Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп’ютерних систем
Кафедра системного програмного забезпечення

Лещенко Катерина Олегівна,
студентка групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмний продукт для пошуку проблемних місць у програмних засобах за допомогою графічного інтерфейсу

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Пригожев Олександр Сергійович,
канд. техн. наук, доцент

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України
Одеський національний політехнічний університет
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Любченко В. В.
«___» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лещенко Катерина Олегівна, студентка групи АС-161

1. Тема роботи: Програмний продукт для пошуку проблемних місць у програмних засобах за допомогою графічного інтерфейсу

Керівник роботи: Пригожев Олександр Сергійович, канд. техн. наук, доцент

затверджені наказом ректора від «25» жовтня 2021 р. № 374-в

2. Зміст роботи:
Проаналізувати існуючі аналоги.
Розробити методи для пошуку логів.
Визначити вимоги до програмного продукту.
Описати послідовність дій користувача у різних ситуаціях.
Спроектувати інтерфейс користувача та порівняти з тим що вийшло наприкінці.
Виконати програмну реалізацію та описати її.
Виконати тестування програмного продукту.
Провести розрахунок ефективності програмного продукту

3. Перелік ілюстративного матеріалу:

1 слайд – мета, основні задачі програмного продукту; 2 слайд – опис предметної області та проблеми; 3 слайд – аналіз аналогів; 4 слайд – методи пошуку логів; 5 слайд – варіанти використання; 6 слайд – основна сторінка програмного продукту; 7 слайд – структура програмного продукту; 8 слайд – діаграма класів; 9 слайд – схема бази даних; 10 слайд – результати тестування; 11 слайд – розрахунок ефективності програмного продукту; 12 слайд – висновки; 13 слайд – конференції у яких дана тема була опублікована

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-6	Пригожев О. С. , канд. техн. наук, доцент	30.08.2021	25.11.2021

5. Дата видачі завдання: «30» серпня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Аналіз аналогів	10.09.21	виконано
2	Розробка методів пошуку логів	18.09.21	виконано
3	Специфікація вимог до системи	27.09.21	виконано
4	Опис послідовностей дій користувача	10.10.21	виконано
5	Програмна реалізація та її опис	1.11.21	виконано
6	Тестування	09.11.21	виконано
7	Розрахунок ефективності	10.11.21	виконано

Здобувач вищої освіти К. О. Лещенко

Керівник роботи О. С. Пригожев

АНОТАЦІЯ

Метою роботи є зменшення часу на пошук проблемних місць у програмному продукті за рахунок графічного інтерфейсу, що дозволяє аналізувати отримані логи. Технологіями розробки є IntelliJ IDEA, як середовище розробки, мова програмування Java, на якій написано основну частину програми, фреймворки Angular за допомогою якого написано фронтенд програми, та Spring Framework, який спростив написання бекендної частини програми. Як результат роботи виконано програмну реалізацію системи для обробки логів, яка дозволяє виконувати збір, пошук, фільтрацію та сортування логів, що розташовані на певних серверах у файлах, розташованих за певним адресом. Логи можна перевірити у режимі реального часу.

Ключові слова: лог, збір, пошук, фільтрація, файл, сервер, реальний час, фронтенд, бекенд, фреймворк.

ABSTRACT

The purpose of the work is to reduce the time spent searching for problem areas in the software product due to the graphical interface that allows you to analyze the received logs. Development technologies are IntelliJ IDEA, as a development environment, the Java programming language, in which the main part of the program is written, the Angular frameworks with which the front-end of the program is written, and the Spring Framework, which simplified the writing of the back-end part of the program. As a result of the work, a software implementation of a system for processing logs was made, which allows collecting, searching, filtering and sorting logs that are located on certain servers in files located at a certain address. Logs can be checked in real time.

Keywords: log, collection, search, filtering, file, server, real time, frontend, backend, framework.

