptActionInfo наведено у таблиці 2.4.

Таблиця 2.4 – Опис сутності RunScriptActionInfo

Назва поля	**Опис**	**Тип**
FileName	Назва файлу, в якому міститься скрипт	string
CommonVariablesAccess	Права доступу до загальних змінних	VariablesAccessEnum
OtherVariablesAccess	Права доступу до інших змінних	VariablesAccessEnum
CustomVariablesAccess	Права доступу до користувацьких змінних	VariablesAccessEnum

2.3 Конструювання програмного забезпечення

Аби зрозуміти організацію рівню бізнес логіки, доречно розділити реалізацію задач програмного забезпечення на чотири складові: прослуховувач повідомлень, парсер, менеджер правил та менеджер дій.

Діаграму послідовностей обробки повідомлень наведено у графічному матеріалі до дипломного проєкту. У таблиці 2.5 наведені основні методи, які реалізують функціонал прослуховування повідомлень.

Таблиця 2.5 – Специфікації функцій пов’язаних з прослуховуванням повідомлень

Назва функції	**Вхідні параметри**	**Вихідні параметри**	**Призначення**
StartProtocolListeners	Налаштування протоколу, endpoints	–	Викликати запуск певного прослуховувача в залежності від протоколу
StartListeningFor SyslogPacket	Налаштування протоколу, endpoint	–	Запустити процес прослуховування певного протоколу
ListenForSyslog Packets	Налаштування протоколу, локальний endpoint, лістенер певного протоколу	–	Ініціалізувати перехоплений пакет
StopListeners	–	–	Викликати зупинку певного прослуховувача в залежності від протоколу

Продовження таблиці 2.5

StopTcpListeners	–	–	Перервати прослуховування TCP пакетів
StopUdpListeners	–	–	Перервати прослуховування UDP пакетів
IsPaused	–	Булеве значення	Визначити, чи роботу сервісу призупинено
IsShutdown	–	Булеве значення	Визначити, чи сервіс “вимкнуто”

У таблиці 2.6 наведені функції, які відповідають за парсинг повідомлень.

Таблиця 2.6 – Специфікації функцій пов’язаних з парсингом

Назва функції	**Вхідні параметри**	**Вихідні параметри**	**Призначення**
Parse	Пакет повідомлення	ILogEntryOrchestratorData – формат повідомлення після парсингу	Розібрати байти повідомлення на основні складові Syslog

Продовження таблиці 2.6

IsLegacyMessage	Повідомлення текстом	Булеве значення	Перевірити чи відповідає повідомлення останньому стандарту
CreateLogEntryFromMatch	Пакет повідомлення, маркер чи повідомлення старого стандарту, match – результат парсингу за допомогою регулярного виразу, rawText – байти повідомлення приведені до строки	SysLogOrchestratorData – реалізація інтерфейсу ILogEntryOrchestratorData безпосередньо для Syslog повідомлень	Створити об’єкт повідомлення після парсингу з результату роботи регулярного виразу
GetRawMessageFromPacket	Пакет повідомлення	Байти повідомлення приведені до строки	Перетворити байти повідомлення на строку

Продовження таблиці 2.6

ExtractMessageDateTime	Маркер чи повідомлення старого стандарту, match	Дата та час	Виділити дату та час з повідомлення
ParseTimeFromMessage	Маркер чи повідомлення старого стандарту, match	Зсув дати та часу	Виділити час з повідомлення
HandleUnparseableDateTime	Маркер чи повідомлення старого стандарту, match, зсув дати та часу	Строка	Обробити некоректно поданий час
GetFacilityAndSeverityFromPriority	Значення пріоритету повідомлення	Значення facility та severity	Вирахувати значення facility та severity зі значення пріоритету

Продовження таблиці 2.6

InferMessageYear	Поточний час, здвиг часу, час, який було розпаршено з повідомлення	Цілочисельне значення	Визначити різницю між часом, коли повідомлення було надіслано, та часом, коли програма повідомлення обробляє
LogRegexMatchFailure	Пакет повідомлення, сире повідомлення текстом	void	Логувати помилку чи некоректну поведінку під час парсингу через регулярний вираз
CalculatePriority	Значення facility та severity	Значення пріоритету	Вирахувати значення пріоритету зі значення facility та severity

Функцій, які реалізують роботу менеджеру правил, наведені у таблиці 2.7.

Таблиця 2.7 – Специфікації функцій пов’язаних з правилами

Назва функції	**Вхідні параметри**	**Вихідні параметри**	**Призначення**
GetAllRules	–	Лист об’єктів типу ILogRule	Отримати всі наявні правила
AddRule	Об’єкт типу ILogRule	Id нового правила у форматі guid	Додати нове правило до системи
AddChildRule	Об’єкт типу ILogRule, id батьківського правила	Id нового правила у форматі guid	Додати підправило до основного правила
UpdateRule	Об’єкт типу ILogRule	Id правила, яке було оновлене	Оновити існуюче правило
DuplicateRule	Id правила	Об’єкт типу IRuleManagerCommandInfo	Дублювати існуюче правило
DeleteRule	Id батьківського правила, маркер чи видаляти підправила	Об’єкт типу IRuleManagerCommandInfo	Видалити правило

Продовження таблиці 2.7

EnableDisableRules	Id правила, маркер enable чи disable	Об’єкт типу IRuleManagerCommandInfo	Вимкнути чи увімкнути правило
InitializeRules	–	void	Ініціалізувати вже існуючі правила під час старту роботи серверу

У таблиці 2.8 наведені функції, які реалізують операції над діями та їх виконання.

Таблиця 2.8 – Специфікації функцій пов’язаних з виконанням дій

Назва функції	**Вхідні параметри**	**Вихідні параметри**	**Призначення**
ExecuteActions	Відомості про правило	void	Виконати дію відповідно до реалізації конкретної дії
LogActionExecuted	IRuleTestResult result, IActionInfoBL actionInfo, ActionResult actionResult	void	

2.4 Аналіз безпеки даних

Створюючи сучасний додаток, неможливо обійти тему безпеки даних та захисту персональної інформації користувача. На кожному етапі роботи програми необхідно забезпечити якомога надійніший проте оптимальних для конкретного типу даних захист від стороннього втручання.

Оскільки типом сховища даних у додатку було обрано File Based JSON storage, оптимальним варіантом захисту даних буде їх кодування. Проте необхідно розібратись, які саме дані необхідно кодувати та яким саме чином. Також треба зазначити, що вся користувацька інформація зберігатиметься локально саме на машині користувача, де встановлений сервер Syslog.

Основний тип даних, які зберігатимуться – інформація о налаштуваннях серверу. Тобто таке, як чи потрібно перехоплювати повідомлення, надіслані протоколом UDP чи TCP, які саме порти прослуховувати, яку версію TLS використовувати. Також конфігурації правил, а саме набори опису фільтрів та дій, зберігатимуться у файлах. Така інформація не потребує кодування, бо не може скомпрометувати користувача у разі її викрадення. Тож її можна зберігати у форматі JSON plain-text.

Та окрім налаштувань серверу необхідно зберігати дані саме про користувачів, їх імена, пошти та паролі. Такі дані вимагають кодування та шифрування. Користувацькі пошти необхідно саме кодувати, а не шифрувати, бо необхідно мати змогу їх розкодувати. Тож використано алгоритм аутентифікованого кодування XChaCha20Poly1305, на вхід якому подається nonce – масив псевдовипадково згенерованих байтів – та безпосередньо строка з поштою користувача. Для зберігання паролів необхідно використовувати саме хешування, бо немає потреби розкодувати пароль. Тож для хешування паролю використовується функція деривації ключа Argon2 і попередньо отримане значення SHA паролю.

Інформацію про користувачів протоколу SNMP v3 достатньо використати Microsoft.AspNetCore.DataProtection бібліотеку. Вона кодує файл цілком, перетворюючи текст будь-якого формату на один рядок символів.

Висновки до розділу

У другому розділі було описано процеси, які відбуватимуться в ході роботи програми такі як, перехоплення повідомлення, парсинг, перевірка на відповідність правилам, виконання дій. Проведено моделювання та аналіз програмного забезпечення та представлено у форматі діаграм активності та послідовності. Також було описано архітектуру застосунку та обґрунтовано доцільність обраних рішень для реалізації API та сховища даних. Представлено алгоритми кодування та шифрування, використані для забезпечення безпеки даних. Крім того було докладно описано основні методі та функції, які забезпечують виконання задачі програмного забезпечення.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вступ

План тестування програмного забезпечення необхідний для того, аби чітко розуміти стадію готовності програмного забезпечення та мати можливість визначити. У розділі буде розглянуто план тестування, який базуватиметься на міжнародному стандарті IEEE 829-2008. Тестування відбуватиметься на рівні “системного тестування” програмного забезпечення. Таке тестування має на меті:

- розробити перелік інструментів, які будуть використані під час тестування;
- сформулювати методологію проведення тестування;
- перелічити функції, які підлягатимуть тестуванню;
- визначити вимоги до тестового середовища;
- забезпечити високий рівень безпеки та захисту даних користувача;
- перевірити відповідність дизайну вимогам, описаним у технічному завданні;
- протестувати вправність інтерфейсу користувача;
- перевірити коректність результатів, отриманих під час фільтрації повідомлень;
- перевірити коректність результатів, отриманих під час виконання дії, зазначених у кожному зі створених правил.

Будуть використані наступні методи тестування: функціональне, тестування продуктивності, користувацького інтерфейсу додатку та системи в цілому. Типи тестів, які будуть виконані: unit тести, реалізовані за допомогою фреймворку xUnit[16] та інтеграційні тести.

Вправність системи повністю буде перевірено через:

- ручне динамічне тестування – введення коректних та некоректних даних по черзі;
- тестування на відповідність функціональним вимогам до програмного забезпечення;
- статичне тестування коди програмного застосунку та серверу;
- тестування інтерфейсу користувача.

3.2 Об’єкти тестування

Програмне забезпечення додатку складається с клієнтської та серверної частин. Обидві повинні бути протестовані, кожна окремо та в інтеграції одна з одною.

Вимоги до системи, де буде проведено тестування клієнтської частини

- операційна система не менше ніж Windows 8
- центральний процесор не менше ніж 2.4 GHz Dual Core processor
- об’єм пам’яті не менше ніж 2GB

Вимоги до системи, де буде проведено тестування серверної частини

- операційна система не менше ніж Windows 8
- центральний процесор не менше ніж 2.4 GHz Dual Core processor
- об’єм пам’яті не менше ніж 2GB
- встановлена .Net v6.0

Система, де проводитиметься тестування обох частин в інтеграції повинна задовольняти усім переліченим вище умовам.

3.3 Функціональність, що підлягає тестуванню

В ході тестування має бути перевірено коректність роботи усієї функціональної складової програмного забезпечення. Функціонал додатку,

який має бути протестований та опис деталей тестування наведено у наступних таблицях:

– прослуховування повідомлень визначених протоколів за вказаними портами (таблиця 3.1);

– створення правил обробки перехоплених повідомлень (таблиця 3.2);

– створення та менеджмент фільтрів у межах правила (таблиця 3.3);

– створення та менеджмент дій у межах правила (таблиця 3.4);

– коректність роботи створених фільтрів та дій (таблиця 3.5);

– парсинг повідомлення відповідно до стандарту RFC5424 (таблиця 3.6).

Таблиця 3.1 – Прослуховування повідомлень визначених протоколів за вказаними портами

Мета тесту	Перевірити можливість перехоплювати повідомлення надіслані протоколами UDP там TCP за визначеними портами.
Початковий стан	Syslog сервер запущено
Вхідні дані	Порт для прослуховування UDP повідомлень (за замовчуванням 514) та TCP (1468 за замовчуванням).
Схема проведення тесту	Необхідно створити дію для виводу отриманого повідомлення на дашборд. Потім надіслати тестове повідомлення за допомогою генератору Syslog повідомлень.
Очікуваний результат	Інформація про отримане повідомлення буде виведено у консоль додатку. Деталі повідомлення буде відображено у дашборді користувача.

Продовження таблиці 3.1

Стан програмного забезпечення після закінчення тестування	Відкрито екран статистики мережі.

Таблиця 3.2 – Створення правил обробки перехоплених повідомлень

Мета тесту	Перевірити коректність створення правил для обробки перехоплених повідомлення
Початковий стан	Syslog сервер запущено
Вхідні дані	Назва правилу
Схема проведення тесту	У меню вкладки Setup обрати пункт New Rule, ввести назву правилу, натиснути далі та зберегти.
Очікуваний результат	У меню правил з’явиться нове правило із назвою, яка була задана при створенні.
Стан програмного забезпечення після закінчення тестування	Відкрито екран із правилами, нове створене правило є серед них у переліку.

Таблиця 3.3 – Створення фільтрів

Мета тесту	Перевірити коректність створення фільтру у межах правила
Початковий стан	Syslog сервер запущено, розпочато процес створення правилу
Вхідні дані	Назва правилу, назва та тип фільтру, деталі фільтрі в залежності від типу
Схема проведення тесту	У меню вкладки Setup обрати пункт New Rule, ввести назву правилу, натиснути далі, додати фільтр, обрати тип, ввести назву та деталі в залежності від типу, зберегти

Продовження таблиці 3.3

Очікуваний результат	У переліку правил з’явиться новостворене правило, у його деталях можна побачити саме той фільтр
Стан програмного забезпечення після закінчення тестування	Відкрито екран із правилами, нове створене правило із визначеним фільтром є серед них у переліку.

Таблиця 3.4 – Створення дій

Мета тесту	Перевірити коректність створення дій у межах правилу
Початковий стан	Syslog сервер запущено, розпочато процес створення правилу
Вхідні дані	Назва правилу, назва та тип дії, деталі в залежності від типу
Схема проведення тесту	У меню вкладки Setup обрати пункт New Rule, ввести назву правилу, натиснути далі, пропустити створення фільтрів, додати дію, обрати тип, ввести назву та деталі в залежності від типу, зберегти
Очікуваний результат	У переліку правил з’явиться новостворене правило, у його деталях можна побачити саме ту дію, яка була створена на попередньому кроці
Стан програмного забезпечення після закінчення тестування	Відкрито екран із правилами, нове створене правило із визначеним фільтром є серед них у переліку.

Таблиця 3.5 – Перевірка роботи фільтрів та дій

Мета тесту	Перевірити коректність роботи фільтру та дії у межах одного правилу
Початковий стан	Syslog сервер запущено
Вхідні дані	Назва правилу «display msg», тип фільтру – за IP-адресою, тип дії – додати до дисплею
Схема проведення тесту	У меню вкладки Setup обрати пункт New Rule, ввести назву правилу, натиснути далі, додати фільтр, обрати тип за IP-адресою, вказати адресу пристрою, з якого буде надіслано тестове повідомлення, додати дію, обрати тип «add to display buffer», зберегти. Надіслати тестове повідомлення за допомогою Syslog генератору.
Очікуваний результат	На дашборді з’являться деталі надісланого повідомлення.
Стан програмного забезпечення після закінчення тестування	Сервер запущено, відкрито дашборд користувача, на ньому відображено деталі надісланого повідомлення.

Таблиця 3.6 – Парсинг повідомлення відповідно до стандарту

Мета тесту	Перевірити коректність парсингу повідомлень
Початковий стан	Syslog сервер запущено
Вхідні дані	Повідомлення, надіслане syslog генератором, яке відповідає стандарту RFC 5424
Схема проведення тесту	Створити правило без фільтрів з дією “add to display buffer”. Надіслати повідомлення

Продовження таблиці 3.6

Очікуваний результат	У консолі серверу відображено інформацію про те, що повідомлення було перехоплене, відображено деталі, що було розпаршено. У дашборді у кожному стовпчику наявне відповідне значення з перехопленого повідомлення.
Стан програмного забезпечення після закінчення тестування	Сервер запущено, відкрито дашборд користувача, на ньому відображено деталі надісланого повідомлення.

3.4 Функціональність, що не підлягає тестуванню

– інклюзивність додатку, тобто доступність для людей с вадами, що обмежують можливості;

– поріг входу до користування – я швидко користувач може здобути навички вільного володіння та користування застосунком;

– виправданість UI-UX рішень – наскільки доцільно сконструйовано зовнішній вигляд додатку та наскільки зручно їм користуватися.

3.5 Методологія тестування

Необхідно перевірити коректність роботи програмного забезпечення згідно опису кожного test case’у. Процес проведення тестування та результати буде залоговано та збережено, аби мати можливість ітеративно відслідковувати стан програмного забезпечення з погляду на тести. Виконання тестових сценаріїв та запис результатів їх перебігу автоматизовано за допомогою інструментів CI-CD у GitHub actions. Після кожного коміту запускається процес компіляції та запуск усіх тестів у рішенні. Після виконання усіх тестів формується звіт про деталі перебігу кожного.

Як тільки тестування завершено, необхідно переглянути звіт аби мати розуміння поточної ситуації та відповідно до результатів тестування та деталей провести коригування поведінки тестів або програмного забезпечення.

3.6 Критерії проходження чи провалу тестування

Базова функціональність для прослуховування повідомлень має бути присутньою та якісно протестованою на позитивні та негативні сценарії. Користувач повинен мати можливість створювати правил, фільтрів, дії та конфігурувати їх. Деталі роботи фільтрів та виконання дій мають бути перевірені досконально. Є неприпустимим невиконання очікуваних функцій жодною одиницею з переліченого функціоналу.

Користувач повинен мати можливість редагувати налаштування деталей прослуховування повідомлень у мережі. Проте ця функціональність не має вирішального значення для роботи застосунку з боку користувача.

Мають бути успішно виконані не менше ніж 95% усіх тестових випадків. Ні один із випадків, які показали поведінку, протилежну чи інакшу від очікуваної, не повинні мати суттєвого значення для можливості використання програмного застосунку кінцевим користувачем.

3.7 Критерії припинення тестування

Потрібно припинити тестування у випадку, коли система не виконує перехоплення повідомлень, не виконує парсинг відповідно до стандарту, правила, фільтри чи дії не створюються чи не модифікуються або не можуть бути виконані.

3.8 Вимоги до тестового середовища

Характеристики тестового середовища повинні відповідати мінімальним нефункціональним вимогам додатку, а саме: операційна система версії

Windows не менш ніж 8, центральний процесор з частотою не менш ніж 2.4 GHz, не менше ні 2 GB вільної пам’яті та встановлена версія .NET 6.0

3.9 Аналіз якості програмного забезпечення

Критерії, за якими зручно проаналізувати якість програмного забезпечення

– за функціональністю:
  а) захищеність програми;
  б) придатність;
– з погляду на надійність:
  а) відмовостійкість;
  б) завершеність
– з боку зручності використання:
  а)доступність користувацького інтерфейсу;
  б) можливість вивчати інтерфейс програми;
– з погляду на можливість супроводжувати:
  а) гнучкість використання та розширення;
  б) придатність до тестування.

У таблиці 3.7 наведено відношення впливу перелічених критеріїв якості програмного забезпечення

Таблиця 3.7 – Відношення впливу критеріїв оцінки якості

Характеристика впливу / Характеристика Росту якості		Функціональність		Надійність	
		Придатність	Захищеність	Відмовостійкість	Завершеність
Функціональність	Придатність	+	-	-	+
	Захищеність	-	+	+	-
Надійність	Відмовостійкість	-	+	+	-
	Завершеність	+	-	-	+
Зручність використання	Доступність	-	-	-	+
	Можливість навчатися	+	-	-	-
Можливість супроводжувати	Гнучкість	-	-	-	-
	Придатність до тестування	-	-	-	-

Продовження таблиці 3.7

Характеристика впливу / Характеристика / Росту якості		Зручність використання		Супроводжуваність	
		Зрозумілість	Доступність	Гнучкість	Придатність до тестування
Функціональність	Придатність	-	+	-	+
	Захищеність	-	-	-	+
Надійність	Відмовостійкість	-	-	-	-
	Завершеність	-	-	-	+
Зручність використання	Зрозумілість	+	+	+	-
	Можливість навчатися	+	+	-	-
Супроводжуваність	Гнучкість	-	-	+	-
	Тестованість	+	-	-	-

Таблиця 3.8 - Відображення моделі зовнішньої якості на експлуатаційну якість

Характеристика впливу / Характеристика Росту якості		Експлуатаційна якість			
		Продуктивність	Задоволеність	Результативність	Безпечність
Функціональність	Придатність	+	+	+	-
	Захищеність	-	+	-	-
Надійність	Відмовостійкість	-	-	-	-
	Завершеність	+	-	-	-
Зручність використання	Зрозумілість	-	+	-	-
	Можливість навчатися	-	+	-	-

Висновки до розділу

У третьому розділі було складено план, за яким необхідно провести тестування програмного забезпечення з моніторингу, обробки та класифікації логів у мережі. Виконання тестування за планом надасть можливість виявити дефекти та проблеми у розроблюваному додатку до того, як доступ отримають кінцеві користувачі. Тобто тестування вигідно з економічної точки зору насамперед тим, що допомагає запобігти значній втраті активних користувачів на стадії першого релізу.

Було проаналізовано та описано функціональність, яка підлягає тестуванню та функціональні особливості, які тестуванню не підлягають. Крім того визначено методологію тестування та критерії проходження, провалу чи припинення тестування. Також було описано вимоги до тестового середовища та проведено аналіз якості програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для того, щоб мати можливість виконати розгортання програмного забезпечення необхідно мати ноутбук чи персональних комп'ютер з операційною системою Windows версії не менше ніж 8, центральним процесором з частотою не менше ніж 2.4 GHz та не менше ніж 2 Gb пам'яті. Крім того треба встановити платформу .NET 6.0 та файл інсталяції з розширенням .exe або .msi.

Після подвійного натискання на файл інсталяції з'явиться вікно початку установки (рисунок 4.1)

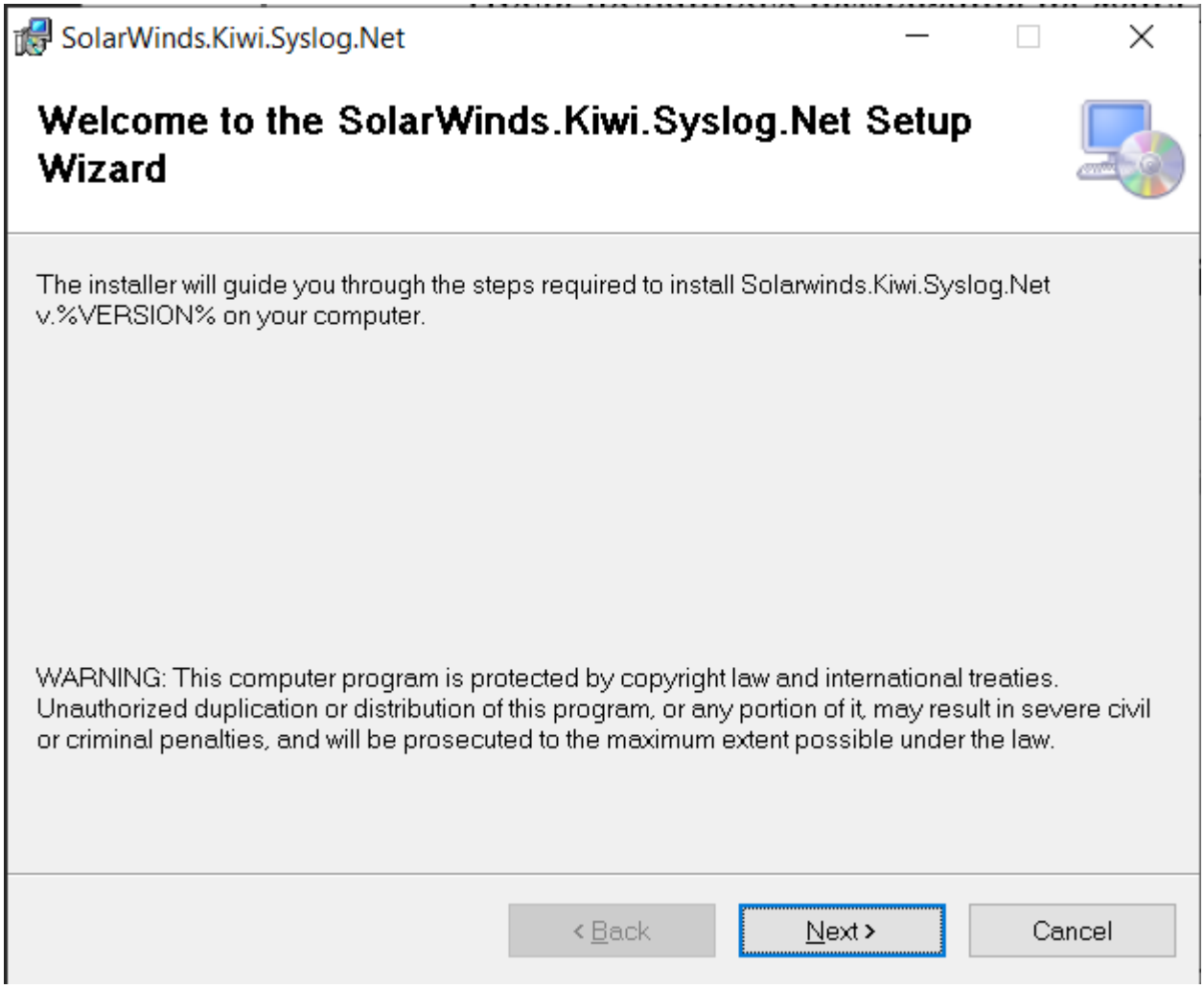


Рисунок 4.1 – Вікно початку установки Syslog Server

Наступним кроком необхідно обрати шлях установки застосунку. На рисунку 4.2 зображено форму вибору шляху установки додатку та користувачів, для яких буде встановлено додаток.

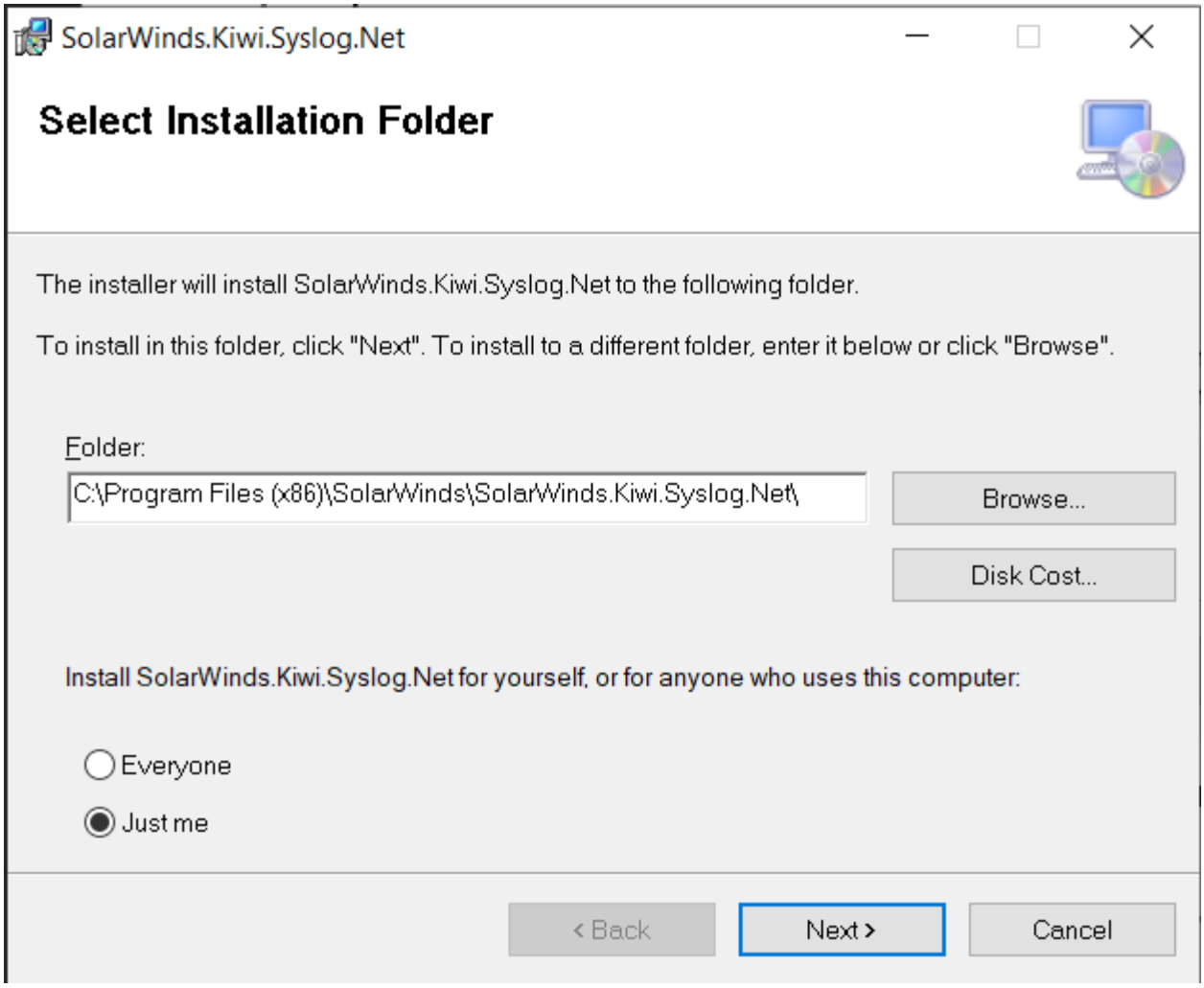


Рисунок 4.2 – Форма вибору шляху установки застосунку

Далі необхідно обрати режим, у якому буде встановлено додаток. На рисунку 4.3 зображено форму вибору режиму роботи застосунку: service mode – додаток працюватиме у режимі Windows сервісу, application mode – як звичайна аплікація.

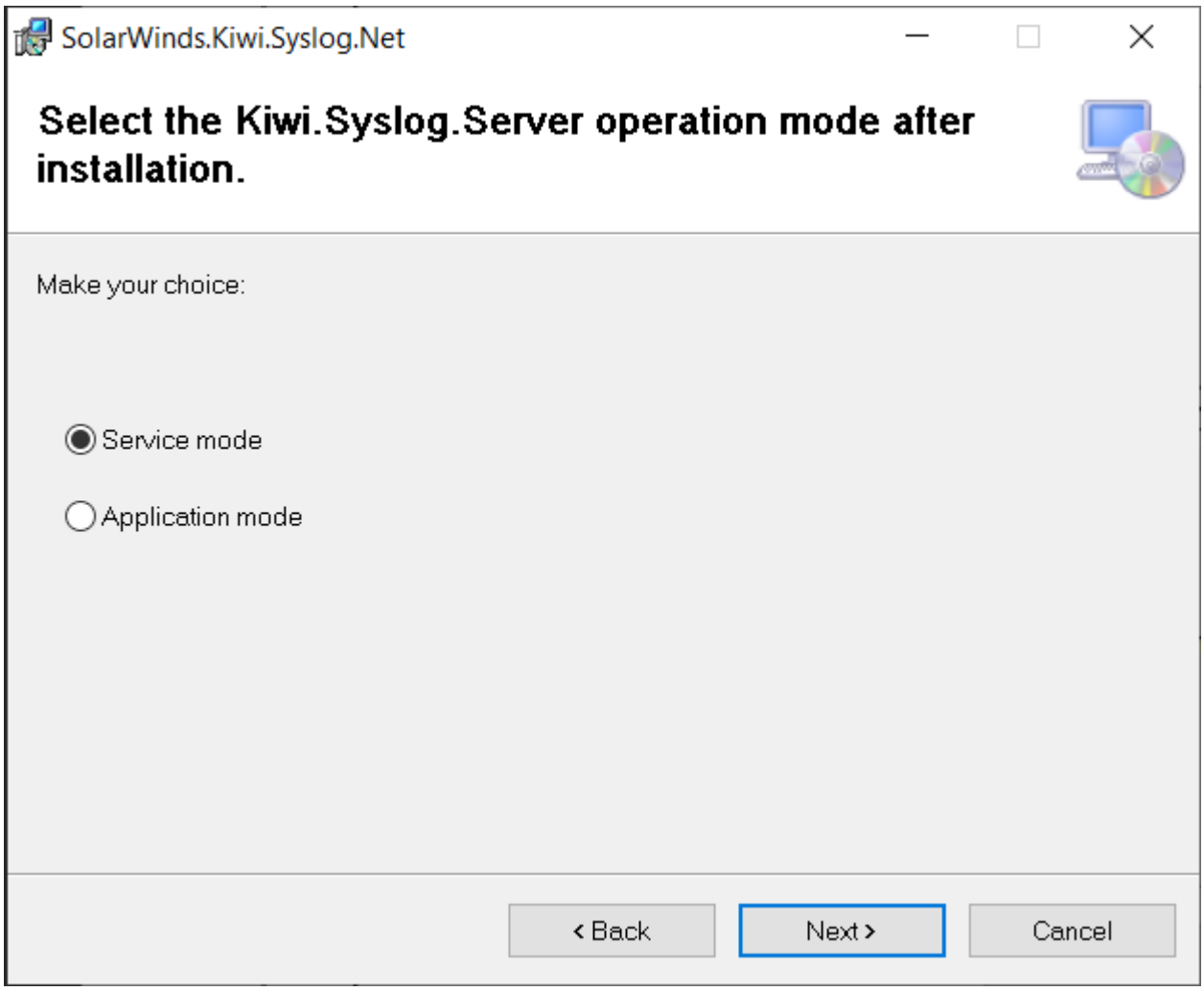


Рисунок 4.3 – Форма вибору режиму роботи застосунку

На наступній формі необхідно підтвердити рішення щодо процесу інсталяції, які були зроблені раніше (рисунок 4.4).

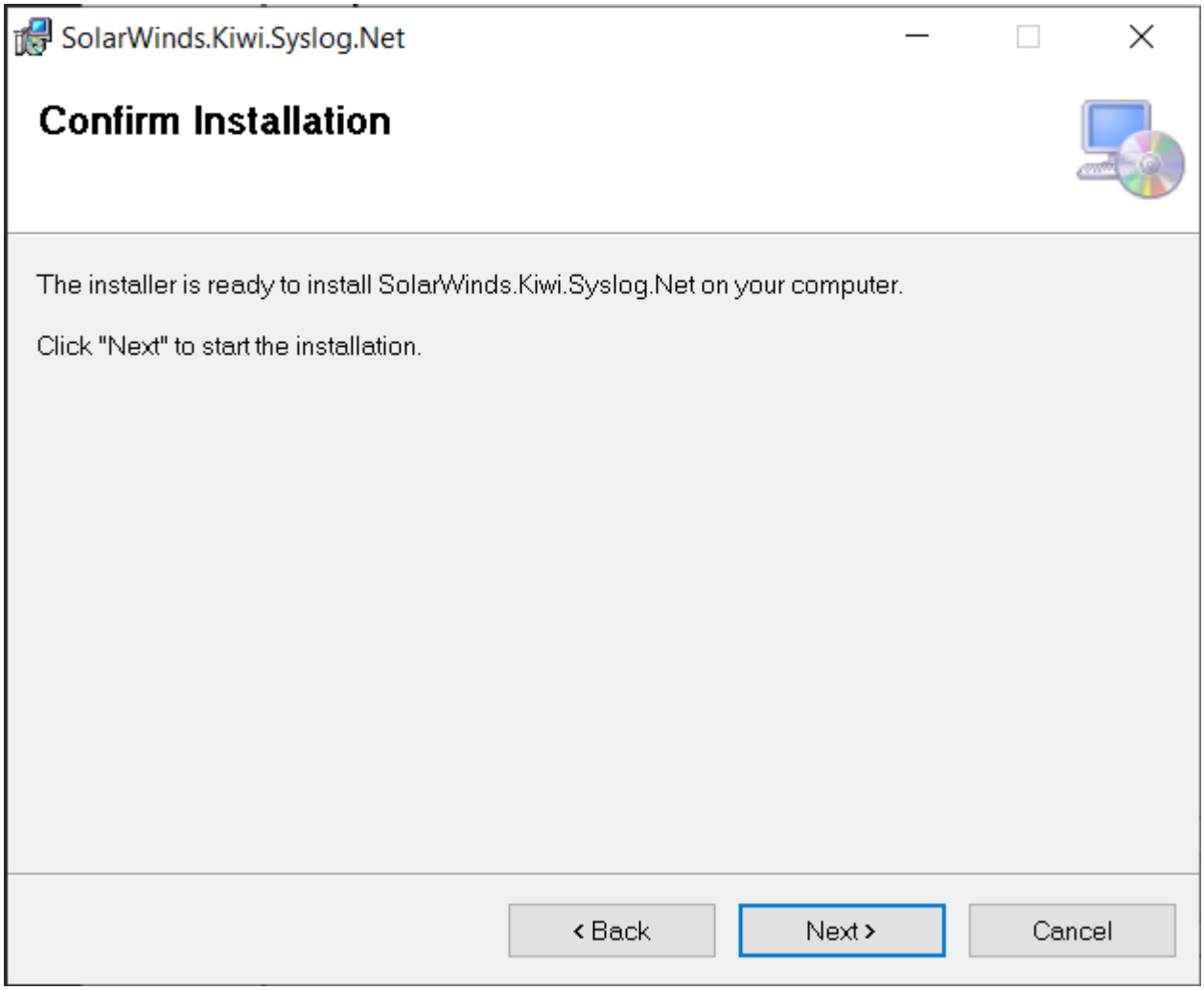


Рисунок 4.4 – Вікно підтвердження рішень про установку

Після натискання кнопки Next з’явиться вікно з прогрес баром процесу установки (рисунок 4.5).

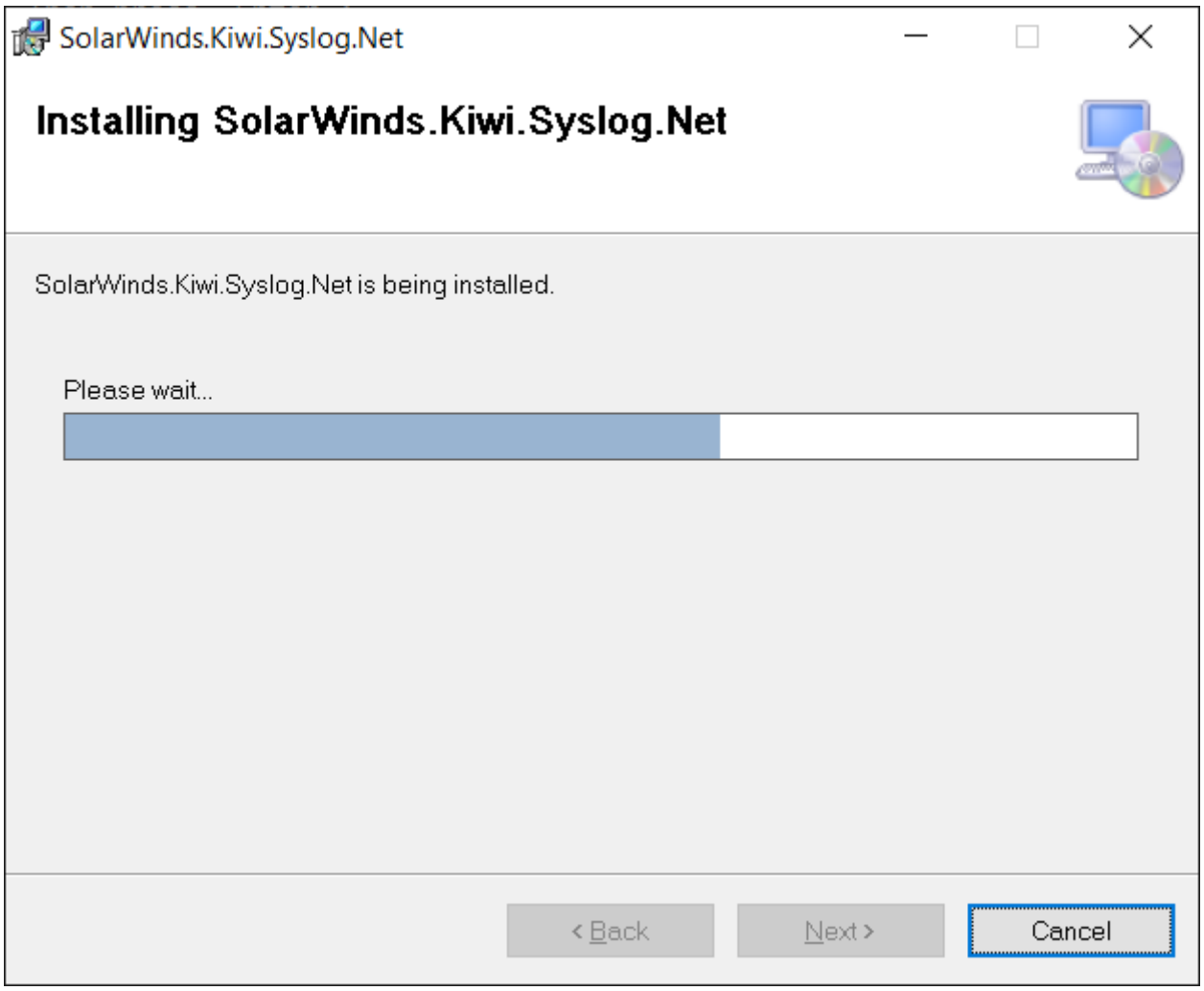


Рисунок 4.5 – Відображення прогресу процесу інсталяції

Під час безпосередньо установки та створення системних файлів з налаштуваннями за замовчуванням необхідно задати пароль адміністратора системи. На рисунку 4.6 зображено консоль вводу паролю адміністратора системи та вимоги до самого паролю.

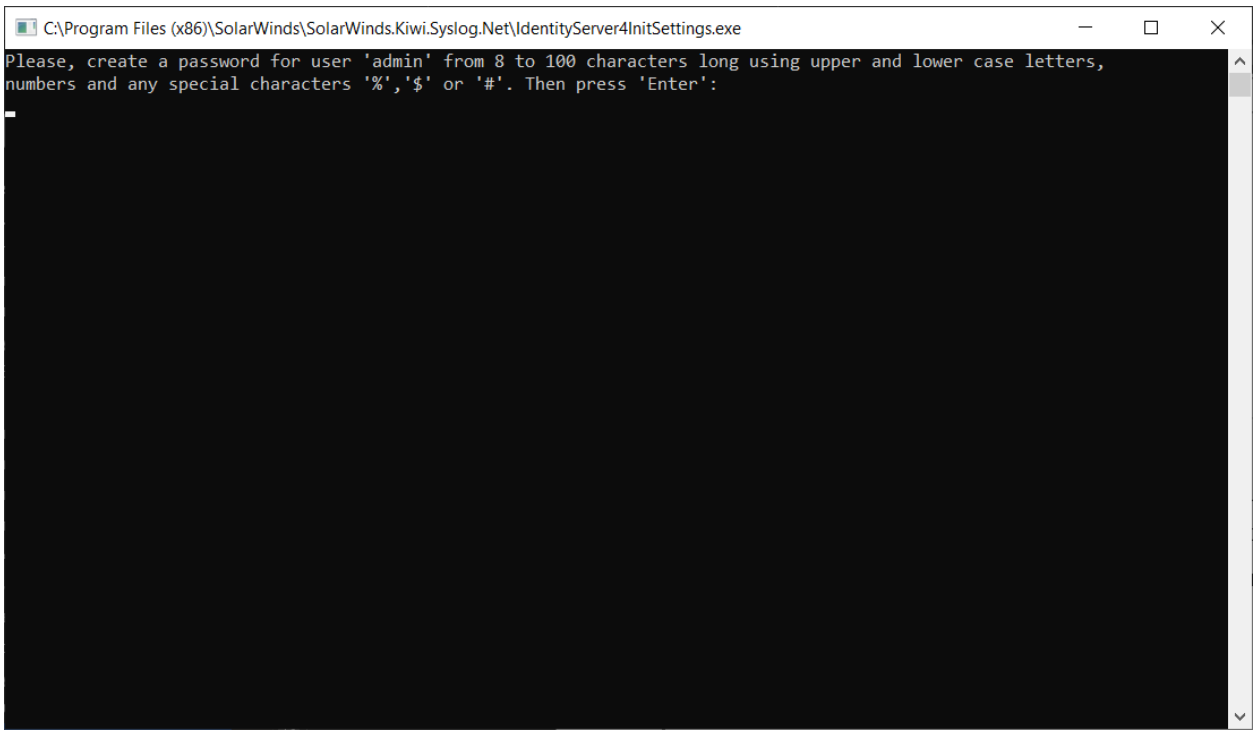


Рисунок 4.6 – Консоль вводу паролю адміністратора системи

Після введення паролю процес встановлення завершиться та на екрані з’явиться вікно, яке сповістить про успішну установку (рисунок 4.7).

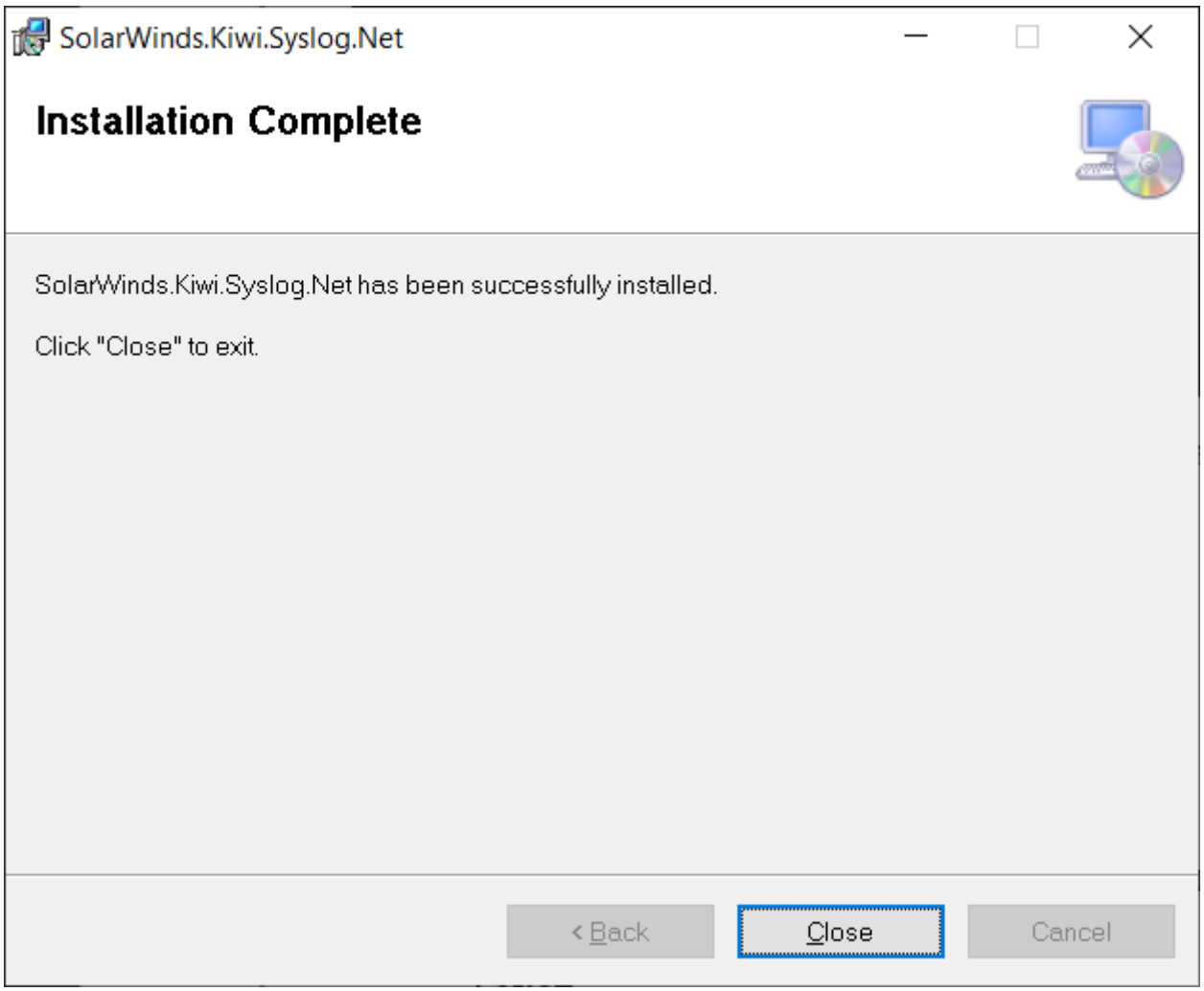


Рисунок 4.7 – Вікно завершення процесу встановлення

Наразі додаток встановлено, тож можна запускати сервер та повністю його адаптувати під користувача. Про те, що застосунок встановлено, можна впевнитися в меню Apps&features.

Після запуску серверу на екрані з’являється консоль з інформацією про старт роботи. На рисунку 4.8 зображено консоль с сервісними логами про те, що сервіс запущено, розпочато прослуховування повідомлень протоколами UDP та TCP за портами за замовчуванням.

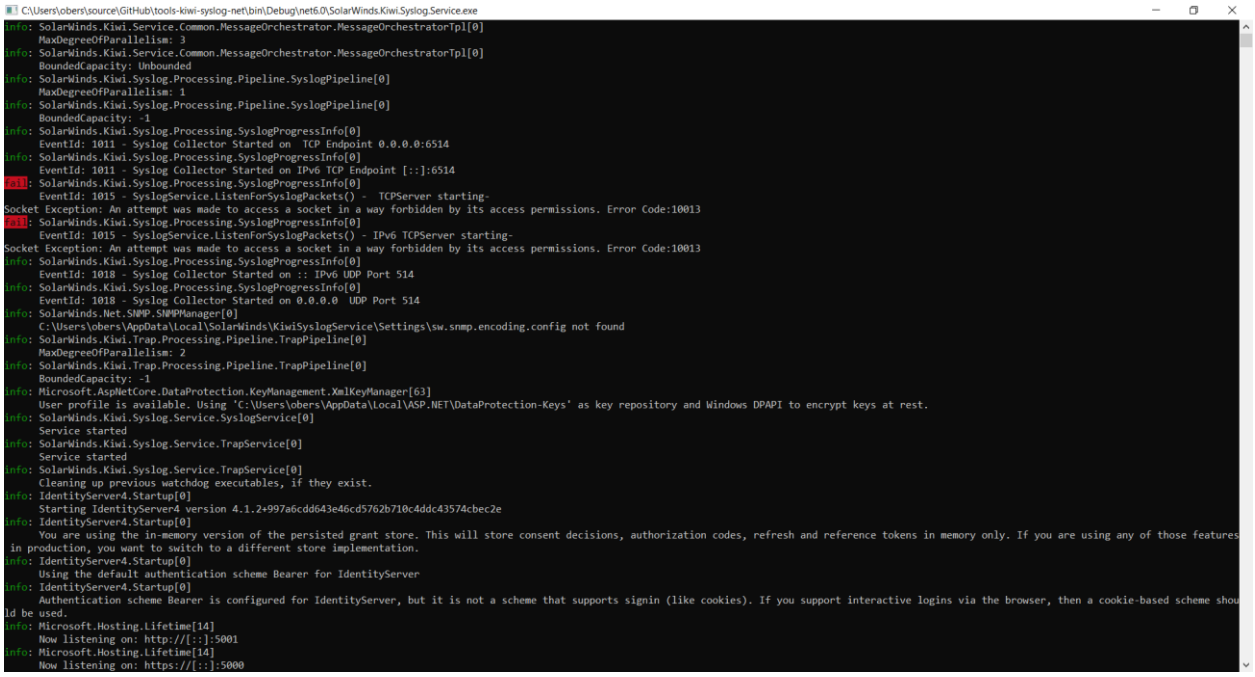


Рисунок 4.8 – Консоль серверу після старту додатку

4.2 Робота з програмним забезпеченням

Для того, щоб розпочати роботу із застосунком необхідно після встановлення запустити серверну частину. У базовому розумінні додаток працює одразу як тільки запущено сервер: порти прослуховуються, повідомлення можуть бути перехоплені, правила будуть виконуватися.

Проте для того, щоб розширити або змінити налаштування за замовчуванням необхідно кастомізувати додаток за допомогою користувацького інтерфейсу. Тож треба дві натиснути на ярлик на робочому столі. У браузері за замовчуванням відкриється вікно авторизації у застосунку. На рисунку 4.9 зображено вікно авторизації користувача у системі.

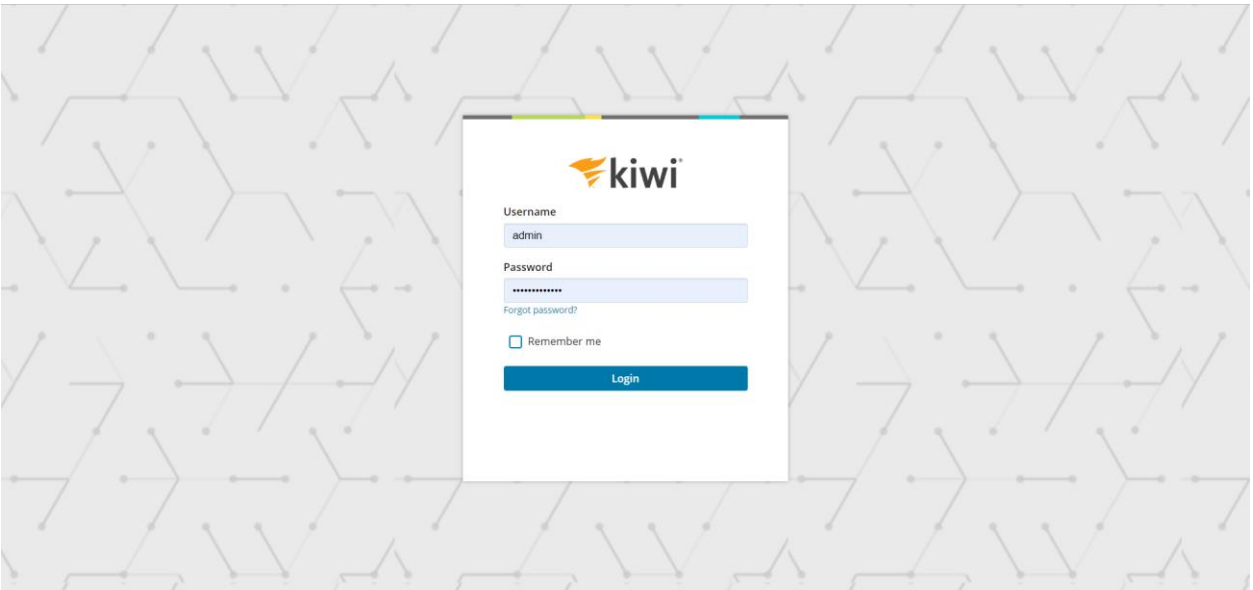


Рисунок 4.9 – Сторінка авторизації

Після вдалого проходження авторизації користувач потрапляє на сторінку зі статистичним дашбордом. На рисунку 4.10 зображено приклад дашборду, який може бачити користувач.

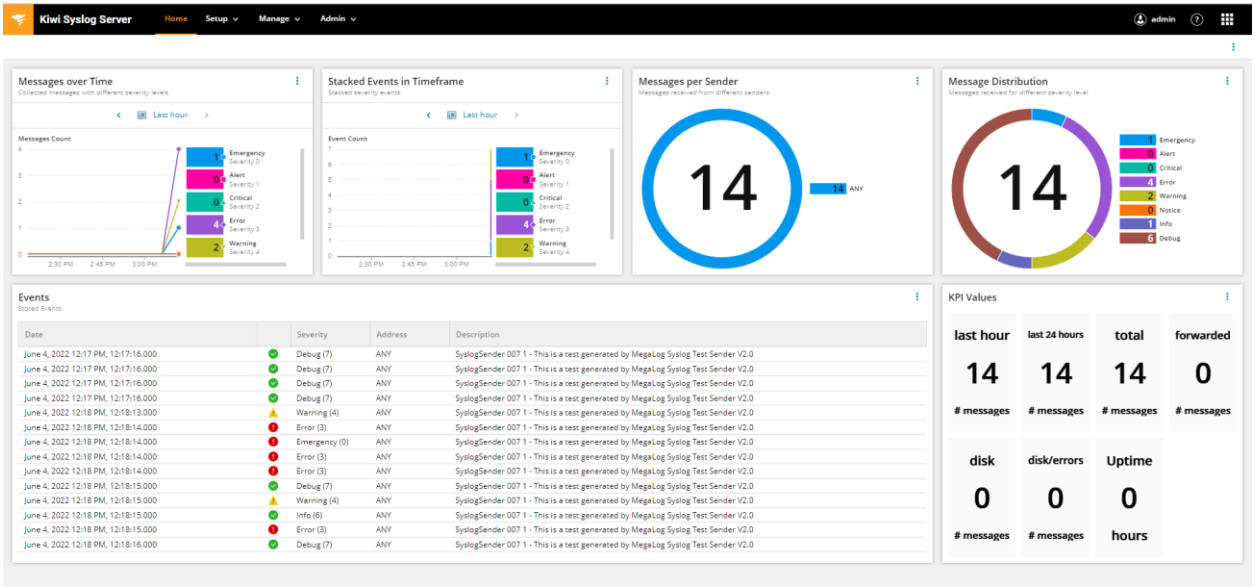


Рисунок 4.10 – Панель статистики

Користувач має можливість редагувати формат відображення інформації на дашборді та формат графіків. Аби редагувати дашборд необхідно натиснути на три крапки у верхньому правому куті та активувати

режим Edit dashboard. На рисунку 4.11 зображено можливість редагувати панель статистичної інформації.

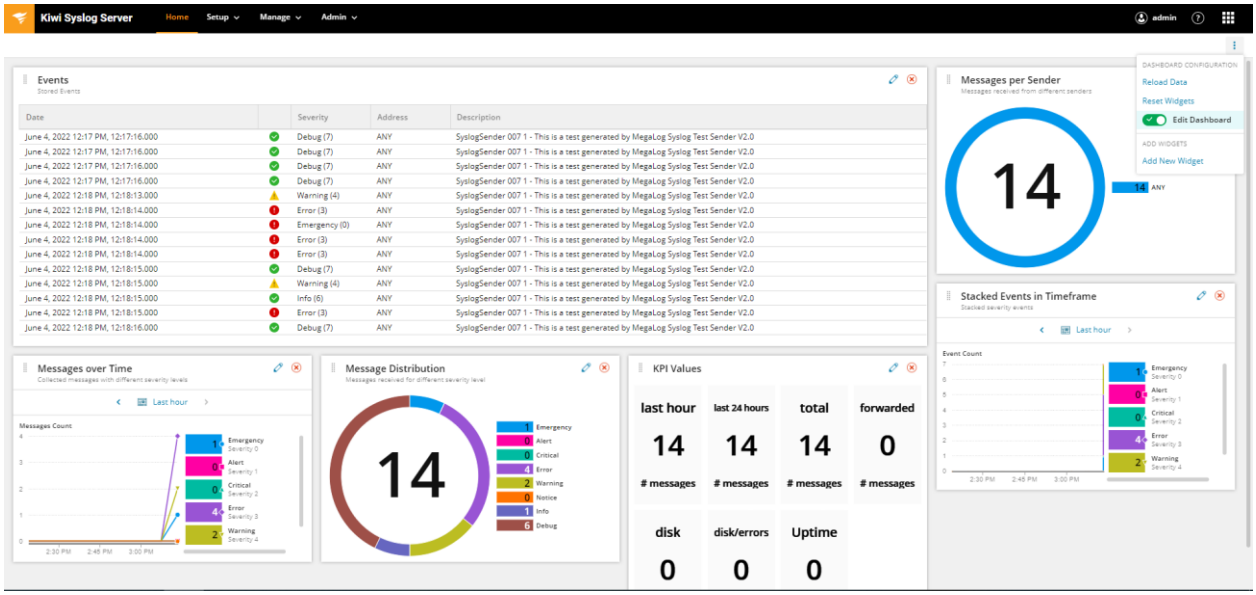


Рисунок 4.11 – Результат редагування вигляду дашборду

Для того, щоб перехоплене повідомлення було залоговано до файлу, необхідно створити правило. Аби створити правило, треба перейти у вкладку “New rule” у верхньому меню “Setup”(рисунок 4.12), вказати назву та натиснути далі.

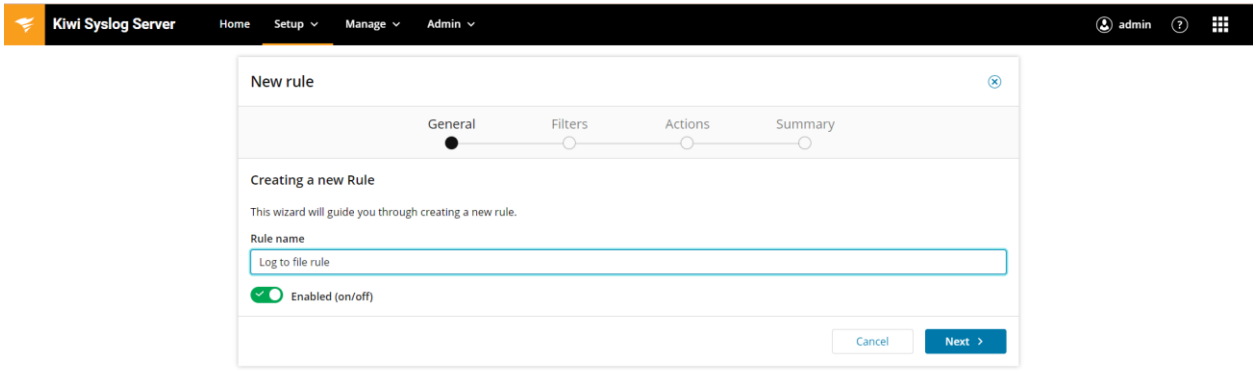


Рисунок 4.12 – Створення нового правила

Аби створити фільтр необхідно у меню створення фільтрів (рисунок 4.13) обрати “Add filter”, тип фільтру та ввести деталі в залежності від типу. На рисунку 4.14 зображено вікно деталей створюваного фільтру.

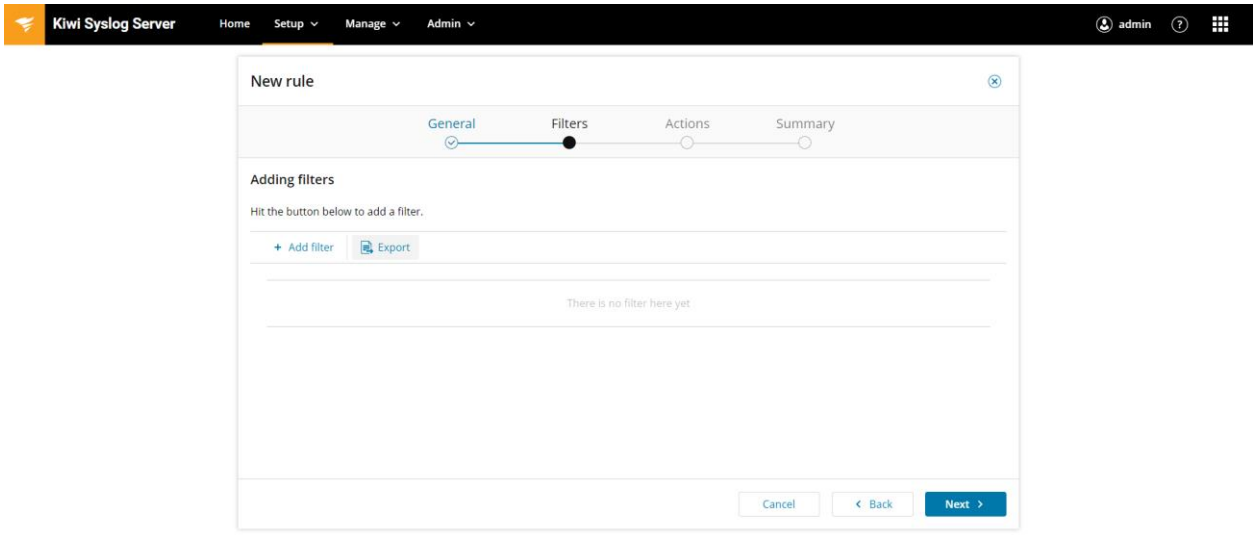


Рисунок 4.13 – Вікно створення фільтрів

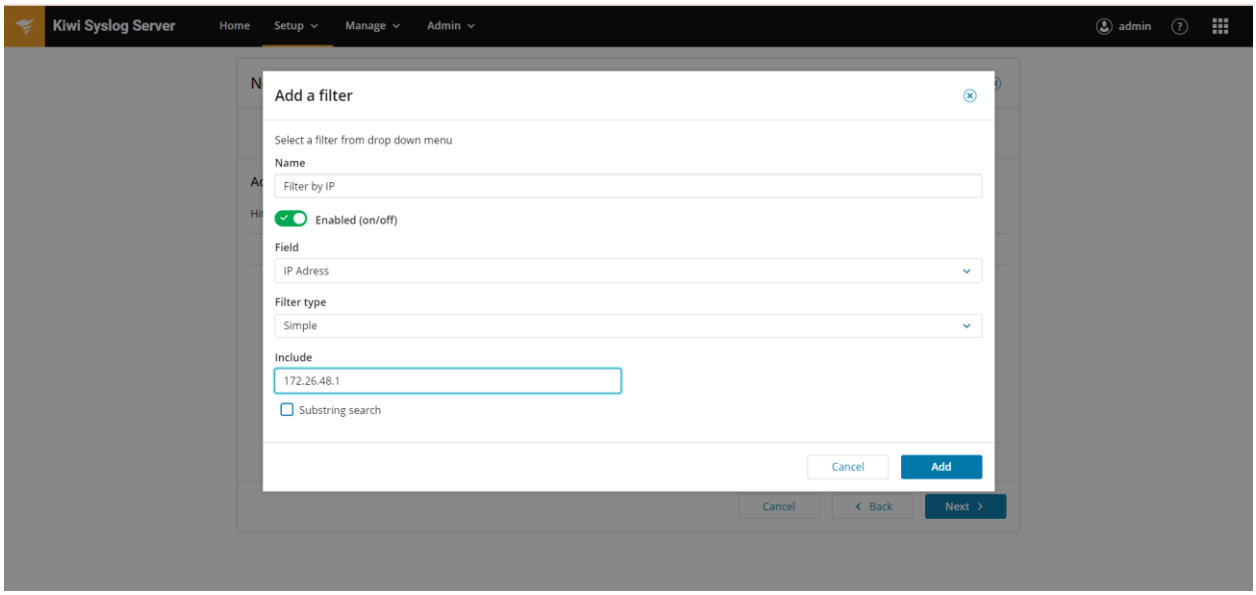


Рисунок 4.14 – Вікно деталей створюваного фільтру

На рисунку 4.15 зображено менеджер правил після створення фільтру.

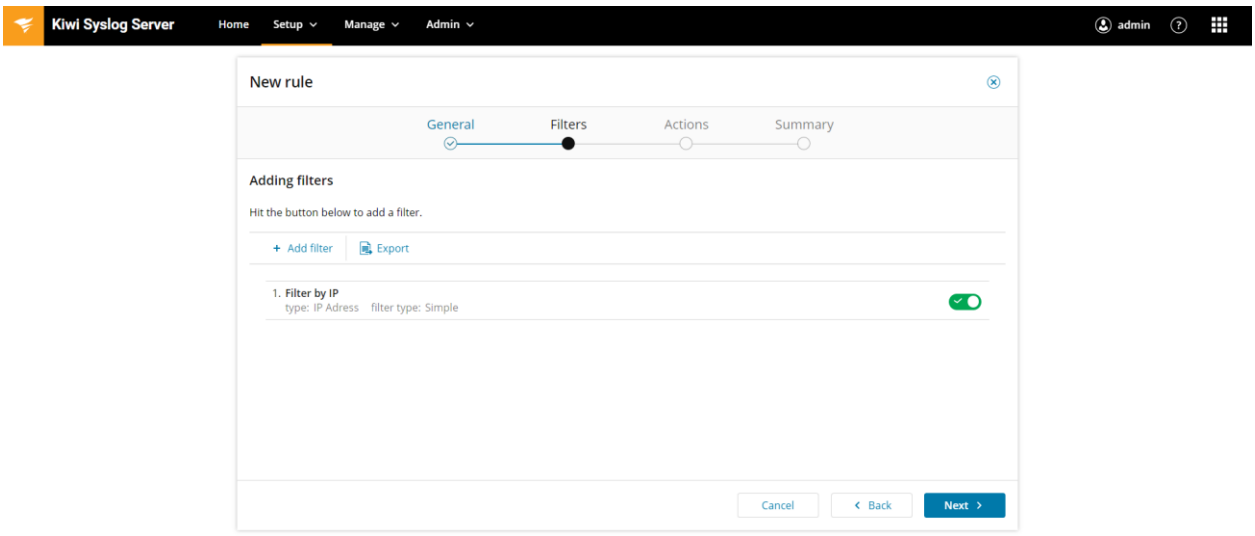


Рисунок 4.15 – Менеджер правил після створення фільтру

Фільтрів може бути додано будь-яка кількість різних типів – за IP-адресою, за ім'ям хоста, за текстом повідомлення чи його частиною, за протоколом, за пріоритетом чи часом доби. На рисунку 4.17 зображено перелік типів фільтрів, які можуть бути створені.

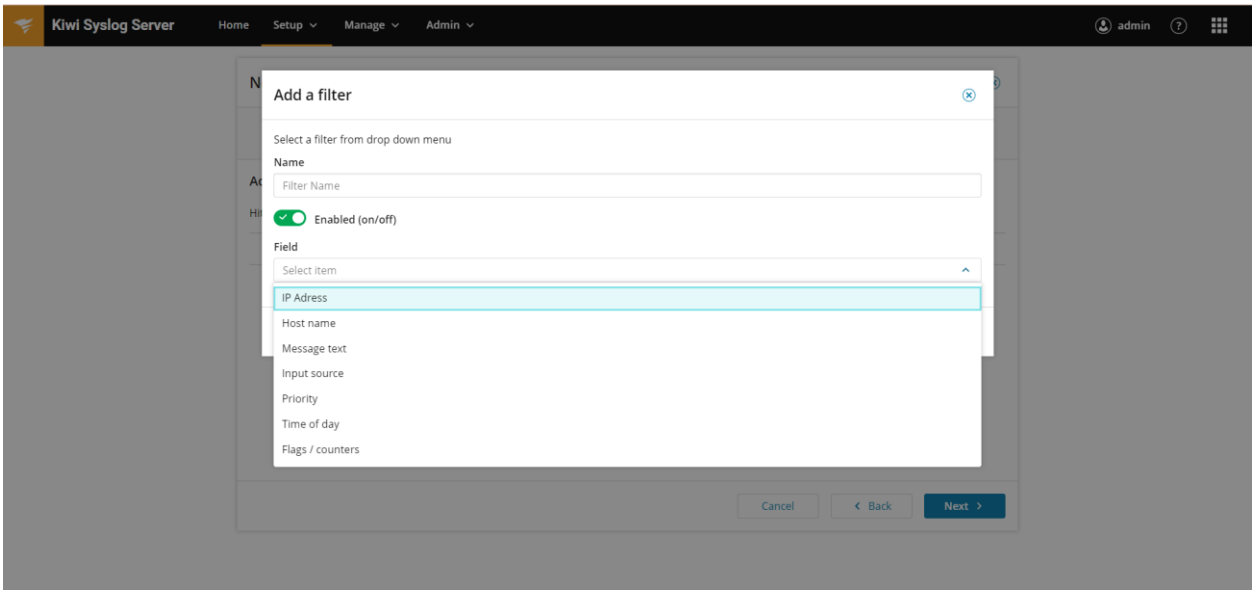


Рисунок 4.16 – Перелік типів фільтрів

Наступним у менеджері правил є вікно створення дії (рисунок 4.17). Серед наявних дій є логування у текстовий файл, пересилання повідомлення на інший сервер, надсилання e-mail повідомлення, виконання PowerShell скрипту, надсилання нового Syslog повідомлення з певними властивостями та логування до Event Log'у.

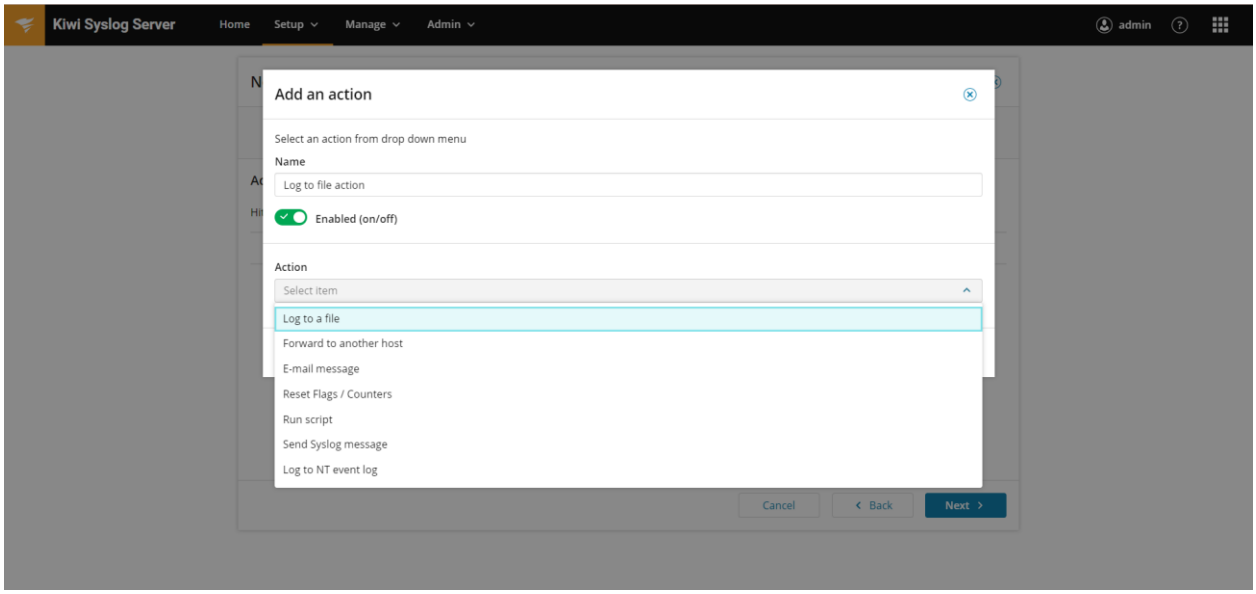


Рисунок 4.17 – Перелік можливих дій

З випадаючого списку необхідно обрати пункт Log to a file та заповнити відповідні деталі. Для цієї дії необхідно вказати шлях до файлу та формат запису інформації. На рисунку 4.18 наведені можливі формати логування повідомлень.

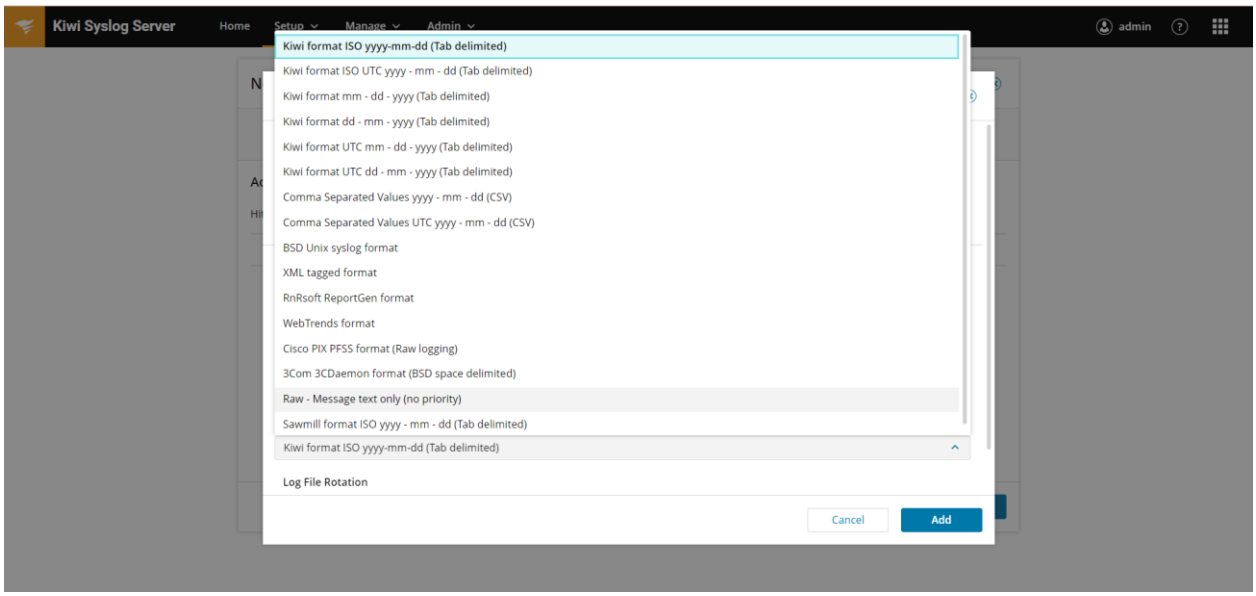


Рисунок 4.18 – Формати логування повідомлень у файл

На рисунку 4.19 наведено деталі створюваної дії.

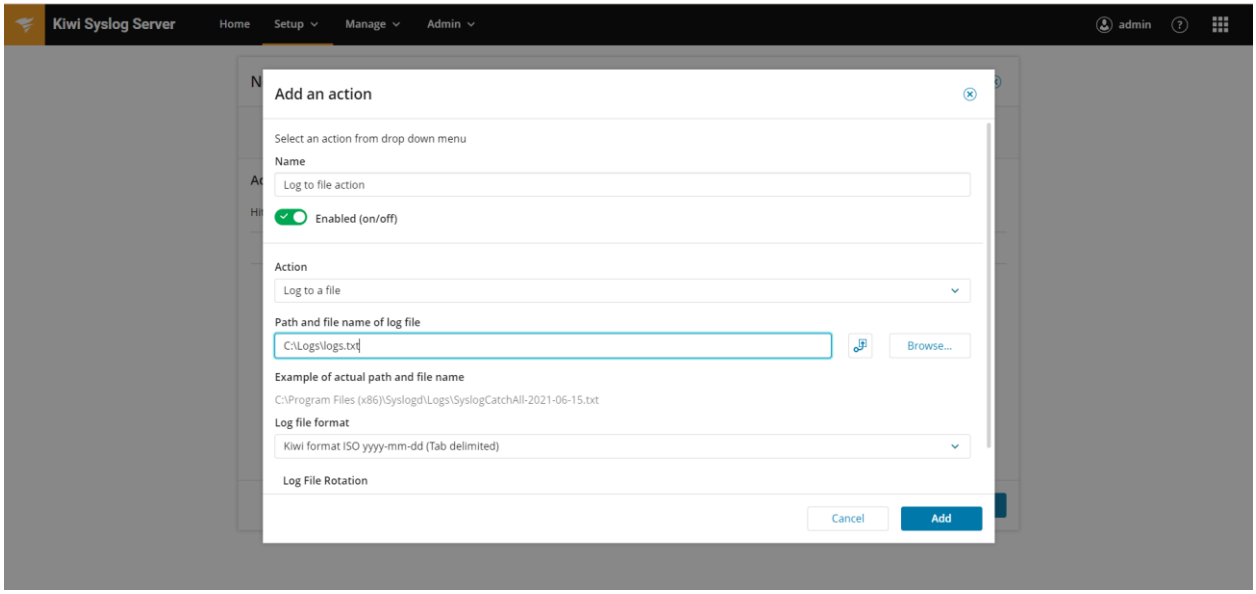


Рисунок 4.19 – Деталі дії Log to file

Після додання дії та натискання кнопки Next буде відображено менеджер створення правил. Необхідно підтвердити всі внесені зміни та створити правило. На рисунку 4.20 зображено менеджер створення правил перед збереженням кінцевих змін.

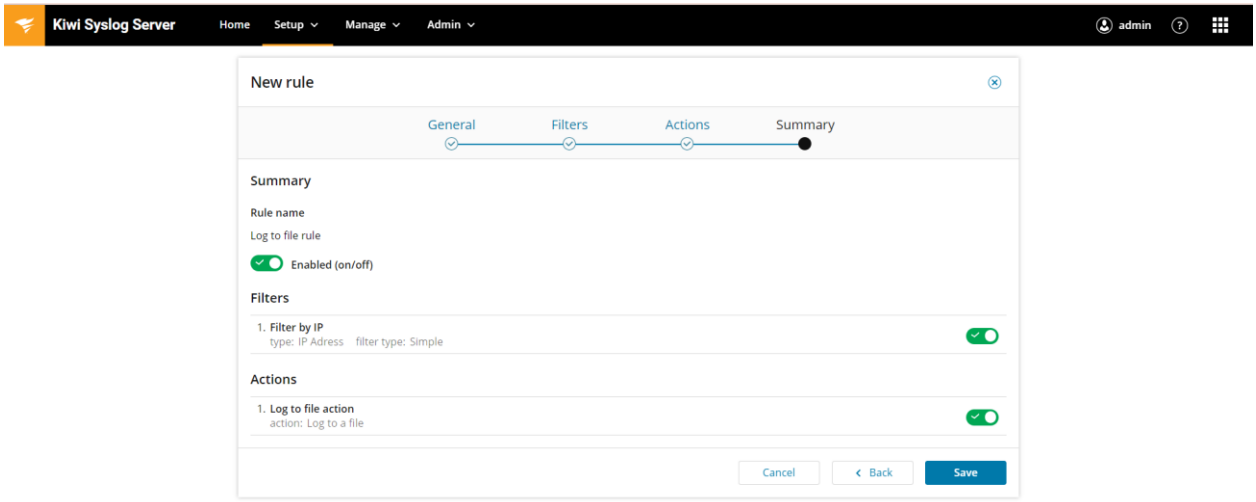


Рисунок 4.20 – Менеджер створення правил перед збереженням змін

Створене правило з’явиться у зальному списку правил (рисунок 4.21)

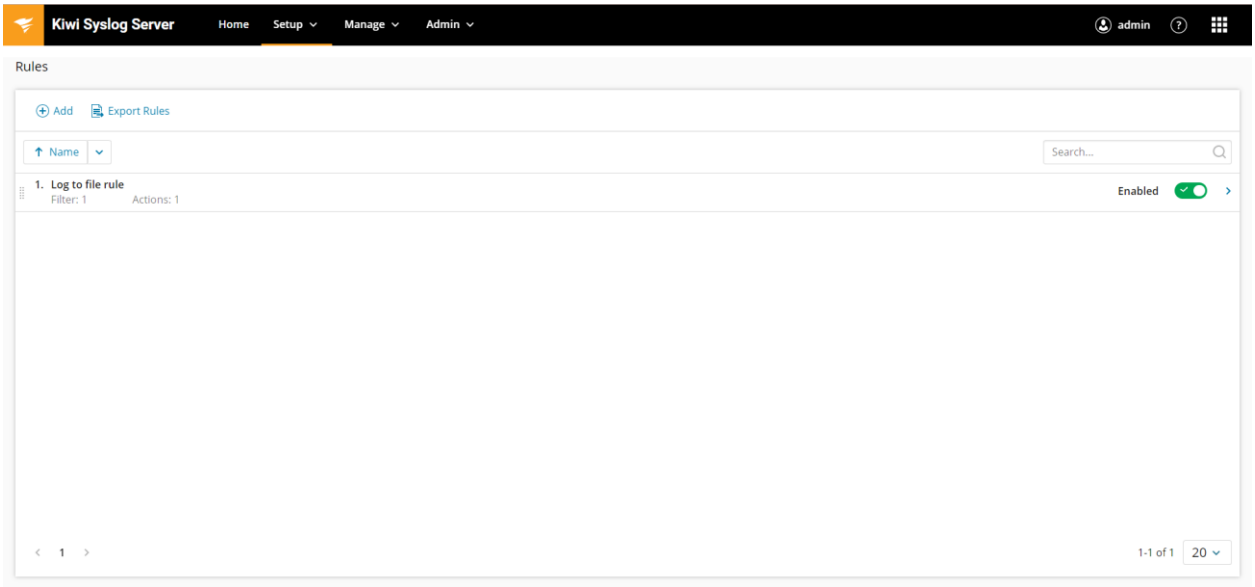


Рисунок 4.21 – Створене правило у загальному списку правил

Для того, аби перевірити працездатність створеного правила, необхідно надіслати тестове повідомлення. Для цього можна використовувати сторонній додаток. Було надіслано 5 тестових повідомлень. На рисунку 4.22 зображено результат роботи правила.

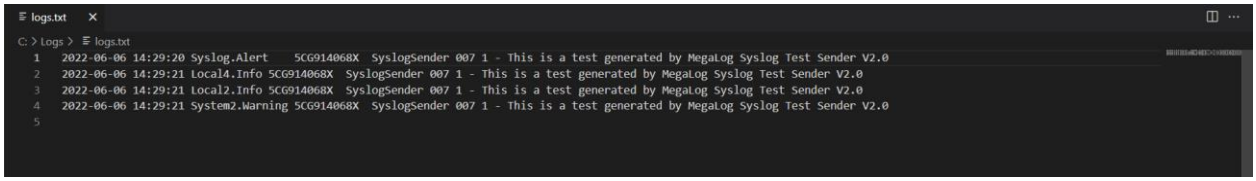


Рисунок 4.22 – Файл результату роботи створеного правилу

За аналогією можна створювати безліч інших правил із різноманітними наборами фільтрів та дії. Є можливість створити правило без фільтрів. В такому випадку дій, додані в межах цього правила, виконуватимуться для кожного перехопленого повідомлення.

Висновки до розділу

У четвертому розділі було описано процедуру розгортання програмного забезпечення. Вона є універсальної, будь-то локальних пристрій чи віддалений сервер. Було описано кроки встановлення додатку за допомогою інсталятору, деталі, які треба вказати в процесі інсталяції та очікуваний результат після

успішної установки. Також було вказано, як перевірити наявність встановленого застосунку після інсталяції.

Крім того було покроково описано один з багатьох можливий сценаріїв роботи з встановленим програмним забезпеченням.

ВИСНОВКИ

В ході роботи над дипломним проєктом було створено застосунок для моніторингу, аналізу, класифікації та обробки логів у локальній обчислювальній мережі. Програмне забезпечення можна використовувати для підтримки працездатності мережі підприємства. Крім того додаток прискорює процес усунення проблем у разі їх виникнення бо спрощує процес локалізації проблеми та знаходження першопричини.

У першому розділі описано і проаналізовано предметну область, а саме пристрої локальної обчислювальної мережі такі як концентратори, маршрутизатори, комутатори. Було викладено особливості Syslog, UDP и TCP протоколів, основні відмінності протоколів транспортного рівня та особливості використання. Також було проведено аналіз відомих технічних рішень та програмних продуктів, наявних на поточний час, їхніх особливостей та наявних функцій. Крім того бути висунуті функціональні та нефункціональні вимоги і докладно описані задачі розроблюваного програмного забезпечення.

У другому розділі описано процеси, які відбуватимуться в ході роботи програми такі як, перехоплення повідомлення, парсинг, перевірка на відповідність правилам, виконання дій. Проведено моделювання та аналіз програмного забезпечення та представлено у форматі діаграми. Також було описано архітектуру застосунку та обґрунтовано доцільність обраних рішень для реалізації API та сховища даних. Серверну частину застосунку було реалізовано на платформі .NET 6.0 мовою C#. Клієнтську частину програмного забезпечення реалізовано на платформі Angular. Представлено алгоритми кодування та шифрування, використані для забезпечення безпеки даних. Крім того було докладно описано основні методі та функції, які забезпечують виконання задачі програмного забезпечення.

У третьому розділі було складено план, за яким необхідно провести тестування програмного забезпечення з моніторингу, обробки та класифікації

логів у мережі. Виконання тестування за планом надасть можливість виявити дефекти та проблеми у розроблюваному додатку до того, як доступ отримають кінцеві користувачі. Тобто тестування вигідно з економічної точки зору насамперед тим, що допомагає запобігти значній втраті активних користувачів на стадії першого релізу. Було проаналізовано та описано функціональність, яка підлягає тестуванню та функціональні особливості, які тестуванню не підлягають. Крім того визначено методологію тестування та критерії проходження, провалу чи припинення тестування. Також було описано вимоги до тестового середовища та проведено аналіз якості програмного забезпечення.

У четвертому розділі було описано процедуру розгортання програмного забезпечення. Описано кроки встановлення додатку за допомогою інсталятору, деталі, які треба вказати в процесі інсталяції та очікуваний результат після успішної установки. Також було вказано, як перевірити наявність встановленого застосунку після інсталяції. Крім того було покроково описано один з багатьох можливий сценаріїв роботи з встановленим програмним забезпеченням.

У результаті роботи над дипломним проектом реалізовано вебзастосунок та серверну частину, яка має API, придатне до використання сторонніми сервісами. Розширення функціональності можливо за рахунок збільшення різноманіття фільтрів та дій, що можуть бути виконані над повідомленнями, або просто у мережі чи системі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Воробієнко П.П., Нікітюк Л.А., Резніченко П.І. Телекомунікаційні та інформаційні мережі. Вид-во «САММІТ-Книга», 2012. 246с.

2) Syslog [Електронний ресурс]: (Стаття) / The Syslog Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.rfc-editor.org/rfc/rfc5424. – Назва з екрану.

3) RFC 3164 [Електронний ресурс]: (Стаття) / The BSD Syslog Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://tools.ietf.org/html/rfc3164.

4) RFC 5424 [Електронний ресурс]: (Стаття) / The Syslog Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc5424

5) Таненбаум Е., Уезеролл Д. Комп'ютерні мережі. П'яте видання. Вид-во «Пітер», 2012. 575 с.

6) Таненбаум Е., Уезеролл Д. Комп'ютерні мережі. П'яте видання. Вид-во «Пітер», 2012. 586 с.

7) Таненбаум Е., Уезеролл Д. Комп'ютерні мережі. П'яте видання. Вид-во «Пітер», 2012. 61 с.

8) RFC 768 [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc768.

9) Kiwi Syslog Server [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.solarwinds.com/kiwi-syslog-server. – Назва з екрану.

10) EventLog Analyzer [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.manageengine.com/products/eventlog/. – Назва з екрану.

11) Nagios Log Server [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.nagios.com/products/nagios-log-server/. – Назва з екрану.

12) Paessler PRTG Network Monitor [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.paessler.com/prtg?gclid=CjwKCAjw77WVBhBuEiwAJ-YoJGO0wEogJf4YQTFhZrEgIK-9RkeHg-57Pc_X1RmvlZoenm3SQBLelxoC0HkQAvD_BwE. – Назва з екрану.

13) N-tier architecture [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://www.baeldung.com/cs/n-tier-architecture. – Назва з екрану.

14) GraphQL [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://graphql.org/. – Назва з екрану.

15) Воробієнко П.П., Нікітюк Л.А., Резніченко П.І. Телекомунікаційні та інформаційні мережі. Вид-во «САММІТ-Книга», 2012. 107с.

16) xUnit [Електронний ресурс]: (Стаття) / User Datagram Protocol – Електрон. дан. (1 файл) – 2009. – Режим доступу: https://xunit.net/. – Назва з екрану.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КЛАСИФІКАЦІЇ ЛОГІВ ЛОКАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ МЕРЕЖІ

Опис програми

КПІ.ІП-8104.045490.03.13

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юрій ОЛІЙНИК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Олександра БЕРШАЦЬКА

Київ – 2022

ТЕКСТ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювальної мережі

(Найменування програми (документа))

CD-R

(Вид носія даних)

26 арк, 336000 Кб

(Обсяг програми, арк., Кб)

Київ – 2022

Файл IPipeline.cs

```
using System.Threading.Tasks;

namespace .Processing.Pipeline
{
   public interface IPipeline
   {
      Task ProcessPacketAsync(IPacket2 packet);
      void Shutdown(int millisecondsTimeout);
   }
}
```

Файл Pipeline.cs

```
using Microsoft.Extensions.Logging;
using Service.Common.MessageOrchestrator;
using Service.Common.Settings;
using Service.Contract.Models;
using Service.Contract.Models.Enums;
using Service.UniqueId.IdGeneration;
using .Processing.Diagnostics;
using .Processing.Parser;
using ;
using Threading.Tasks.Dataflow;
using System;
using System.Runtime.CompilerServices;
using System.Threading;
using System.Threading.Tasks;
using System.Threading.Tasks.Dataflow;

namespace .Processing.Pipeline
{
   internal sealed class Pipeline : IPipeline, IDisposable
   {
      private readonly ILogger<Pipeline> logger;
      private readonly IMessageOrchestratorTpl messageOrchestrator;
      private readonly IMessageOrchestratorBlock orchestratorBlock;

      private readonly IParser parser;
      private readonly ILogEntryIdentifierGenerator identifierGenerator;
      private readonly IDiagnostics performanceCounters;
      private readonly CancellationTokenSource cancellationTokenSource;

      private readonly TransformBlock<IPacket2, ReceivedPacket> identifierGeneratorBlock;
      private readonly TransformBlock<ReceivedPacket, ILogEntryOrchestratorData> parsingBlock;
      private readonly ActionBlock<ILogEntryOrchestratorData> discardBlock;
      private readonly IBufferedBlock completionBlock;

            private const double ParsingMaxDopCoefficient = 1.0 / 6;
      internal int MaxDegreeOfParallelism { get; }
      internal int BoundedCapacity { get; }
      internal Task Completion => completionBlock.Completion;

      public Pipeline(ILogger<Pipeline> logger,
                      ILogServiceSettings settings,
                      IMessageOrchestratorTpl messageOrchestrator,
                      IDiagnostics performanceCounters,
                      IParser parser,
                      ILogEntryIdentifierGenerator identifierGenerator)
      {
         this.logger = logger ?? throw new ArgumentNullException(nameof(logger));
         if (settings == null)
```

```
        {
            throw new ArgumentNullException(nameof(settings));
        }
        cancellationTokenSource = new CancellationTokenSource();
        this.messageOrchestrator = messageOrchestrator ?? throw new
ArgumentNullException(nameof(messageOrchestrator));
        orchestratorBlock = messageOrchestrator.TargetBlock;
        this.performanceCounters = performanceCounters ?? throw new
ArgumentNullException(nameof(performanceCounters));
        this.parser = parser ?? throw new ArgumentNullException(nameof(parser));
        this.identifierGenerator = identifierGenerator ?? throw new ArgumentNullException(nameof(identifierGenerator));
        MaxDegreeOfParallelism =
CountMaxDegreeOfParallelismFromOrchestratorSetting(orchestratorBlock.MaxDegreeOfParallelism);
        BoundedCapacity = settings.BuffersCapacity > 0 ? settings.BuffersCapacity : DataflowBlockOptions.Unbounded;

        identifierGeneratorBlock = new TransformBlock<IPacket2, ReceivedPacket>(
            AssignIdInternal,
            new ExecutionDataflowBlockOptions
            {
                CancellationToken = cancellationTokenSource.Token,
                // let's introduce small buffer here to allow TPL to better balance the load
                BoundedCapacity = 10,
                MaxDegreeOfParallelism = 1,
            }
            );
        parsingBlock = new TransformBlock<ReceivedPacket, ILogEntryOrchestratorData>(
            ParsePacketInternal,
            new ExecutionDataflowBlockOptions
            {
                CancellationToken = cancellationTokenSource.Token,
                BoundedCapacity = BoundedCapacity,
                MaxDegreeOfParallelism = MaxDegreeOfParallelism,
                EnsureOrdered = false
            });
        discardBlock = new ActionBlock<ILogEntryOrchestratorData>(
            DiscardMessageInternal,
            new ExecutionDataflowBlockOptions
            {
                CancellationToken = cancellationTokenSource.Token,
                BoundedCapacity = 1,
                MaxDegreeOfParallelism = 1
            });
        // order of targets is important
        _ = identifierGeneratorBlock.LinkTo(parsingBlock,
            new DataflowLinkOptions { PropagateCompletion = true });
        _ = parsingBlock.LinkTo(discardBlock,
            new DataflowLinkOptions { PropagateCompletion = true },
            (data) => data == null || data.Discarded);
        _ = parsingBlock.LinkTo(orchestratorBlock,
            new DataflowLinkOptions { PropagateCompletion = true });

        completionBlock = identifierGeneratorBlock.ToBufferedPipeline(
            new[]
                {
                    parsingBlock.Completion,
                    orchestratorBlock.Completion,
                    discardBlock.Completion
                },
            () => identifierGeneratorBlock.InputCount +
                  identifierGeneratorBlock.OutputCount +
                  parsingBlock.InputCount +
                  parsingBlock.OutputCount +
                  orchestratorBlock.Count +
                  discardBlock.InputCount);
        logger.LogInformation($"MaxDegreeOfParallelism: {MaxDegreeOfParallelism}");
```

```
            logger.LogInformation($"BoundedCapacity: {BoundedCapacity}");
        }

    internal static int CountMaxDegreeOfParallelismFromOrchestratorSetting(int messageOrchestratorMaxDop)
        {
            return Math.Max((int)Math.Round(messageOrchestratorMaxDop * ParsingMaxDopCoefficient), 1);
        }

        /// <summary>
        /// This is just for quick orientation, because there still can be some blocks processing.
        /// </summary>
        internal int CurrentQueueCount => completionBlock.Count;

        // to mitigate Rule S4457
        public Task ProcessPacketAsync(IPacket2 packet)
        {
            if (packet == null)
            {
                throw new ArgumentNullException(nameof(packet));
            }

            if (MaxDegreeOfParallelism == 1)
            {
                // we don't want to block receiving thread
                return Task.Run(() => BufferMessageInternal(packet), cancellationTokenSource.Token);
            }

            return BufferMessageInternal(packet);
        }

        private async Task BufferMessageInternal(IPacket2 packet)
        {
            try
            {
                bool sendCompleted = await identifierGeneratorBlock.SendAsync(packet).ConfigureAwait(false);

                if (!sendCompleted)
                {
                    logger.LogWarning($"Packet from {packet.ReportingIpAsString} dropped, processing pipeline doesn't accept
input");
                    DiscardMessageInternal(null);
                }
            }
            catch (Exception e)
            {
                logger.LogError($"Processing packet from {packet.ReportingIpAsString} failed with exception", e);
            }
        }

        private ReceivedPacket AssignIdInternal(IPacket2 Packet)
        {
            var packetWithId = new ReceivedPacket(Packet);
            try
            {
                var uid = identifierGenerator.GenerateIdentifiers(LogSourceType.);
                packetWithId.LogEntryId = uid;
            }
            catch (Exception e)
            {
                logger.LogError($"Error generating timestamp {Packet.ReportingIpAsString}", e);
            }
            return packetWithId;
        }

        private ILogEntryOrchestratorData ParsePacketInternal(ReceivedPacket receivedPacket)
```

```
        {
            performanceCounters.SetParserQueueLength(parsingBlock.InputCount);

            // discard the message if ID wasn't assigned
            if (receivedPacket.LogEntryId == null)
                return null;

            try
            {
                var data = parser.Parse(receivedPacket.Packet);
                data.Id = (long)receivedPacket.LogEntryId.Value;
                return data;
            }
            catch (Exception e)
            {
                logger.LogError($"Error parsing message from {receivedPacket.Packet.ReportingIpAsString}", e);
                return null;
            }
        }

        private void DiscardMessageInternal(ILogEntryOrchestratorData logEntryData)
        {
            // increasing performance counter should be safe (no exception) operation
            if (logEntryData == null || logEntryData.Discarded)
            {
                performanceCounters.Discarded();
            }
        }

        internal void Complete()
        {
            completionBlock.Complete();
        }

        public void Shutdown(int millisecondsTimeout)
        {
            if (!completionBlock.Complete(millisecondsTimeout, count => logger.LogInformation($"Pending items in buffer:
{count}")))
            {
                var count = completionBlock.Count;
                logger.LogInformation($"Forcing shutdown of the pipeline");
                Cancel();
                logger.LogWarning($"Pipeline cancelled, items in buffer: {count}");
            }
        }

        internal void Cancel()
        {
            logger.LogInformation($"Cancelling the pipeline");
            cancellationTokenSource.Cancel();
            messageOrchestrator.Cancel();
        }

        public void Dispose()
        {
            cancellationTokenSource?.Dispose();
        }

        internal class ReceivedPacket
        {
            public ReceivedPacket(IPacket2 Packet)
            {
                Packet = Packet;
            }
```

```
            public UniqueId LogEntryId { get; set; }

            public IPacket2 Packet { get; }
        }
    }
}
```

Файл IParcer.cs

```
using Service.Contract.Models;

namespace .Processing.Parser
{
   public interface IParser
   {
      /// <summary>
      /// Accepts a Packet as a parameter and returns an an instance of ILogEntryData
      /// </summary>
      ILogEntryOrchestratorData Parse(IPacket2 packet);

      long WarnedIPs { get; }
   }
}
```

Файл Parcer.cs

```
using Microsoft.Extensions.Logging;
using Service.Common.Diagnostics;
using Service.Common.Encodings;
using Service.Common.Settings;
using Service.Contract.Models;
using Service.Contract.Models.Enums;
using Service.Contract.Utility;
using Service.Model.AlertEvents;
using Service.Model.Utility;
using .Processing.Diagnostics;
using ;
using .Contract;
using .Strings;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Runtime.Caching;
using System.Text;
using System.Text.RegularExpressions;

namespace .Processing.Parser
{
   internal class Parser : IDisposable, IParser
   {
      internal class TimeoutTest : TestPointIdentifier
      {
      }

      internal class ParseErrorTest : TestPointIdentifier
      {
      }

      private const string FacilityFieldKey = "Facility";
      private const string SeverityFieldKey = "Severity";
      private const string PriorityFieldKey = "Priority";
      private const string InputSourceFieldKey = "InputSource";
      private const int NumberOfBytesToSearchInPacketForBom = 200;

      private readonly ILogger<Parser> logger;
```

```
        internal static class RegexGroup
        {
            public const string Priority = "Priority";
            public const string  Ip = " Ip";
            public const string Timestamp = "Timestamp";
            public const string EpochMicroseconds = "EpochMicroseconds";
            public const string EpochTime = "EpochTime";
            public const string HostName = "HostName";
            public const string Message = "Message";
            public const string LegacyDateTime = "LegacyDateTime";
            public const string Month = "Month";
            public const string Day = "Day";
            public const string Hour = "Hour";
            public const string Minute = "Minute";
            public const string Second = "Second";
            public const string TimeZoneOffset = "TimeZoneOffset";
            public const string TimeZone = "TimeZone";
            public const string ProcId = "ProcId";
            public const string AppName = "AppName";
            public const string MessageId = "MessageId";
        }

    private static readonly byte[] Utf8Bom = Encoding.UTF8.GetPreamble();
        private readonly Encoding currentEncoding;
        private readonly IDateTimeProvider dateTimeProvider;
        private readonly IDiagnostics performanceCounters;
        private readonly ILogServiceSettings serviceSettings;

        internal readonly MemoryCache parserWarnedIpCache =
    new(".Processing.ServiceImplementation.ParserWarnedIpCache");
        public long WarnedIPs => parserWarnedIpCache.GetCount();
        private readonly CacheItemPolicy policy = new();

        private readonly TimeSpan inferenceWindowInMinutes;
        private readonly Regex legacyRegEx;
        private readonly Regex rfc5424RegEx;
        private readonly string priorityPattern = @"<(?<Priority>\d{1,3})>";
        private readonly string originalAddressPattern = @"(?:Original Address=(?< Ip>[\w\.\:]*)\s)";

        public Parser(IDiagnostics performanceCounters,
                      IDateTimeProvider dateTimeProvider,
                      ILogServiceSettings serviceSettings,
                      IEncodingProvider encodingProvider,
                      ILogger<Parser> logger)
        {
            this.logger = logger ?? throw new ArgumentNullException(nameof(logger));
            this.performanceCounters = performanceCounters ?? throw new
    ArgumentNullException(nameof(performanceCounters));
            this.dateTimeProvider = dateTimeProvider ?? throw new ArgumentNullException(nameof(dateTimeProvider));
            this.serviceSettings = serviceSettings ?? throw new ArgumentNullException(nameof(serviceSettings));
            policy.AbsoluteExpiration = DateTimeOffset.Now.AddMinutes(60);

            currentEncoding = encodingProvider?.GetEncoding(serviceSettings.EncodingType) ?? throw new
    ArgumentNullException(nameof(encodingProvider));

            var regExTimeout = TimeSpan.FromMilliseconds(this.serviceSettings.RegExMatchTimeoutInMilliseconds);
            // Summaries of the parsing regex can be found at: https://cp.com/pages/viewpage.action?pageId=98518265
            rfc5424RegEx =
                new
                    Regex(
                        @"(?<RawMessage>(" +
                        priorityPattern +
                        @"\s*)" +
                        originalAddressPattern +
                        @"?([1-9]{0,2})\s" +
```

```
                @"(?<Timestamp>-|((?<EpochTime>\d{10})\.(?<EpochMicroseconds>\d+))|((\d{4})-(\d{2})-
(\d{2})T(\d{2}):(\d{2}):(\d{2})(.\d+)?(Z|((\+|-)(\d{2}):(\d{2}))))|([^ ]+))\s" +
                @"(?<HostName>-|\S{1,255})\s" +
                @"(?<Message>(?<AppName>-|\S{1,255})\s(?<ProcId>-|\S{1,255})\s(?<MessageId>-|\S{1,255})\s(-
|(\[(([^=\]""]{1,32})\s*=\s*""([^""\]\\]*)"")*])+)(\s(?:(.|\n)*))?|(?<MessageCatchCall>.+))" +
                @"?)",
                RegexOptions.Compiled,
                regExTimeout);

            legacyRegEx =
                new
                    Regex(
                        @"^\s*(?:" +
                        priorityPattern +
                        ")?" +
                        originalAddressPattern +
                        @"?:?\s*(?:\d*\s*\*)?(?<LegacyDateTime>(?:\s*(?<Year2>\d{4})\s)?(?:(?<Month>" +

@"(?:Jan)|(?:Feb)|(?:Mar)|(?:Apr)|(?:May)|(?:Jun)|(?:Jul)|(?:Aug)|(?:Sep)|(?:Oct)|(?:Nov)|(?:Dec)|(?:JAN)|(?:FEB)|(?:MAR
)|(?:APR)|(?:MAY)|(?:JUN)|(?:JUL)|(?:AUG)" +

@"|(?:SEP)|(?:OCT)|(?:NOV)|(?:DEC))\s*(?<Day>\d{1,2})(?:\s(?<Year1>\d{4}))?\s+(?:(?<Hour>\d{1,2}):(?<Minute>\d
{2}):(?<Second>\d{2})(\.(?<PartialSecond>\d+))?)" +
                @"(\s(?<TimeZone>([A-Z]{3})))?(?<TimeZoneOffset>\s(?:-|\+)\d{2}:\d{2})?:?[^\w-
:])?)?\s*(?:(?<HostName>[\w-.]+):?\s+)?(?<Message>(.|\n)*)",
                RegexOptions.Compiled,
                regExTimeout);

            inferenceWindowInMinutes = TimeSpan.FromMinutes(this.serviceSettings.InferTimezoneWithinMinuteRange);
        }

        public void Dispose()
        {
            parserWarnedIpCache?.Dispose();
        }

        /// <summary>
        /// Accepts a Packet as a parameter and returns an instance of ILogEntryData
        /// </summary>
        /// <exception cref="ArgumentNullException">When <paramref name="packet" /> is null.</exception>
        /// <returns>Parsed message when it succeeds or message flagged as discarded in case error during parsing
occurs.</returns>
        public ILogEntryOrchestratorData Parse(IPacket2 packet)
        {
            if (packet == null)
            {
                throw new ArgumentNullException(nameof(packet));
            }

            try
            {
                //TODO: MALTBY - Needed?
                //LocaleThreadState.EnsurePrimaryLocale();

                using (new PerformanceTracker(performanceCounters.RecordParsePacket))
                {
                    var messageText = GetRawMessageFromPacket(packet);
                    if (logger.IsEnabled(LogLevel.Debug))
                    {
                        logger.LogDebug($" message:{messageText}");
                    }

                    var isLegacy = IsLegacyMessage(messageText);
                    Match match = null;
                    try
```

```
            {
                match = isLegacy
                        ? legacyRegEx.Match(messageText)
                        : rfc5424RegEx.Match(messageText);
                if (TestPointIdentifier.ContainsTestPoint<TimeoutTest>(messageText, logger))
                {
                    match = null;
                    throw new RegexMatchTimeoutException();
                }
                //We weren't able to parse per one of the RFC standards so log a warning (one per hour).
                if (!match.Success || TestPointIdentifier.ContainsTestPoint<ParseErrorTest>(messageText, logger))
                {
                    LogRegexMatchFailure(packet, messageText, false);
                    if (TestPointIdentifier.ContainsTestPoint<ParseErrorTest>(messageText, logger))
                    {
                        match = Regex.Match(messageText, "");
                    }
                }
            }
            catch (RegexMatchTimeoutException)
            {
                LogRegexMatchFailure(packet, messageText, true);
            }

            var logEntry = CreateLogEntryFromMatch(packet, isLegacy, match, messageText);

            if (logEntry != null &&
                LogEntryOrchestratorData.LogSourceTypesNeedingForwardData.Contains(logEntry.SourceType))
            {
                logEntry.RawMessage = packet.RawPacket.ToArray();
            }

            return logEntry;
        }
    }
    catch (Exception e)
    {
        logger.LogError($" parsing failed. {e}");
        OrchestratorData logEntry = new();
        logEntry.Discard();
        return logEntry;
    }
}

private string GetRawMessageFromPacket(IPacket2 packet)
{
    string messageText;
    var rawPacketStrippedOfBom = StripBomCharacters(packet.RawPacket);

    //Use our current encoding, which may be driven by a user setting, unless it is overridden by a BOM at the
beginning of the message
    using (var stream = new MemoryStream(rawPacketStrippedOfBom))
    using (var reader = new StreamReader(stream, currentEncoding, true))
    {
        var maxCharactersToRead = (int)Math.Min(stream.Length, serviceSettings.MaxMessageLength);
        var messageCharacters = new char[maxCharactersToRead];
        _ = reader.Read(messageCharacters, 0, maxCharactersToRead);
        messageText = new string(messageCharacters).TrimEnd('\0');
    }

    return messageText;
}

private OrchestratorData CreateLogEntryFromMatch(IPacket2 packet, bool isLegacy, Match match, string rawText)
    {
```

```
            var logEntry = new OrchestratorData()
            {
                SocketPeerAddress = packet.ReportingIpAsString,
                RawMessageText = rawText
            };
            try
            {
                var priority = match?.Groups[RegexGroup.Priority].Value;
                var version = match?.Groups[2].Value;

                GetFacilityAndSeverityFromPriority(priority, out var facilityValue, out var severityValue);

                string unparseableDateString = null;
                var messageDateTimeOffset = ExtractMessageDateTime(isLegacy, match, ref unparseableDateString);

                //Use the  Ip over any hostname found in the message
                var  Ip = match?.Groups[RegexGroup. Ip].Value;
                logEntry.IpAddress = !string.IsNullOrEmpty( Ip) ?  Ip : packet.ReportingIpAsString;

                var messageHostname = match?.Groups[RegexGroup.HostName].Value;

                var hostname = (!string.IsNullOrEmpty( Ip) ?  Ip : null) ??
                               (messageHostname != "" ? messageHostname : null);

                var procId = match?.Groups[RegexGroup.ProcId].Value;
                var messageId = match?.Groups[RegexGroup.MessageId].Value;
                var appName = match?.Groups[RegexGroup.AppName].Value;

                // Property values for NetworkHostId and SourceType will be set in the MessageSourceManager down the
pipleline
                // along with 'MachineType' and 'Vendor' in the 'Fields' dictionary property
                logEntry.MessageDateTime = messageDateTimeOffset?.UtcDateTime;

                string severityName = null;
                if (severityValue != null)
                {
                    severityName = Enum.GetName(typeof(Severity), severityValue.Value);
                }
                logEntry.Level = severityName ?? "Unknown";

                logEntry.Message = match?.Groups[RegexGroup.Message].Value.TrimStart(' ', ':') ?? rawText;
                logEntry.NetworkHostCaption = hostname;

                if (severityValue != null && facilityValue != null)
                {
                    var fields = CreateFacilityAndSeverityFields(facilityValue.Value, severityValue.Value);
                    fields.Add(PriorityFieldKey, new LogEntryFieldData(DataFormat.Numeric, new
FieldValue(CalculatePriority(facilityValue.Value, severityValue.Value)), logger));
                    logEntry.AddFieldValues(fields);
                }

                logEntry.LocalAddress = serviceSettings.LocalIpAddress;
                logEntry.LocalPort = packet.Protocol switch
                {
                    Protocol.Tcp => serviceSettings.TcpListenPort,
                    Protocol.TcpOverTls => serviceSettings.TcpSecureListenPort,
                    Protocol.Udp => serviceSettings.UdpListenPort,
                    _ => 0,
                };

                logEntry.RFCHostname = messageHostname;
                logEntry.RFCDate = isLegacy ? match.Groups[RegexGroup.LegacyDateTime].Value.Trim(' ') :
match.Groups[RegexGroup.Timestamp].Value;
                logEntry.RFCPIDTag = match?.Groups[RegexGroup.ProcId].Value;
```

```
            var rfcHeader = priority != string.Empty ? $"<{priority}>" : string.Empty;
            rfcHeader += $"{version} {logEntry.RFCDate} {messageHostname} {appName} {procId} {messageId}";
            rfcHeader = Regex.Replace(rfcHeader, @"\s+", " ");
            logEntry.RFCHeader = rfcHeader;

            logEntry.AddFieldValues(new Dictionary<string, ILogEntryFieldData>() { { InputSourceFieldKey, new
LogEntryFieldData(DataFormat.Text, new FieldValue(packet.Protocol.ToString().ToUpper()), logger) } });

            // Commented as we don't have NetworkHostMapper as well as opportunity to check if host is known
            //If we do not know the host name, it might actually be part of the message so add it back.
            //if (!string.IsNullOrEmpty(logEntry.NetworkHostCaption) && string.IsNullOrEmpty( Ip))
            //{
            //    logger.LogTrace($"'{logEntry.NetworkHostCaption}' did not match any known host, adding back to
message.");
            //    logEntry.Message = logEntry.NetworkHostCaption + " " + logEntry.Message;
            //    logEntry.NetworkHostCaption = null;
            //}

            //If we are using the   IP as our hostname/IP value than we should add the message hostname back to the
message
            if (!string.IsNullOrEmpty( Ip))
            {
                logEntry.Message = messageHostname + " " + logEntry.Message;
            }

            //unparseableDateString will be null if we were able to parse the date otherwise we need to add back the raw
string representation
            logEntry.Message = unparseableDateString + logEntry.Message;

            if (logger.IsEnabled(LogLevel.Information))
            {
                var facility = logEntry.Fields.ContainsKey(FacilityFieldKey) ?
logEntry.Fields[FacilityFieldKey].Value.NumericValue.ToString() : "NULL";
                var severity = logEntry.Fields.ContainsKey(SeverityFieldKey) ?
logEntry.Fields[SeverityFieldKey].Value.NumericValue.ToString() : "NULL";
                logger.LogInformation(
                    $" message parsed,
Src:{logEntry.IpAddress},Hostname:{logEntry.NetworkHostCaption},DateTime:{logEntry.MessageDateTime},Discard:{
logEntry.Discarded},IsLegacy:{isLegacy},Severity:{severity},Facility:{facility},Level:{logEntry.Level},Message:
{logEntry.Message}");
            }
        }
        catch (Exception ex)
        {
            logger.LogError(ex.ToString());
        }

        return logEntry;
    }

    private DateTimeOffset? ExtractMessageDateTime(bool isLegacy, Match match, ref string unparseableDateString)
    {
        if (match == null)
        {
            return null;
        }

        DateTimeOffset? messageDateTimeOffset;
        try
        {
            messageDateTimeOffset = ParseTimeFromMessage(isLegacy, match, dateTimeProvider.GetUtcDateTime());
            if (messageDateTimeOffset == null)
            {
                unparseableDateString = HandleUnparseableDateTime(isLegacy, match, out messageDateTimeOffset);
            }
```

```
            }
            catch
            {
                unparseableDateString = HandleUnparseableDateTime(isLegacy, match, out messageDateTimeOffset);
            }

            return messageDateTimeOffset;
        }

        private static byte[] StripBomCharacters(byte[] byteArray)
        {
            var numberOfBytesToCheckForBom = Math.Min(NumberOfBytesToSearchInPacketForBom, byteArray.Length) -
Utf8Bom.Length;

            for (var i = 0; i <= numberOfBytesToCheckForBom; i++)
            {
                int bomIndex = 0;
                while (bomIndex < Utf8Bom.Length && byteArray[i + bomIndex] == Utf8Bom[bomIndex])
                {
                    bomIndex++;
                }

                if (bomIndex == Utf8Bom.Length)
                {
                    var strippedBytes = new byte[byteArray.Length - Utf8Bom.Length];

                    Array.Copy(byteArray, 0, strippedBytes, 0, i);                                  //Copy everything up to the
BOM
                    Array.Copy(byteArray, i + Utf8Bom.Length, strippedBytes, i, byteArray.Length - i - Utf8Bom.Length);
//Copy everything after the BOM

                    return strippedBytes;
                }
            }

            return byteArray;
        }

private static string HandleUnparseableDateTime(bool isLegacy, Match match, out DateTimeOffset?
messageDateTimeOffset)
        {
            var unparseableDateString = isLegacy ? match.Groups[RegexGroup.LegacyDateTime].Value.Trim() :
match.Groups[RegexGroup.Timestamp].Value.Trim();
            if (!string.IsNullOrEmpty(unparseableDateString))
            {
                unparseableDateString += " ";
            }

            messageDateTimeOffset = null;
            return unparseableDateString;
        }

        private DateTimeOffset? ParseTimeFromMessage(bool isLegacy, Match match, DateTime utcTime)
        {
            if (!isLegacy)
            {
                var timeStamp = match.Groups[RegexGroup.Timestamp].Value;
                var epochTime = match.Groups[RegexGroup.EpochTime].Value;
                if (!string.IsNullOrEmpty(epochTime))
                {
                    return EpochToDateTime(epochTime, match.Groups[RegexGroup.EpochMicroseconds].Value);
                }

                return DateTime.Parse(
                    timeStamp,
```

```
                CultureInfo.InvariantCulture,
                DateTimeStyles.None);
        }

        // If we have a "legacy" timestamp we should at least have month, day and time to the second
        // We may not have a timezone offset or even the year. In that case check to see if the message
        // could conceivable arrived within a few minutes -- if so, assume local time
        if (string.IsNullOrEmpty(match.Groups[RegexGroup.LegacyDateTime].Value))
        {
            return null;
        }

        var year = ParseYear(match);
        var month = DateTime.ParseExact(match.Groups[RegexGroup.Month].Value, "MMM",
CultureInfo.InvariantCulture).Month;
        var day = int.Parse(match.Groups[RegexGroup.Day].Value);
        var hour = int.Parse(match.Groups[RegexGroup.Hour].Value);
        var minute = int.Parse(match.Groups[RegexGroup.Minute].Value);
        var second = int.Parse(match.Groups[RegexGroup.Second].Value);
        TimeSpan offset;

        var parsedDateTime = new DateTime(year ?? 1, month, day, hour, minute, second); //year may not be valid at this
point

        // if we weren't able to parse out the year but the message appears to be from within the last five minutes we can
infer
        // the message year is the current year unless it arrives in the window around New Year's day.
        var currentYear = utcTime.Year;

        if (year == null)
        {
            year = InferMessageYear(utcTime, currentYear, parsedDateTime);
            parsedDateTime = new DateTime((int)year, month, day, hour, minute, second);
        }

        // We only know we have a valid timestamp if we have an offset
        // Representations such as 'PDT' are not globablly unique
        if (!string.IsNullOrEmpty(match.Groups[RegexGroup.TimeZoneOffset].Value))
        {
            _ = TimeSpan.TryParse(match.Groups[RegexGroup.TimeZoneOffset].Value, out offset);
        }
        // 'UTC' is the one timezone value (along with all offsets) we respect
        else if (match.Groups[RegexGroup.TimeZone].Value.ToUpperInvariant() == "UTC")
        {
            offset = new TimeSpan();
        }
        // If we don't have an explicit timezone offset assume local or UTC when the recieve time appears to be within five
minutes of one or the other.
        else
        {
            if (!AttemptToInferTimezoneOffset(utcTime, parsedDateTime, out offset))
            {
                return null;
            }
        }

        var offsetValue = new DateTimeOffset((int)year, month, day, hour, minute, second, offset);
        return offsetValue;

        static DateTime EpochToDateTime(string timestamp, string microseconds)
        {
            return DateTimeOffset.FromUnixTimeSeconds(long.Parse(timestamp))
                                 .UtcDateTime
                                 .AddMilliseconds(long.Parse(microseconds) / 1000000F);
        }
```

```
    }

    private Dictionary<string, ILogEntryFieldData> CreateFacilityAndSeverityFields(int facility, int severity)
    {
        return new Dictionary<string, ILogEntryFieldData>
            {
                { FacilityFieldKey, new LogEntryFieldData(DataFormat.Numeric, new FieldValue(facility), logger) },
                {
                    "FacilityName",
                    new LogEntryFieldData(
                        DataFormat.Text,
                        new FieldValue(Resources.GetString(
                            $"_Facility_{facility}"/*,i18nExternalized.Culture*/)
                            ), logger)
                },
                { SeverityFieldKey, new LogEntryFieldData(DataFormat.Numeric, new FieldValue(severity), logger) }
            };
    }

    private bool AttemptToInferTimezoneOffset(DateTime currentUtcTime, DateTime parsedDateTime, out TimeSpan
offset)
    {
        if (inferenceWindowInMinutes.Minutes == 0)
        {
            return HandleCannotInferTimezone(out offset);
        }

        var messageToUtcOffset = parsedDateTime - currentUtcTime;

        //Round to the nearest hour
        var nearestOffsetHour = messageToUtcOffset.Minutes >= 30 ? messageToUtcOffset.Hours + 1 :
messageToUtcOffset.Hours;
        nearestOffsetHour = messageToUtcOffset.Minutes <= -30 ? messageToUtcOffset.Hours - 1 : nearestOffsetHour;

        //Valid UTC offsets fall between -12 and +14
        if (nearestOffsetHour < -12 || nearestOffsetHour > 14)
        {
            return HandleCannotInferTimezone(out offset);
        }

        var messageTimeToTest = new DateTimeOffset(parsedDateTime, new TimeSpan());

        messageTimeToTest += TimeSpan.FromHours(nearestOffsetHour * -1);

        var currentUtcTimeWithOffset = new DateTimeOffset(currentUtcTime, new TimeSpan());

        // If we are outside our tolerance window between when the message was sent and when it was received then we
cannot infer the timezone
        if (messageTimeToTest <= currentUtcTimeWithOffset.Subtract(inferenceWindowInMinutes) ||
            messageTimeToTest >= currentUtcTimeWithOffset.Add(inferenceWindowInMinutes))
        {
            return HandleCannotInferTimezone(out offset);
        }

        offset = TimeSpan.FromHours(nearestOffsetHour);
        return true;
    }

    private static bool HandleCannotInferTimezone(out TimeSpan offset)
    {
        offset = new TimeSpan();
        return false;
    }
```

Файл CommandProcessorDIExtension.cs

```
using Microsoft.Extensions.DependencyInjection;

namespace ConsoleProcessor
{
   public static class CommandProcessorDIExtension
   {
      public static void RegisterCommandProcessor(this IServiceCollection collection)
      {
         collection.AddScoped<ICommandProcessor, ConsoleCommandProcessor>();
      }
   }
}
```

Файл ConsoleCommandProcessor.cs

```
using System;

namespace ConsoleProcessor
{
   public class ConsoleCommandProcessor: ICommandProcessor
   {
      public string SourceFile { get; set; }
      private string SourceFileName { get; set; }
      public string SourceFormat { get; set; }
      public string GoalFormat { get; set; }
      public string OutputFile { get; set; }


      public ConsoleCommandProcessor()
      {
         SourceFile = "";
         GoalFormat = "";
         OutputFile = "";
      }

      public bool ProcessCommand(string[] args)
      {
         for (var i = 0; i < args.Length; i++)
         {
            args[i] = args[i].ToLower();

            if (args[i].StartsWith("--source="))
            {
               SourceFile = args[i][(args[i].IndexOf('=') + 1)..];
               SourceFormat = SourceFile[(SourceFile.LastIndexOf('.')+1)..];
               SourceFileName = SourceFile[..^SourceFormat.Length];
            }
            else if (args[i].StartsWith("--goal-format="))
            {
               GoalFormat = args[i][(args[i].IndexOf('=') + 1)..].ToLower();
            }
            else if (args[i].StartsWith("--output="))
            {
               OutputFile = args[i][(args[i].IndexOf('=') + 1)..];
               OutputFile += "." + GoalFormat;
            }
            else
            {
               Console.WriteLine($"Argument {args[i]} is invalid.");
               Environment.Exit(1);
            }
```

```
            }

            if (SourceFile == "")
            {
                Console.WriteLine("Argument --source is either entered incorrectly or is missing");
                Environment.Exit(1);
            }

            if (GoalFormat == "" && !SourceFormat.Contains("obj") && !SourceFormat.Contains("cowscene"))
            {
                Console.WriteLine("Argument --goal-format is either entered incorrectly or is missing");
                Environment.Exit(1);
            }

            if (OutputFile == "")
            {
                OutputFile = SourceFileName + GoalFormat;
            }

            return true;
        }
    }
}
```

Файл RuleProcessor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using ConverterBase;
using ConverterBase.GeomHelper;
using Priority_Queue;
using Raytracer.Optimisation;
using Raytracer.Scene.Interfaces;
using static System.Numerics.Vector3;

namespace Raytracer.Tracing
{
    public class Raytracer: ITracer
    {
        public List<List<Pixel>> Trace(ISceneCreator sceneCreator, List<INode> octrees)
        {
            var camera = sceneCreator.ParamsProvider.Camera;
            var screenZ = sceneCreator.ParamsProvider.ScreenZ;

            var watch = System.Diagnostics.Stopwatch.StartNew();
            var num = 0;

            var image = new List<List<Pixel>>();
            for (var i = 0; i < sceneCreator.ParamsProvider.ImageHeight; i++)
            {
                image.Add(new List<Pixel>());
                var y = sceneCreator.SetXScreenCoordinate(i);
                for (var j = 0; j < sceneCreator.ParamsProvider.ImageWidth; j++)
                {
                    var x = sceneCreator.SetYScreenCoordinate(j);

                    var pixelCenterPoint = new Vector3(x, y, screenZ);
                    var rayDirection = Normalize(pixelCenterPoint - camera);

                    // float t = 0;
                    // var priorityQueue = new SimplePriorityQueue<INode, float>();
```

```
                var triangles = GetIntersectedTriangles(sceneCreator, octrees, rayDirection, ref num);
                var normal = new Vector3();
                var sphereIntersectionPoint = new Vector3();

                if (IsSphereIntersect(camera, rayDirection, ref normal, ref sphereIntersectionPoint))
                {
                    var lightRay = Normalize(sceneCreator.ParamsProvider.LightPosition - sphereIntersectionPoint);
                    var dotProduct = Dot(lightRay, normal);
                    var facingRatio = Math.Max(0, dotProduct);
                    image[i].Add(new Pixel((byte)(239*facingRatio), (byte)(154*facingRatio), (byte)(154*facingRatio)));
                }
                else if (!triangles.Any())
                {
                    image[i].Add(new Pixel(232, 234, 246));
                }
                else
                {
                    //var facingRatio = 0.18f / Math.PI * 30 * 0.5 * Math.Max(0f, dotProduct);
                    var (nearestTriangle, intersectionPoint, barycentricIntersectionPoint) = GetNearestTriangle(triangles,
sceneCreator.ParamsProvider.Camera);

                    // trying to draw shadows
                    var shadowRay = Normalize(intersectionPoint - sceneCreator.ParamsProvider.LightPosition);
                    var barrier = GetIntersectedTriangles(sceneCreator, octrees, shadowRay, ref num);

                    if (!barrier.Any())
                    {
                        var temp = new Vector3(0, 0, 0);
                        if (IsSphereIntersect(intersectionPoint, shadowRay, ref temp, ref temp))
                        {
                            image[i].Add(new Pixel(0, 0, 0));
                        }
                        else
                        {
                            normal = nearestTriangle.GetNormal();
                            // normal = nearestTriangle.GetBarycentricNormal(barycentricIntersectionPoint);
                            var lightRay = Normalize(sceneCreator.ParamsProvider.LightPosition - intersectionPoint);
                            var dotProduct = Dot(lightRay, normal);
                            var facingRatio = Math.Max(0, dotProduct);
                            var albedo = 0.18;

                            image[i].Add(new Pixel((byte) (159 * facingRatio), (byte) (168 * facingRatio),
                                (byte) (218 * facingRatio)));
                        }
                    }
                    else
                    {

                        image[i].Add(new Pixel(0, 0, 0));
                    }

                }
            }
        }

        watch.Stop();
        var time = watch.ElapsedMilliseconds / 1000;
        Console.WriteLine($"Exec time: {time} s");
        Console.WriteLine($"Num of intersection tests: {num}");

        return image;
    }

private List<(Triangle, Vector3, Vector3)> GetIntersectedTriangles(ISceneCreator sceneCreator, List<INode> octrees,
Vector3 rayDirection, ref int num)
```

```
        {
            var priorityQueue = new SimplePriorityQueue<INode, float>();
            var triangles = new List<(Triangle, Vector3, Vector3)>();
            float t = 0;

            foreach (var octree in octrees)
            {
                if (IsRayIntersectBox(octree.MinBoundary, octree.MaxBoundary, sceneCreator.ParamsProvider.Camera,
rayDirection, ref t))
                {
                    priorityQueue.Enqueue(octree,t);
                }

                Triangle nearestTriangle = null;
                var nearestTriangleIntersectionPoint = new Vector3();
                var minDistance = float.MaxValue;
                var barycentricIntersectionPoint = new Vector3();
                var intersectionPoint = new Vector3(float.MaxValue, float.MaxValue, float.MaxValue);
                var barycentricIntersectionPointOfNearestTriangle = new Vector3();
                var flag = false;
                while (priorityQueue.Count != 0)
                {
                    var node = priorityQueue.Dequeue();
                    var intersectedTriangle =
                        FindIntersectionInBox(node.Faces, rayDirection, sceneCreator.ParamsProvider.Camera, ref num, ref
intersectionPoint, ref barycentricIntersectionPoint);
                    var distanceBetweenCameraAndTriangle = Distance(intersectionPoint,
sceneCreator.ParamsProvider.Camera);

                    if (distanceBetweenCameraAndTriangle < minDistance)
                    {
                        minDistance = distanceBetweenCameraAndTriangle;
                        nearestTriangle = intersectedTriangle;
                        nearestTriangleIntersectionPoint = intersectionPoint;
                        barycentricIntersectionPointOfNearestTriangle = barycentricIntersectionPoint;
                        flag = true;
                    }

                    if (node.IsLeaf())
                    {
                        if (flag)
                        {
                            triangles.Add((nearestTriangle, nearestTriangleIntersectionPoint,
barycentricIntersectionPointOfNearestTriangle));
                            break;
                        }
                    }
                    else
                    {
                        foreach (var child in node.ChildNodes
                            .Where(child => IsRayIntersectBox(child.MinBoundary, child.MaxBoundary,
sceneCreator.ParamsProvider.Camera, rayDirection, ref t)))
                        {
                            priorityQueue.Enqueue(child,t);
                        }
                    }
                }

            }

            return triangles;
        }

        private (Triangle, Vector3, Vector3) GetNearestTriangle(List<(Triangle, Vector3, Vector3)> triangles, Vector3
camera)
```

```
        {
            var minDistance = float.MaxValue;
            var nearestTriangle = new Triangle();
            var nearestTriangleIntersectionPoint = new Vector3();
            var barycentricIntersectionPointOfNearestTriangle = new Vector3();
            foreach (var triangle in triangles)
            {
                var distanceBetweenCameraAndTriangle = Distance(triangle.Item2, camera);
                if (distanceBetweenCameraAndTriangle > 0 && distanceBetweenCameraAndTriangle < minDistance)
                {
                    minDistance = distanceBetweenCameraAndTriangle;
                    nearestTriangle = triangle.Item1;
                    nearestTriangleIntersectionPoint = triangle.Item2;
                    barycentricIntersectionPointOfNearestTriangle = triangle.Item3;
                }
            }

            return (nearestTriangle, nearestTriangleIntersectionPoint, barycentricIntersectionPointOfNearestTriangle);
        }

        private Triangle FindIntersectionInBox(List<Triangle> facesInBox, Vector3 rayDirection, Vector3 camera, ref int
num, ref Vector3 intersectionPoint, ref Vector3 outBarycentricIntersectionPoint)
        {
            Triangle nearestTriangle = null;
            var minDistance = float.MaxValue;
            foreach (var triangle in facesInBox)
            {
                num++;
                if (triangle.IsIntersectTriangle(camera, rayDirection, ref intersectionPoint, ref outBarycentricIntersectionPoint))
                {
                    var distanceBetweenCameraAndTriangle = Distance(intersectionPoint, camera);

                    if (distanceBetweenCameraAndTriangle < minDistance)
                    {
                        nearestTriangle = triangle;
                        minDistance = distanceBetweenCameraAndTriangle;
                    }
                }
            }

            return nearestTriangle;
        }

        private bool IsRayIntersectBox(Vector3 pMin, Vector3 pMax, Vector3 origin, Vector3 direction, ref float t)
        {
            var t1 = (pMin.X - origin.X) * (1f / direction.X);
            var t2 = (pMax.X - origin.X) * (1f / direction.X);
            var t3 = (pMin.Y - origin.Y) * (1f / direction.Y);
            var t4 = (pMax.Y - origin.Y) * (1f / direction.Y);
            var t5 = (pMin.Z - origin.Z) * (1f / direction.Z);
            var t6 = (pMax.Z - origin.Z) * (1f / direction.Z);

            var tMin = Math.Max(Math.Max(Math.Min(t1, t2), Math.Min(t3, t4)), Math.Min(t5, t6));
            var tMax = Math.Min(Math.Min(Math.Max(t1, t2), Math.Max(t3, t4)), Math.Max(t5, t6));

            // if tmax < 0, ray (line) is intersecting AABB, but the whole AABB is behind us
            if (tMax < 0)
            {
                t = tMax;
                return false;
            }

            // if tmin > tmax, ray doesn't intersect AABB
            if (tMin > tMax)
            {
```

```
                t = tMax;
                return false;
            }

            // if tmin < 0 then the ray origin is inside of the AABB and tmin is behind the start of the ray so tmax is the first
intersection
            t = tMin < 0 ? tMax : tMin;
            return true;
        }

        // find intersection with sphere
        private bool IsSphereIntersect(Vector3 origin, Vector3 direction, ref Vector3 normal, ref Vector3 intersectionPoint)
        {
            // sphere equation
            var center = new Vector3(-.1f, -.0f, 1.5f);
            var r = .16f;
            var k = origin - center;

            var a = Dot(direction, direction);
            var b = 2 * Dot(direction, k);
            var c = Dot(k, k) - r * r;

            var D = b * b - 4 * a * c;
            if (D >=0)
            {
                var t1 = (-b + D) / (2 * a);
                var t2 = (-b - D) / (2 * a);

                var v1 = origin + direction * t1;
                var v2 = origin + direction * t2;

                intersectionPoint = Distance(origin, v1) < Distance(origin, v2) ? v1 : v2;

                normal = Normalize(intersectionPoint - center);
                return true;
            }

            return false;
        }

        // find intersection with plane
        private bool IsIntersectPlane(Vector3 normal, Vector3 point, Vector3 origin, Vector3 direction, ref float t)
        {
            var denom = Dot(normal, direction);

            if (denom > 1e-6)
            {
                Vector3 pl = point - origin;
                t = Dot(pl, normal) / denom;

                return t >= 0;
            }
            return false;
        }

    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КЛАСИФІКАЦІЇ ЛОГІВ ЛОКАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ МЕРЕЖІ

Програма та методика тестування

КПІ.ІП-8104.045490.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юрій ОЛІЙНИК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Олександра БЕРШАЦЬКА

Київ – 2022

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення моніторингу та обробки системних логів у локальній обчислювальній мережі.

Програмне забезпечення додатку складається с клієнтської та серверної частин. Обидві повинні бути протестовані окремо та в інтеграції одна з одною. Необхідно провести мануальне тестування та налаштувати процеси автоматичного виявлення дефектів аби спростити процес виявлення дефектів під час розробки та додавання нового функціоналу.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення у відповідно до функціональних вимог;

– знаходження проблем та недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу;

– визначення вимоги до тестового середовища;

– забезпечення високого рівню безпеки та захисту даних користувача;

– перевірка відповідності дизайну вимогам, описаним у технічному завданні;

– перевірка вправності інтерфейсу користувача;

– перевірка коректності процесу створення фільтрів та дій;

– перевірка коректності результатів, отриманих під час фільтрації повідомлень;

– перевірка коректності результатів, отриманих під час виконання дії, зазначених у кожному зі створених правил.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом «сірої скриньки». Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (базове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування користувацького інтерфейсу.

Маємо протестувати кожний можливий варіант використання. Результат кожного тесту та інформація про його перебіг буде збережена до сервісу. Людина, що тестує, проводить кожний із тестів і позначає результат його виконання за кожною із умов. Після того, як тестування було завершено, буде проведено перегляд звіту із тестування. У разі знаходження будь-якої із помилок, їх перелік надається розробнику сервісу для їх подальшого виправлення.

Основна функціональність системи та всі її складові мають функціонувати як очікувалося та бути окресленими в окремих випадках тестування. Не повинно бути виявлено дефектів, при яких основна функціональність системи не виконує, або тільки частково виконує свою задачу. Кінцевий користувач має мати змогу успішно налаштувати параметри прослуховування повідомлень, створити правили обробки повідомлень, які були перехоплені, створювати та редагувати фільтри та дії в межах правил. Необхідно мати змогу пересвідчитися в коректності роботи заданих фільтрів та дій.

Будуть використані наступні методи тестування: функціональне, тестування системної продуктивності, тестування користувацького інтерфейсу та системи в цілому.

Тестування буде виконуватися завдяки інструментам Unit test та manual testing. Не менше ніж 95% від усіх тестових випадків мають бути успішно пройдені. Жоден із невдало пройдених випадків тестування не повинен мати критичного значення для можливостей кінцевого користувача використовувати систему.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність веб-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням нормальних, граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування клієнтської частини додатку у різних браузерах;

– тестування при великому навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Під час тестування має бути перевірена уся функціональна складова програмного забезпечення. Функції програмного забезпечення, що підлягають тестуванню:

– прослуховування повідомлень визначених протоколів за вказаними портами;

– створення правил обробки перехоплених повідомлень;

– створення та менеджмент фільтрів у межах правила;

– створення та менеджмент дій у межах правила;

– коректність роботи створених фільтрів та дій;

– парсинг повідомлення відповідно до стандарту RFC5424.

Тестування виконується вручну, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності в користуванні.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КЛАСИФІКАЦІЇ ЛОГІВ ЛОКАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ МЕРЕЖІ

Керівництво користувача

КПІ.ІП-8104.045490.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юрій ОЛІЙНИК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Олександра БЕРШАЦЬКА

Київ – 2022

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

«Syslog Server» - це Web-застосунок для моніторингу та обробки системних логів локальної обчислювальної мережі орієнтований перш за все на системних адміністраторів. Застосунок значно спрощує задачу підтримки мережі у робочому стані та прискорює процес вирішення проблем, які виникають, бо надає інструменти запобігання виникненню проблем, швидкої локалізації проблеми та знаходження першопричини.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- операційна система Windows версії не менше ніж 8;
- для встановлення додатку на пристрої повинно бути не менше 2 GB вільної пам’яті;
- .NET 6.0;
- центральний процесор не менш ніж 2.4 GHz Dual Core processor.

2.2 Завантаження застосунку

Застосунок можна завантажити за допомогою запуску інсталятора – файлу з розширенням .exe. Під час завантаження необхідно обрати папку, де будуть розміщені файли додатку та створити пароль адміністратору системи. Під час завантаження будуть встановлені необхідні компоненти, якщо вони відсутні на комп’ютері користувача.

Після подвійного натискання на файл інсталяції з’явиться вікно початку установки (рисунок 2.1)

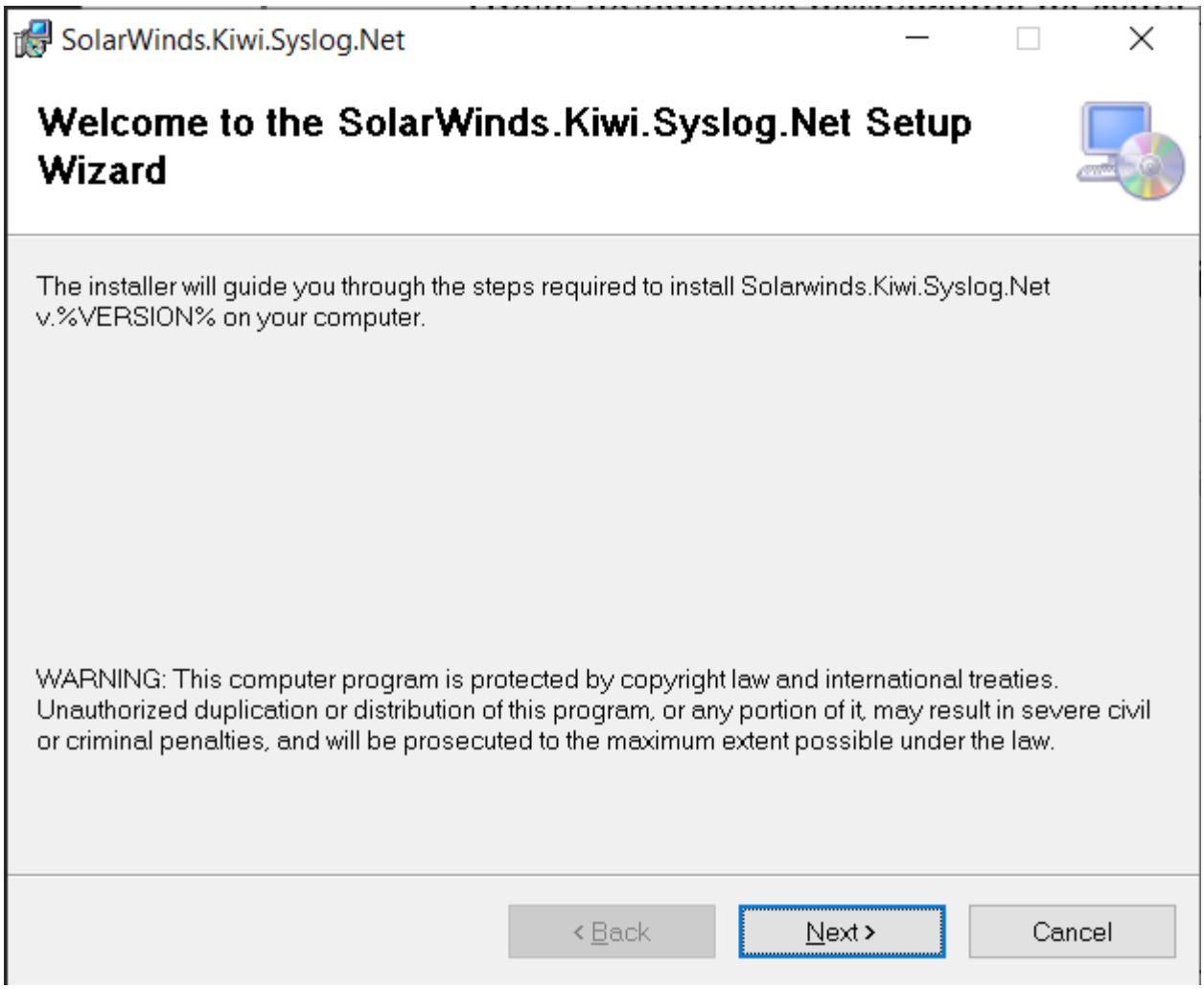


Рисунок 2.1 – Менеджер установки Syslog Server

Наступним кроком необхідно обрати шлях установки застосунку. На рисунку 2.2 зображено форму вибору шляху установки додатку та користувачів, для яких буде встановлено додаток.

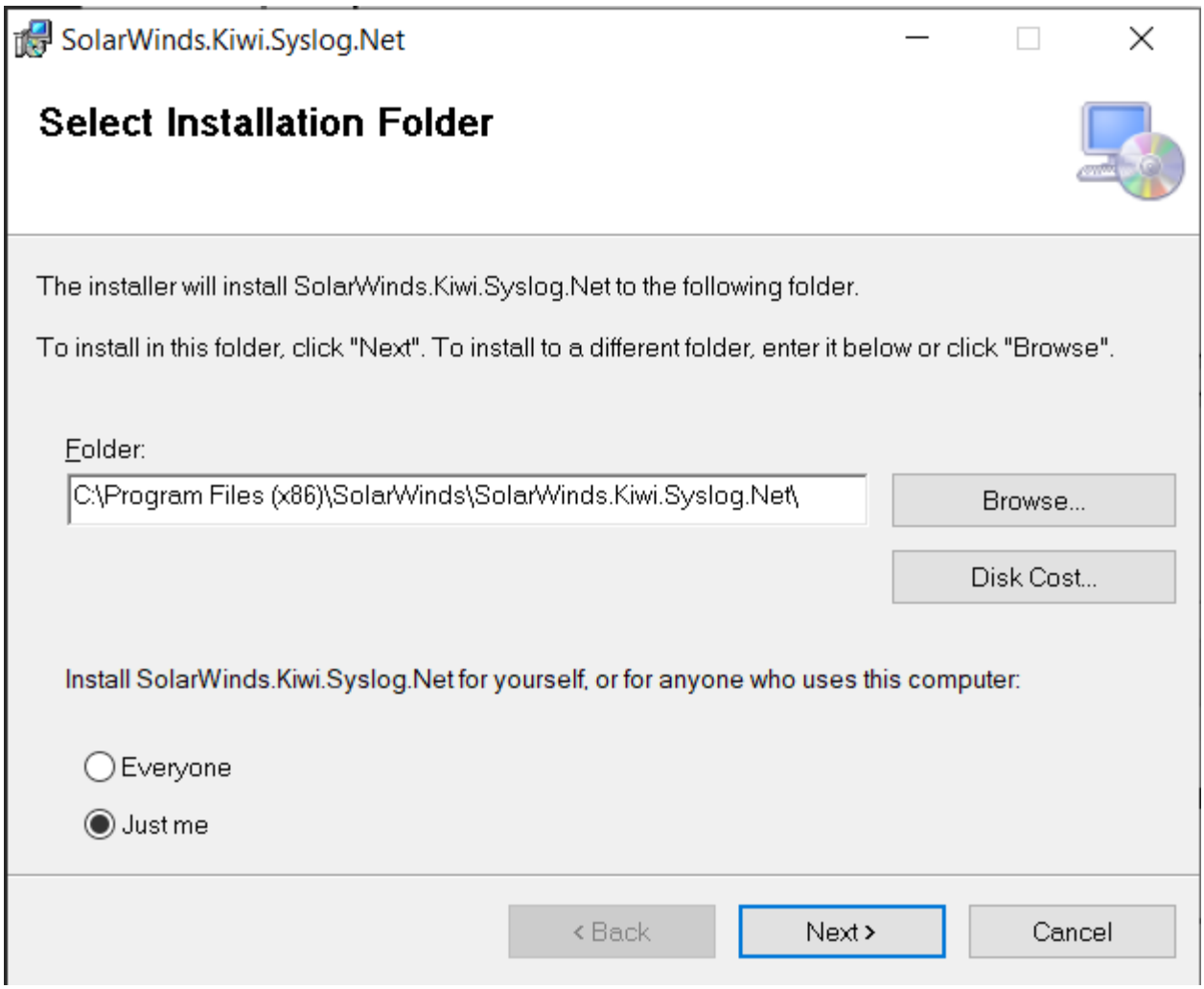


Рисунок 2.2 – Форма вибору шляху установки застосунку

Далі необхідно обрати режим, у якому буде встановлено додаток. На рисунку 2.3 зображено форму вибору режиму роботи застосунку: service mode – додаток працюватиме у режимі Windows сервісу, application mode – як звичайна аплікація.

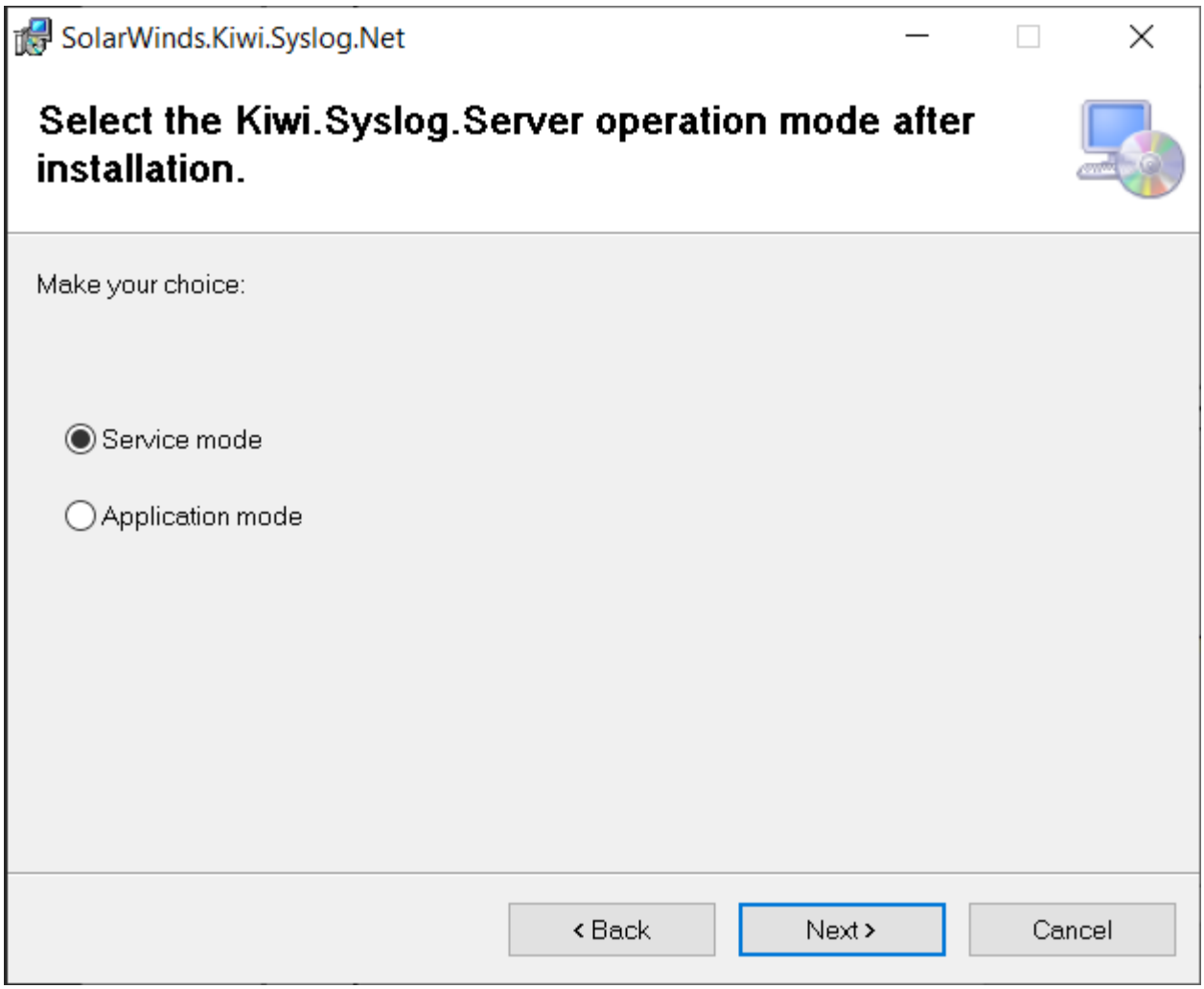


Рисунок 2.3 – Форма вибору режиму роботи застосунку

На наступній формі необхідно підтвердити рішення щодо процесу інсталяції, які були зроблені раніше (рисунок 2.4).

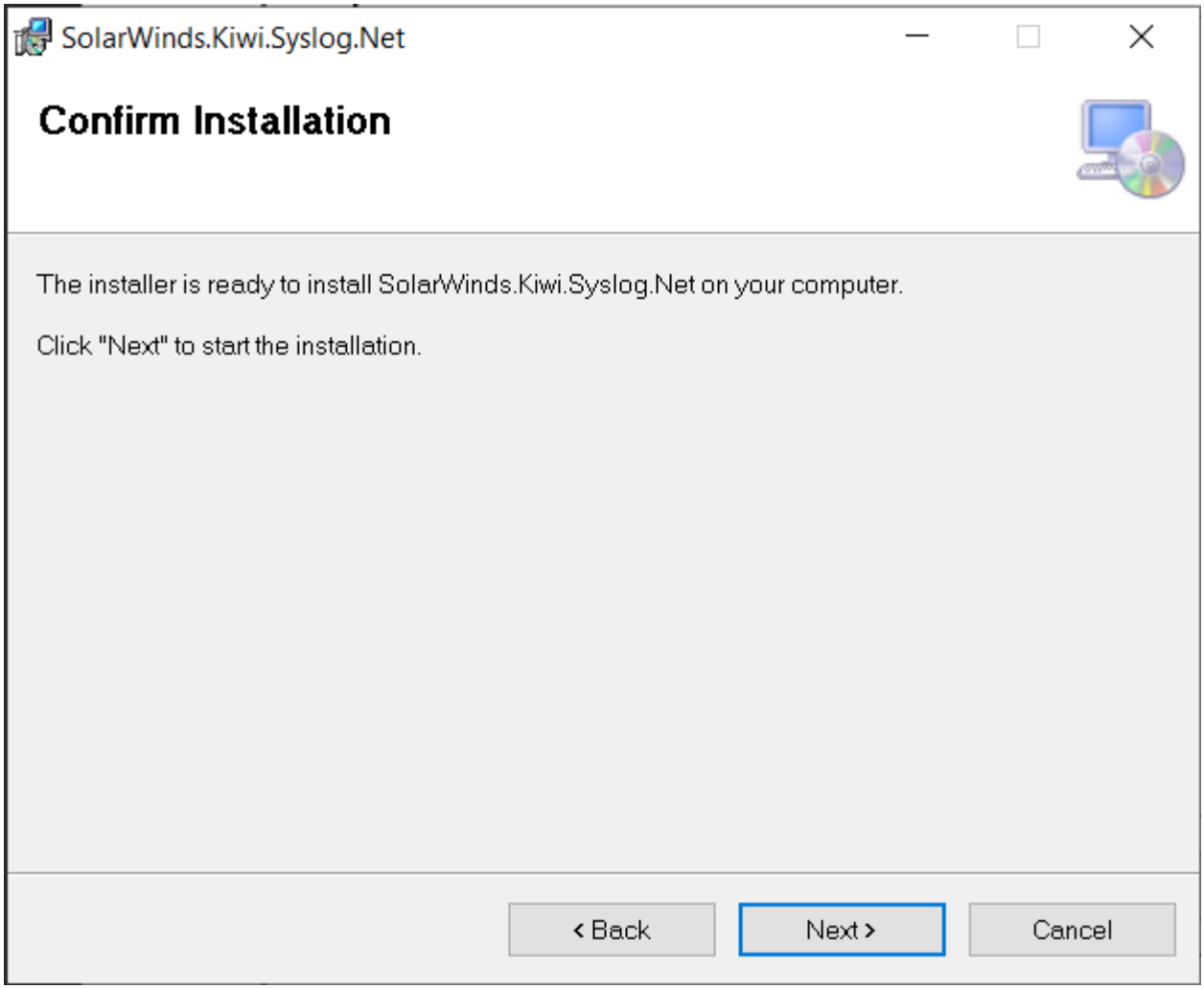


Рисунок 2.4 – Вікно підтвердження рішень про установку

Після натискання кнопки Next з’явиться вікно з прогрес баром процесу установки (рисунок 2.5).

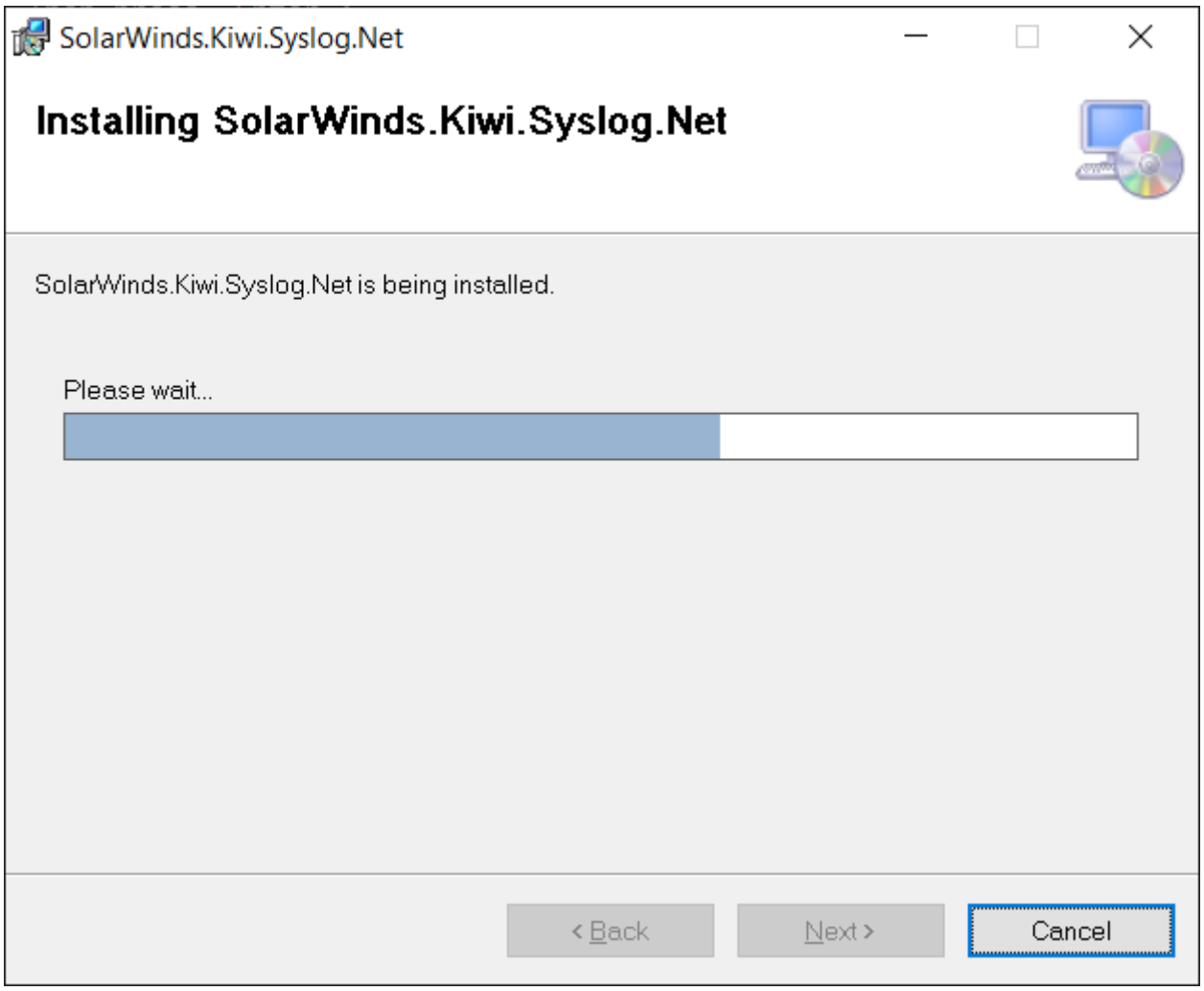


Рисунок 2.5 – Відображення прогресу процесу інсталяції

Під час безпосередньо установки та створення системних файлів з налаштуваннями за замовчуванням необхідно задати пароль адміністратора системи. На рисунку 2.6 зображено консоль вводу паролю адміністратора системи та вимоги до самого паролю.

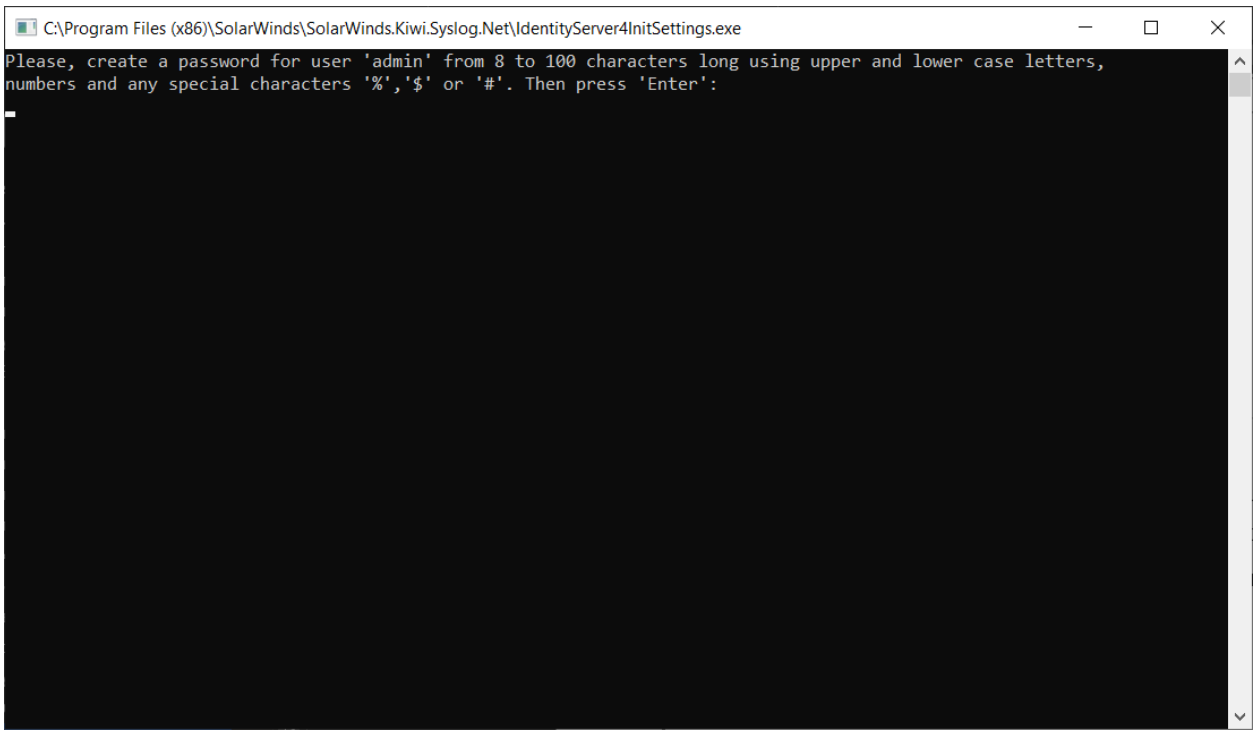


Рисунок 2.6 – Консоль вводу паролю адміністратора системи

Після введення паролю процес встановлення завершиться та на екрані з’явиться вікно, яке сповістить про успішну установку (рисунок 2.7).

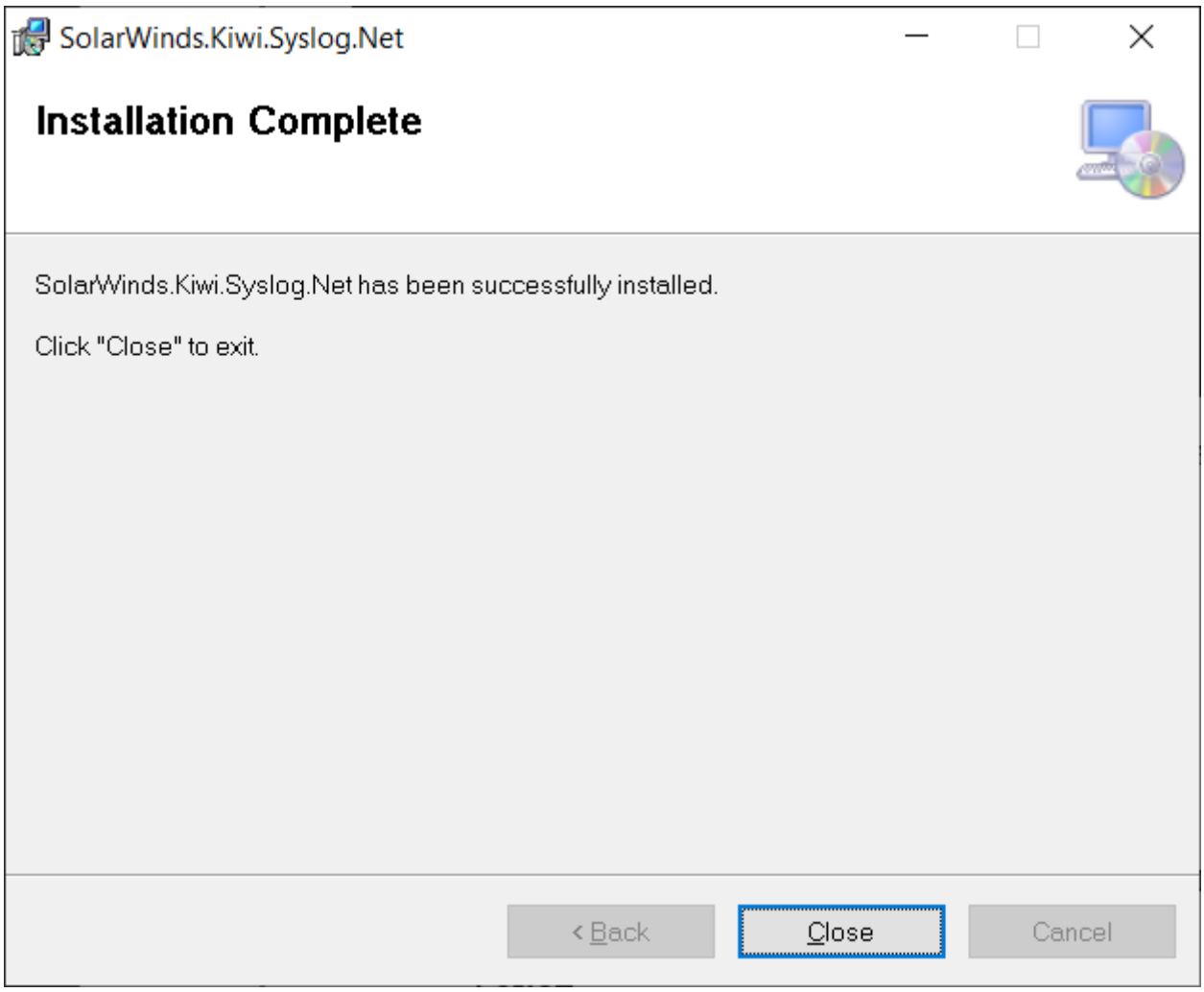


Рисунок 2.7 – Вікно завершення процесу встановлення

Наразі додаток встановлено, тож можна запускати сервер та повністю його адаптувати під користувача. Про те, що застосунок встановлено, можна впевнитися в меню Apps&features.

Після запуску серверу на екрані з’являється консоль з інформацією про старт роботи. На рисунку 2.8 зображено консоль с сервісними логами про те, що сервіс запущено, розпочато прослуховування повідомлень протоколами UDP та TCP за портами за замовчуванням.

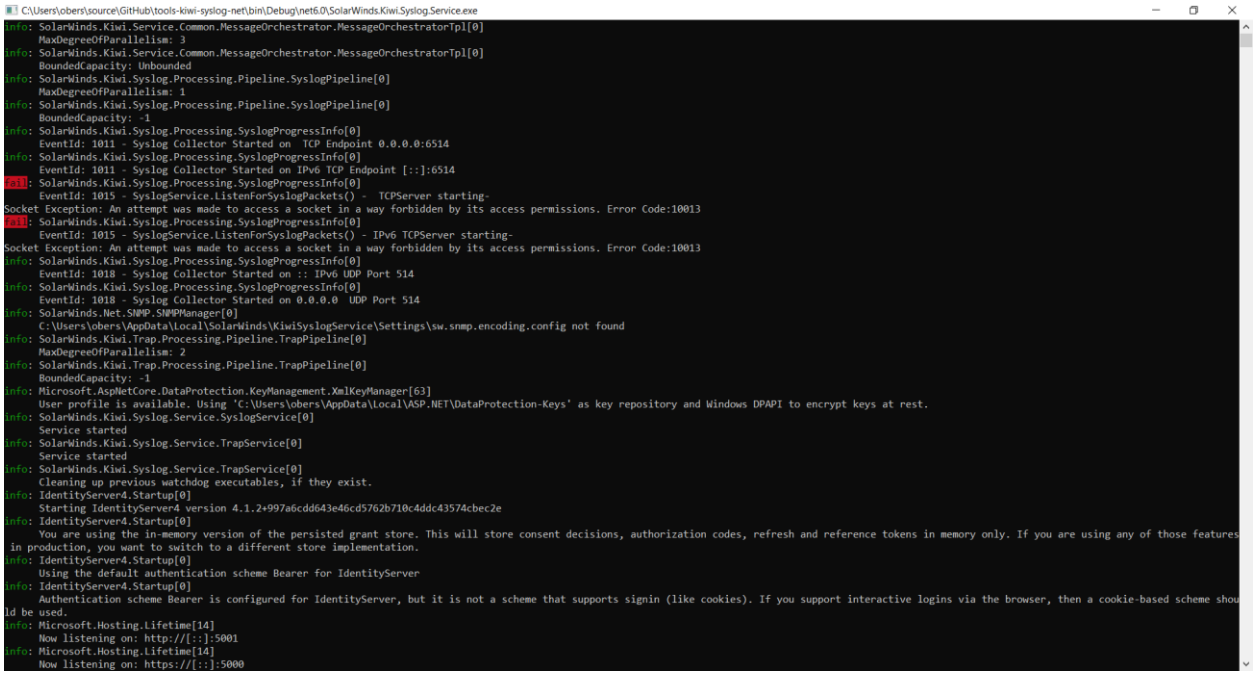


Рисунок 2.8 – Консоль серверу після старту додатку

2.3 Перевірка коректної роботи

Після успішного завантаження на робочому столі користувача повинні з’являтися два ярлика. Перший ярлик для запуску безпосередньо серверу. При натисканні на нього повинна з’явитися консоль, у якій буде відображено інформацію про старт серверу та систем прослуховування повідомлень. Другий ярлик для запуску клієнтського інтерфейсу. При натисканні на нього повинно відкритися вікно браузера за замовчування та вікно авторизації у системі.

3 РОБОТА ІЗ ЗАСТОСУНКОМ

Під час запуску клієнтської частини програмного застосунку відкривається вікно браузера за замовчування та вікно авторизації у системі. На рисунку 3.1 зображено вікно авторизації користувача у системі.

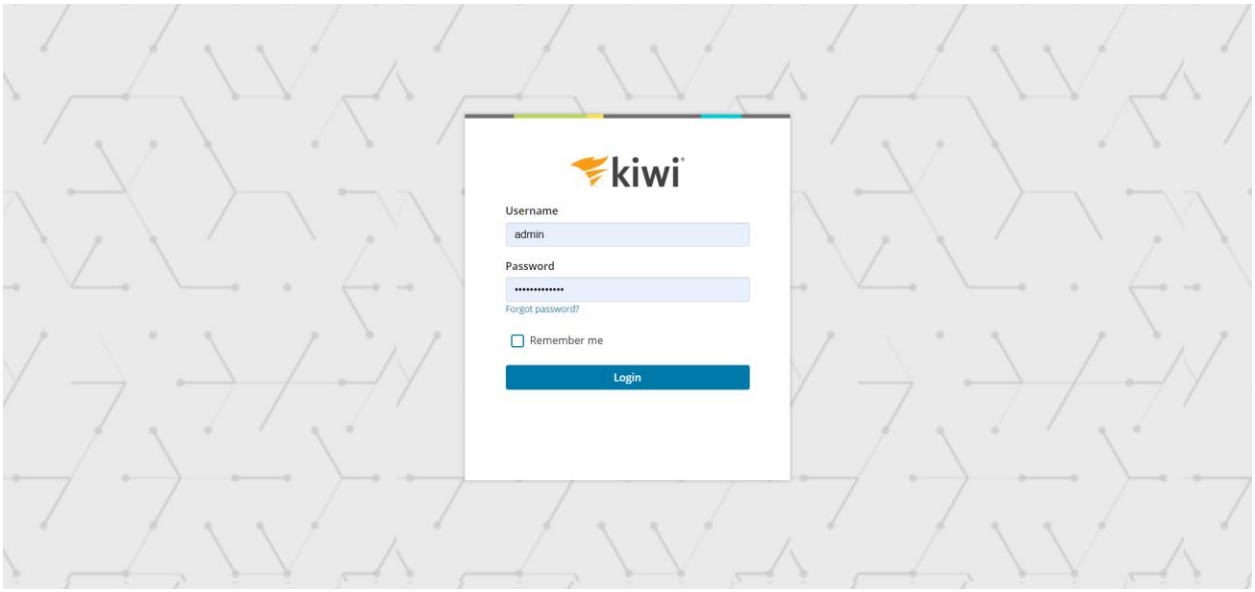


Рисунок 3.1 – Сторінка авторизації

Після вдалого проходження авторизації користувач потрапляє на сторінку зі статистичним дашбордом. На рисунку 3.2 зображено приклад дашборду, який може бачити користувач.

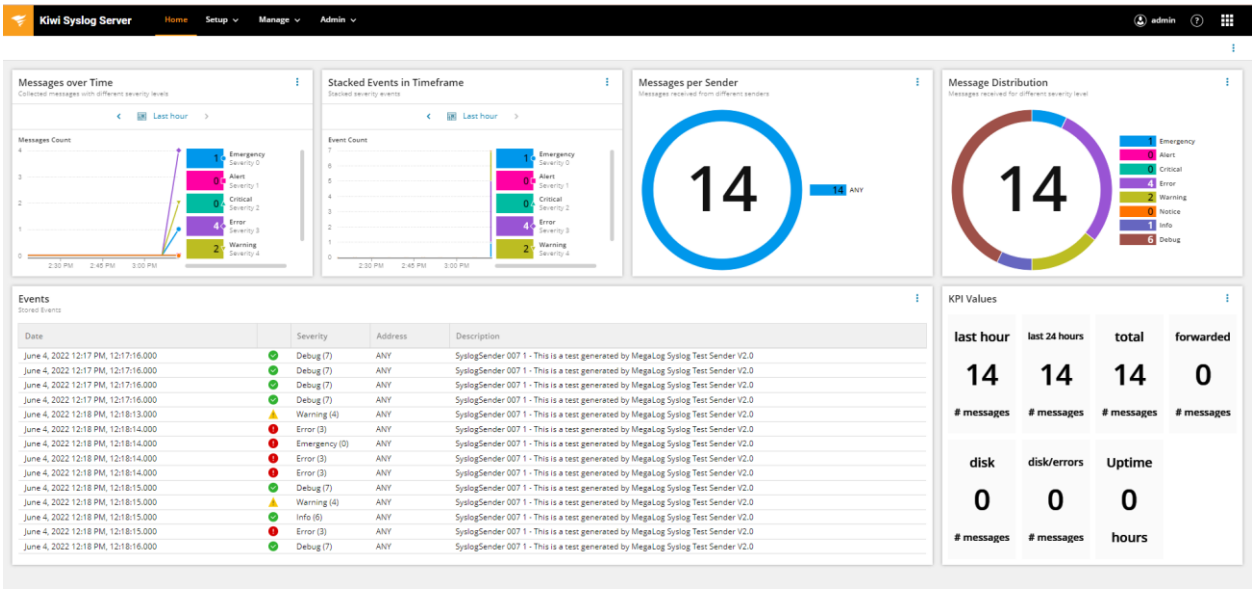


Рисунок 3.2 – Панель статистики

Користувач має можливість редагувати формат відображення інформації на дашборді та формат графіків. Аби редагувати дашборд необхідно натиснути на «три крапки» у верхньому правому куті та активувати режим Edit dashboard. На рисунку 3.3 зображено можливість редагувати панель статистичної інформації.

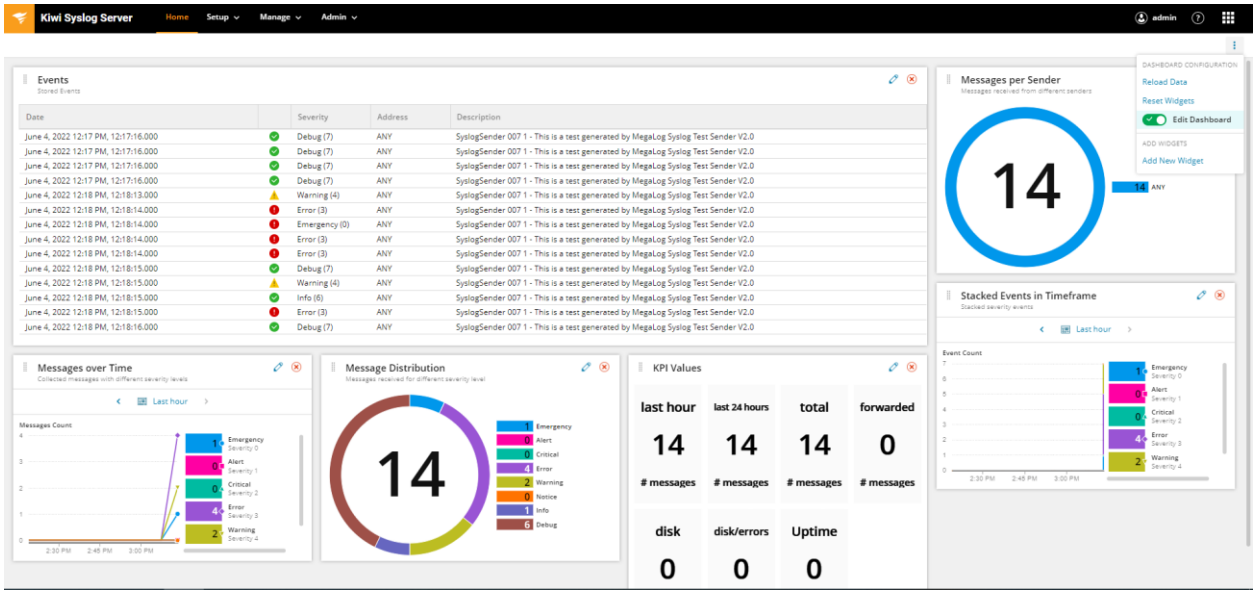


Рисунок 3.3 – Результат редагування вигляду дашборду

Для того, щоб перехоплене повідомлення було залоговано до файла, необхідно створити правило. Щоб створити правило, треба перейти у вкладку “New rule” у верхньому меню “Setup”(рисунок 3.4), вказати назву та натиснути далі.

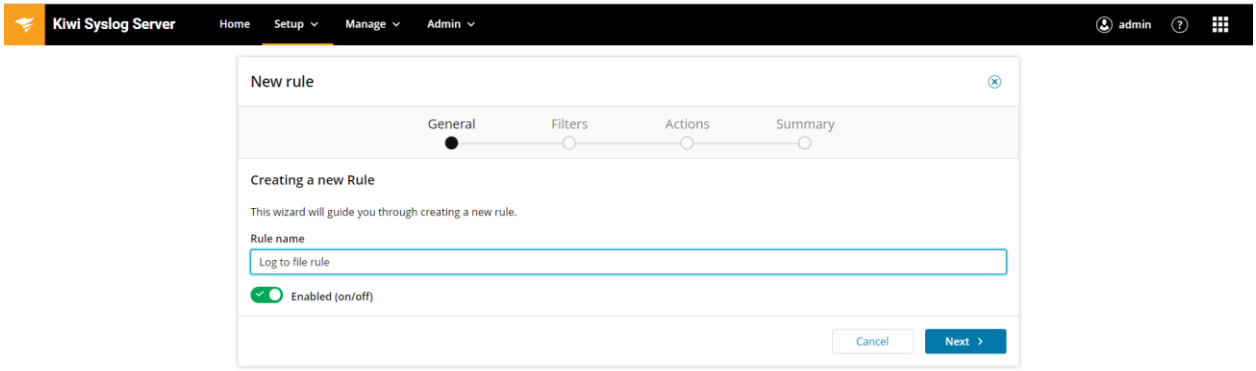


Рисунок 3.4 – Створення нового правила

Аби створити фільтр необхідно у меню створення фільтрів (рисунок 3.5) обрати “Add filter”, тип фільтру та ввести деталі в залежності від типу. На рисунку 3.6 зображено вікно деталей створюваного фільтру.

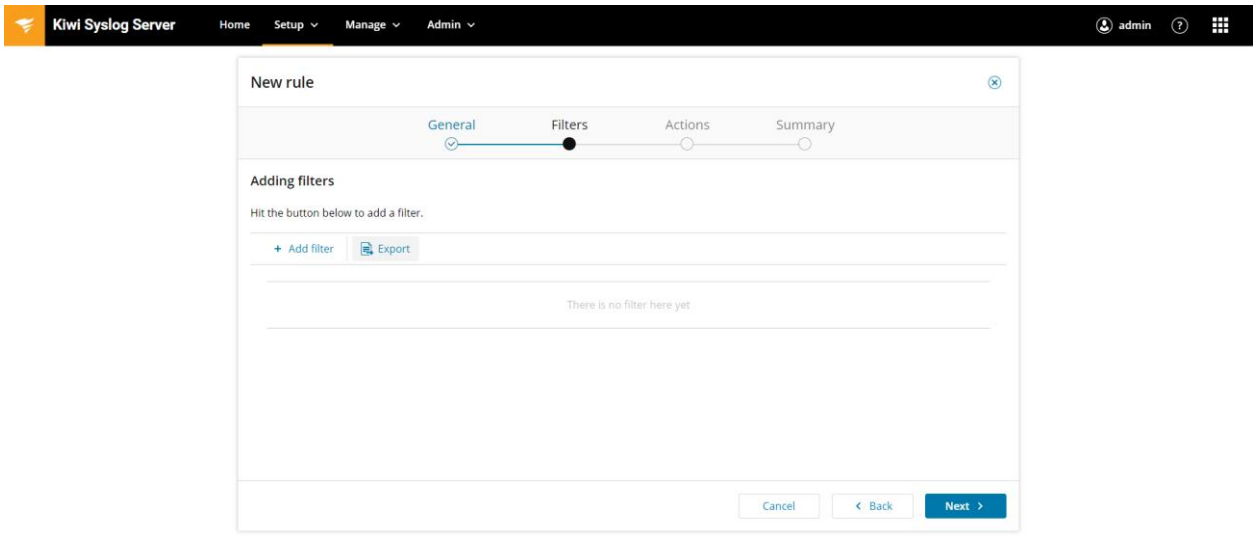


Рисунок 3.5 – Вікно створення фільтрів

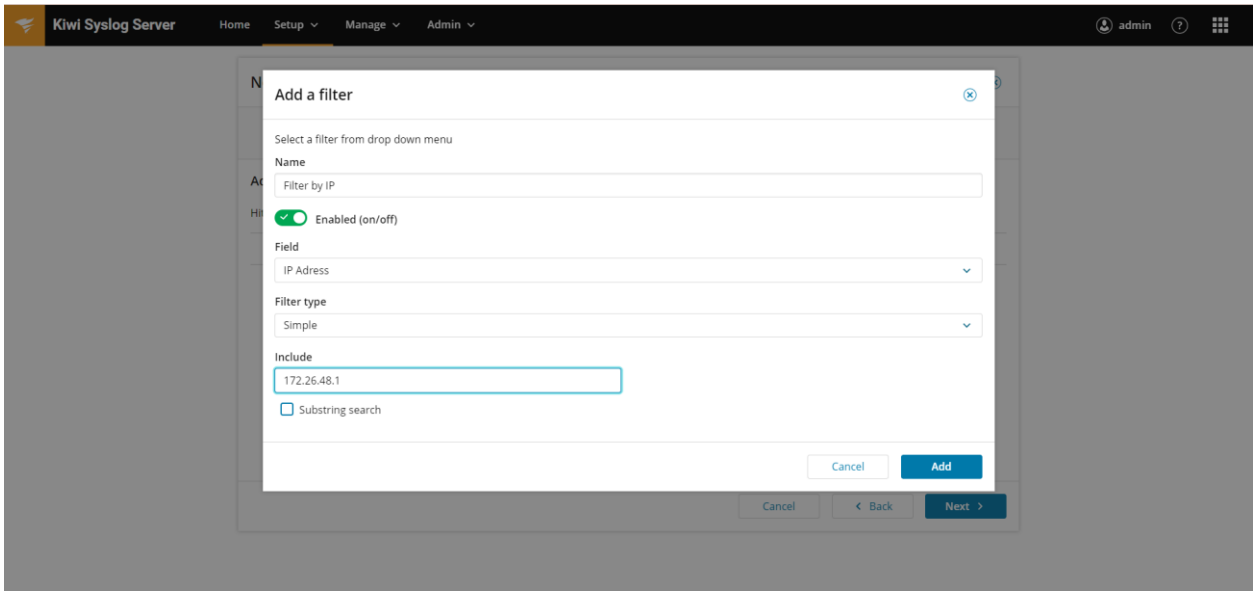


Рисунок 3.6 – Вікно деталей створюваного фільтру

На рисунку 3.7 зображено менеджер правил після створення фільтру.

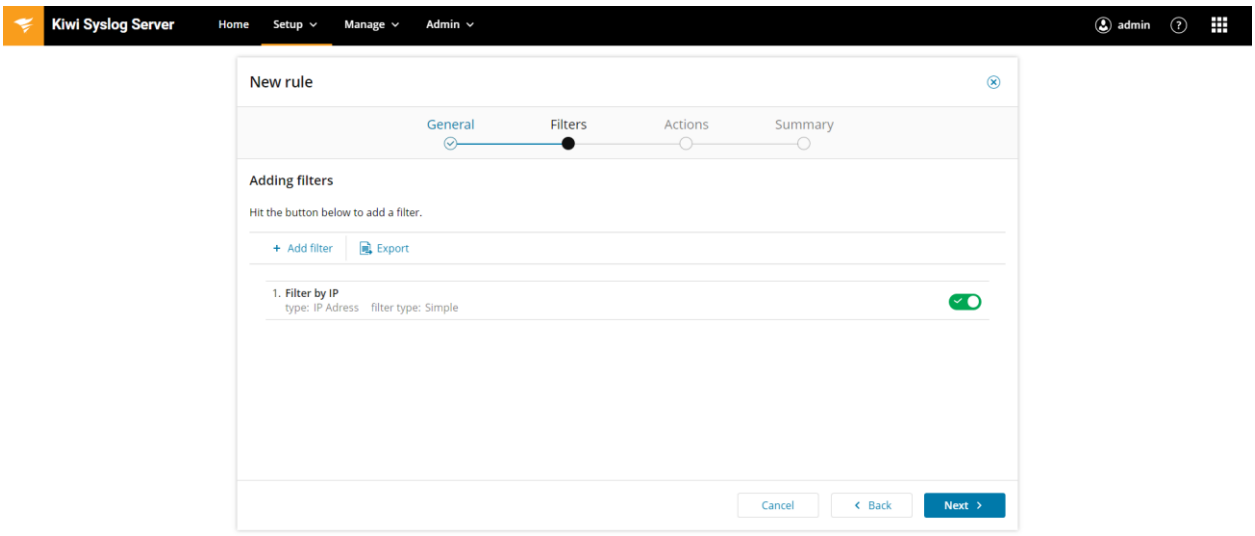


Рисунок 3.7 – Менеджер правил після створення фільтру

Фільтрів може бути додано будь-яка кількість різних типів – за IP-адресою, за ім'ям хоста, за текстом повідомлення чи його частиною, за протоколом, за пріоритетом чи часом доби. На рисунку 3.8 зображено перелік типів фільтрів, які можуть бути створені.

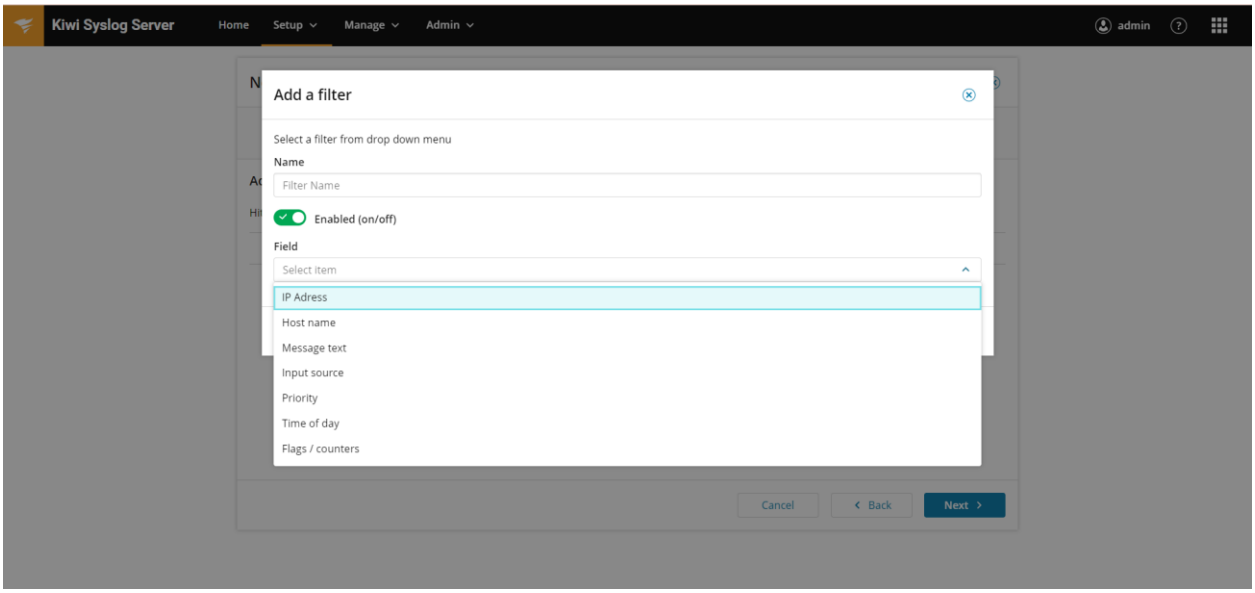


Рисунок 3.8 – Перелік типів фільтрів

Наступним у менеджері правил є вікно створення дії (рисунок 3.9). Серед наявних дій є логування у текстовий файл, пересилання повідомлення на інший сервер, надсилання e-mail повідомлення, виконання PowerShell скрипту, надсилання нового Syslog повідомлення з певними властивостями та логування до Event Log'у.

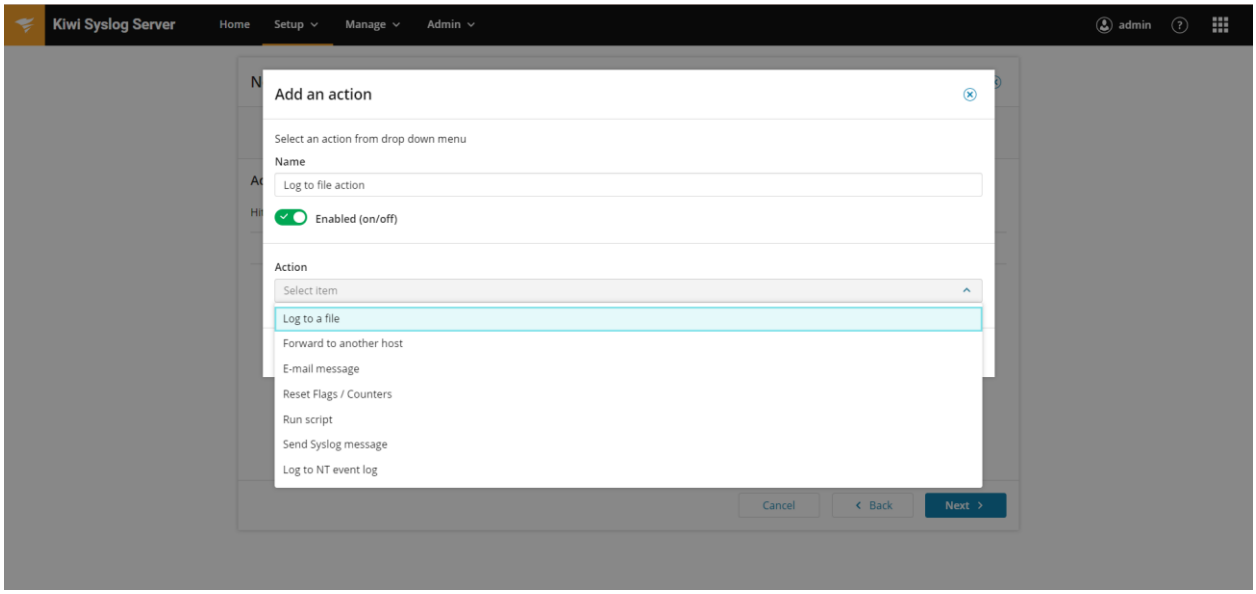


Рисунок 3.9 – Перелік можливих дій

З випадаючого списку необхідно обрати пункт Log to a file та заповнити відповідні деталі. Для цієї дії необхідно вказати шлях до файлу та формат запису інформації. На рисунку 3.10 наведені можливі формати логування повідомлень.

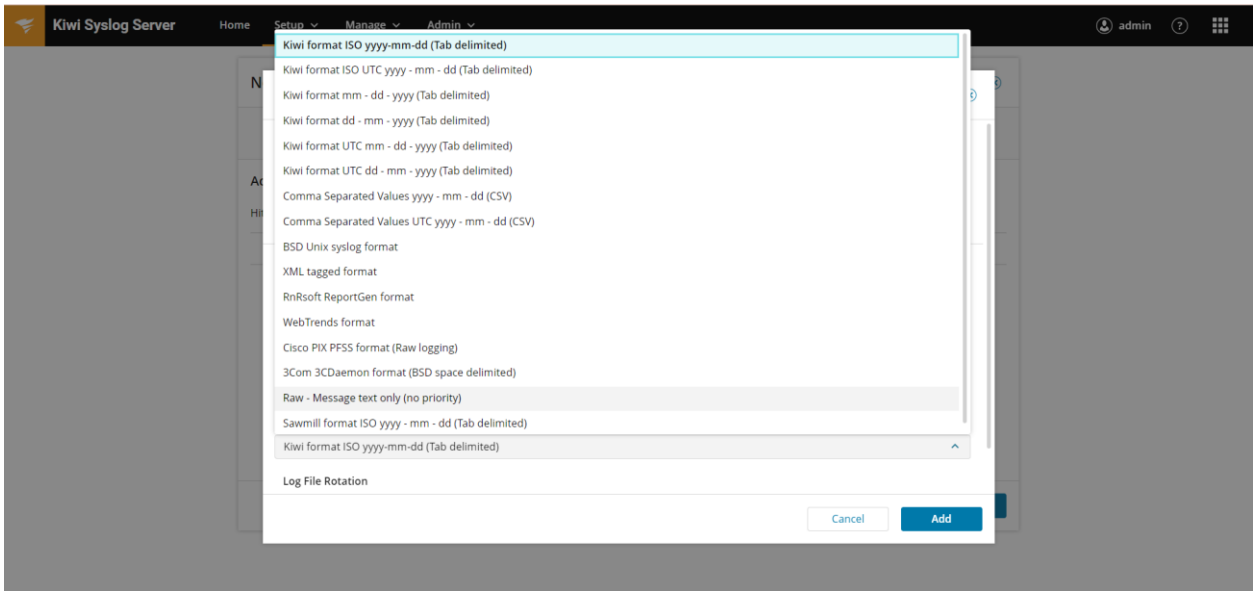


Рисунок 3.10 – Формати логування повідомлень у файл

На рисунку 3.11 наведено деталі створюваної дії.

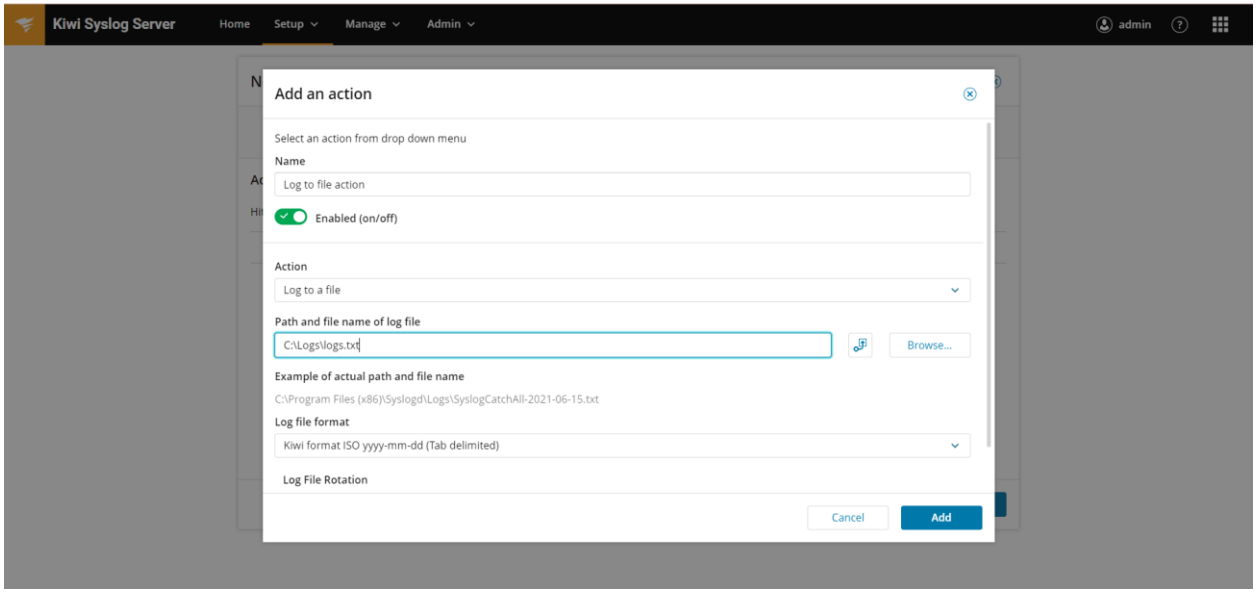


Рисунок 3.11 – Деталі дії Log to file

Після додання дії та натискання кнопки Next буде відображено менеджер створення правил. Необхідно підтвердити всі внесені зміни та створити правило. На рисунку 3.12 зображено менеджер створення правил перед збереженням кінцевих змін.

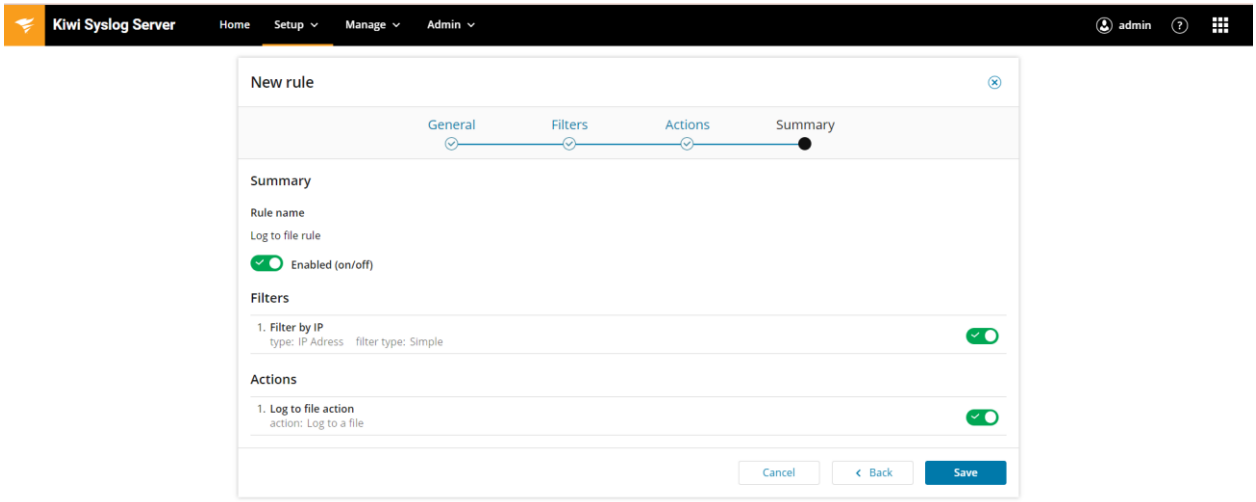


Рисунок 3.12 – Менеджер створення правил перед збереженням змін

Створене правило з’явиться у зальному списку правил (рисунок 3.13)

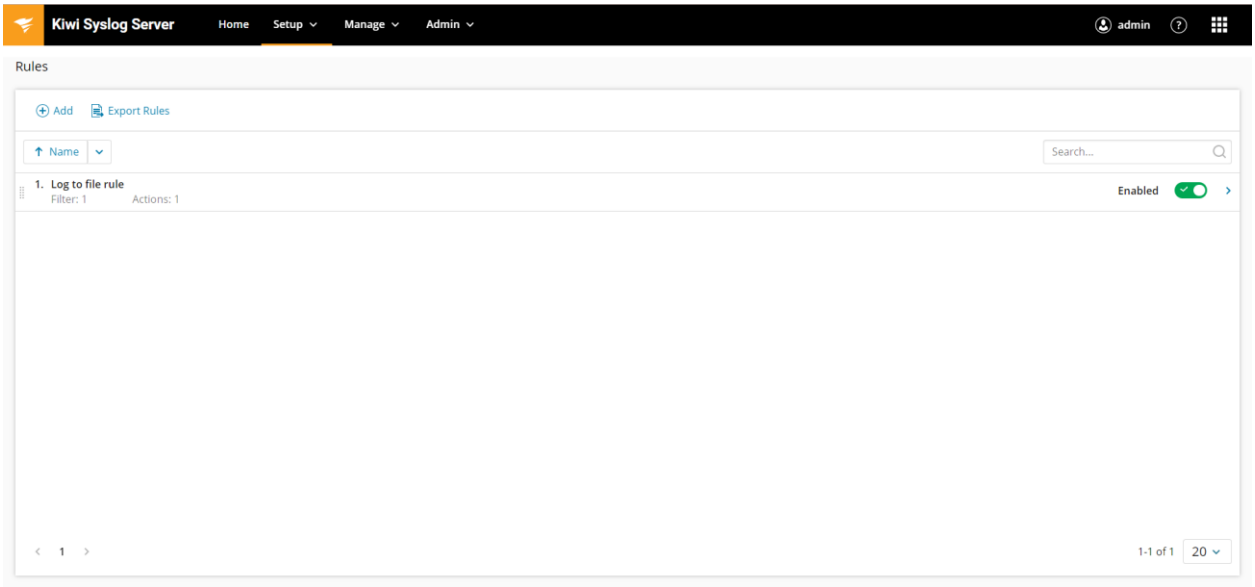


Рисунок 3.13 – Створене правило у загальному списку правил

Для того, аби перевірити працездатність створеного правила, необхідно надіслати тестове повідомлення. Для цього можна використовувати сторонній додаток. Було надіслано 5 тестових повідомлень. На рисунку 3.14 зображено результат роботи правила.

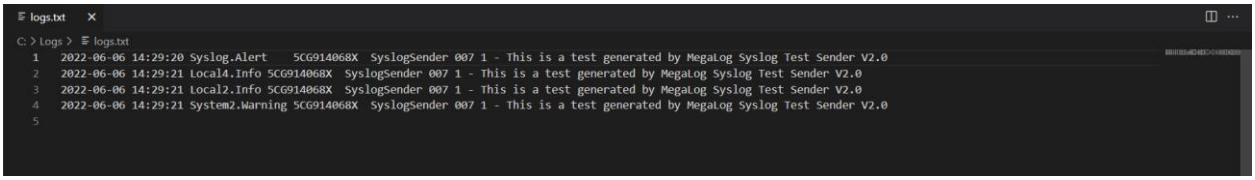


Рисунок 3.14 – Файл результату роботи створеного правилу

За аналогією можна створювати безліч інших правил із різноманітними наборами фільтрів та дії. Є можливість створити правило без фільтрів. В такому випадку дій, додані в межах цього правила, виконуватимуться для кожного перехопленого повідомлення.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КЛАСИФІКАЦІЇ ЛОГІВ ЛОКАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ МЕРЕЖІ

Графічний матеріал

КПІ.ІП-8104.045490.05.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юрій ОЛІЙНИК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Олександра БЕРШАЦЬКА

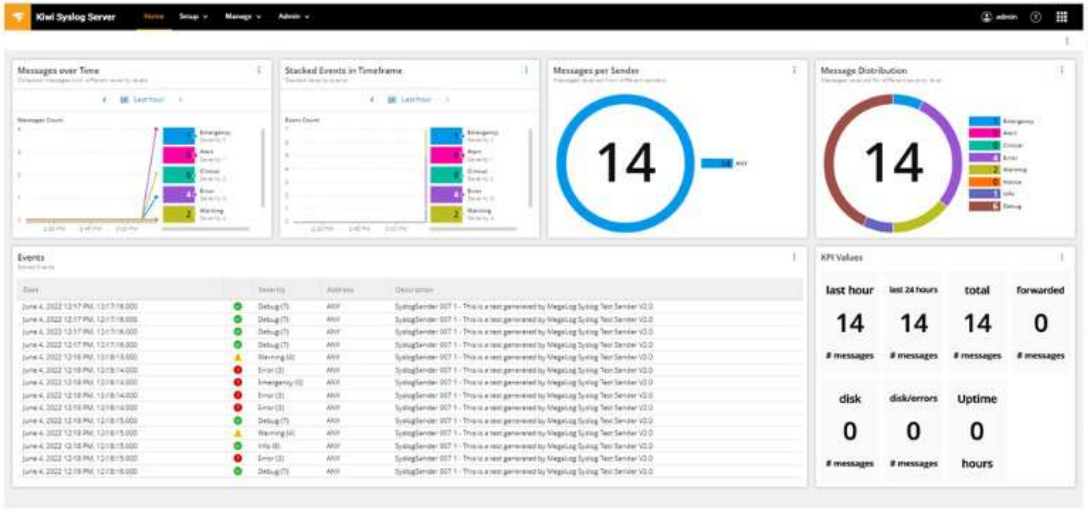
Київ – 2022

КПІ.ІП-8104.045490.05.99.СС

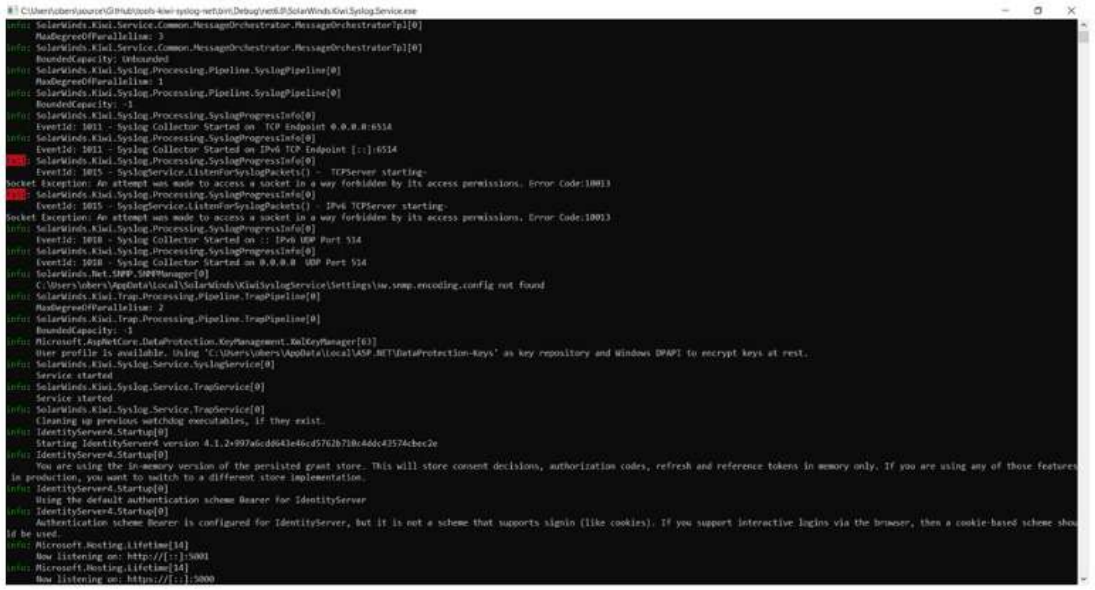
					КПІ.ІП-8104.045490.05.99.СС	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання	
Розробив		Бершацька О.О				Аркуш 1 / Аркушів 1
Перевірив		Олійник Ю.А.				
Т. кон.						
Н. кон.		Ліщук К.І.			Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювалюної мережі	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81
Затвердив		Олійник Ю.А.				

КПІ.ІП-8104.045490.05.99.СС

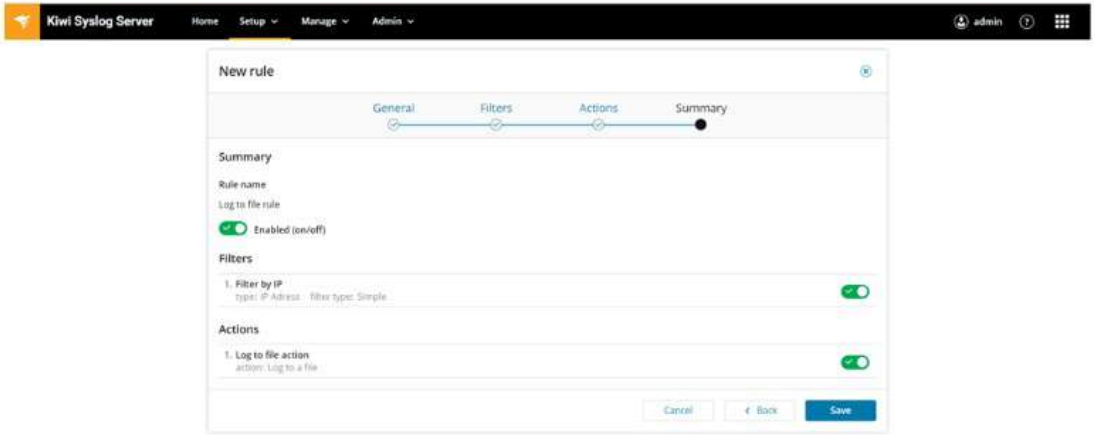
					КПІ.ІП-8104.045490.05.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна бізнес-процесів
Розробив		Бершацька О.О			Літера: — Маса: — Масштаб: —
Перевірив		Олійник Ю.А.			
Т. кон.					Аркуш 1 / Аркушів 1
					Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювалюної мережі
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81
Затвердив		Олійник Ю.А.			



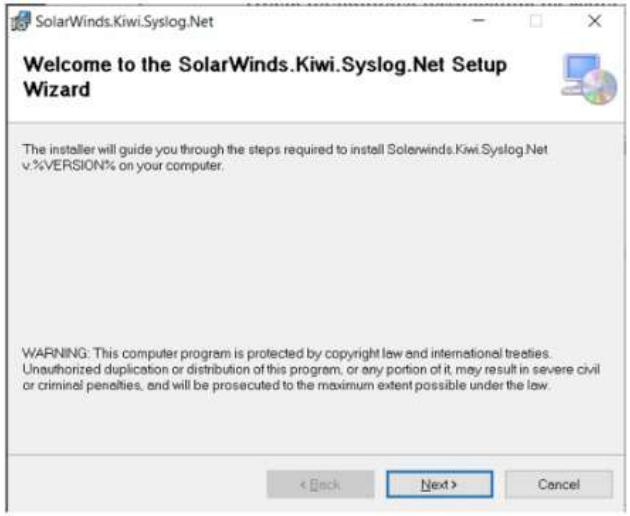
Панель статистики та результатів роботи серверу



Консоль серверу після старту додатку



Менеджер створення правил перед збереженням змін



Менеджер установки Syslog Server

					КПІ.ІП-8104.045490.05.99.КЗ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера / Маса / Масштаб
Розробив		Бершацька О.О				
Перевірив		Олійник Ю.А.				Аркуш 1 / Аркушів 1
Т. кон.						
					Програмне забезпечення моніторингу та класифікації системних логів локальної обчислювалюної мережі	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81
Н. кон.		Ліщук К.І.				
Затвердив		Олійник Ю.А.				