ВСТУП

У створеній кваліфікаційній роботі розглядаються питання пошуку проблемних місць у програмному продукті, що зараз у епоху коли комп'ютери та комп'ютерні програми використовуються на кожному кроці. Зараз комп'ютерні програми спрощують життя людини у кожній сфері і маркетингу, і обслуговування, і транспорту, і медицини, тощо. Але майже завжди люди, що створюють програмні продукти, та ті що використовують потім їх думають по різному. З цього виникає проблема, яка інколи призводить до неправильної роботи усього програмного продукту. Відстежити цю проблему час від часу становиться дуже важко, або інколи навіть і не можливо. Для цього робота кожного програмного продукту супроводжується логами. Логи – службова інформація, що накопичується під час роботи програмного продукту, та відображає нормальну, або помилкову поведінку системи. Отже, вони показують як те, що програмний продукт працює правильно, так і те що виникла якась проблемна ситуація. Для кращого розуміння правильності роботи програмного продукту зазвичай цих логів дуже багато, отже знайти щось важливе досить складно. Існуючі на сьогоднішній день програми для збору логів потребують вивчення додаткових мов, щоб була можливість швидко працювати з великою кількістю логів.

З вищесказаного можна визначити, що проблемою є складність пошуку проблемних місць у програмному продукті.

Створення даного програмного продукту зможе допомогти і програмісту швидше знаходити проблеми у програмному засобі та виправляти їх для наступної покращеної версії. Тому баги можуть виникати і на етапі створення програмного продукту, хоч їх і вирішення не коштує на стільки ж дорого як і на етапі коли програмний продукт видають замовнику, проте вирішувати їх необхідно також досить швидко.

Тому, актуальність цієї роботи полягає у тому, що створюваний програмний продукт допоможе фахівцям DevOps, що займаються покращенням роботи

програмного продукту для користувачів, щоб вона дійсно пришвидшити роботу у певній сфері, а отже і пришвидшити етап пошуку проблем у програмному продукті.

Метою даної роботи є пришвидшення пошуку проблемних місць (похибок) у програмному продукті за рахунок графічного інтерфейсу для збору (перенесення з усіх необхідних адрес накопичених даних в даний програмний продукт), фільтрації (процес визначення найбільш необхідних за часом та рівнем логів), та сортувань (процес упорядкування логів за однією з трьох необхідних характеристик: час, рівень, текст) логів, що знаходяться на серверах, у файлах розташованих за певною адресою.

Зараз ця проблема вирішується, але потребує витрачення багато часу на вивчення необхідних мов, що допоможуть виконувати, та зменшувати час на пошук необхідних проблемних місць. Отже метрикою буде час.

Як методи дослідження поставленої мети було обрано спостереження, порівняння, аналіз. За допомогою методу спостереження під час переддипломної практики проводилося спостереження за проблемами які виникають у програмістів. Найбільш частою виявилася саме проблема аналізу логів. Отже, на наступному етапі було вирішено провести порівняння існуючих аналогів, та виявлення недоліків, на які і була спрямована подальша розробка програмного продукту. Після розробки було проведено аналіз створеного продукту, та порівняння кількості затраченого часу на пошук проблемних місць у програмному продукті.

Результати кваліфікаційної роботи було опубліковано у вигляді тез до конференції "Сучасні інформаційні технології 2021" ("Modern Information Technology 2021" – MIT-2021) [1] і у Міжнародній науково-практичній конференції "Технічне регулювання, метрологія, інформаційні та транспортні технології" ("Technical regulation, metrology, information and transport technologies") [2].

1 ПРОБЛЕМИ ІСНУЮЧИХ РЕДАКТОРІВ ЛОГІВ

1.1 Обґрунтування розробки

У даній дипломній роботі виконано розробку програмного продукту на тему «Програмний продукт для пошуку проблемних місць у програмних засобах за допомогою графічного інтерфейсу». За допомогою цього продукту буде виконуватися пошук проблемних місць у інших програмних засобах. Для цього будуть збиратися логи з усіх серверів, що додано користувачами, або адміністраторами до програмного продукту, обраних у цих серверах директоріях, та розташованих у них файлах. Зібрані логи сортуються за датою та рівнем, та фільтруються за рівнем, текстом, що присутній у логах, та датою. Також немала увага приділена безпеці, адже цим додатком користувачі надають данні до серверів, на яких може зберігатися секретна інформація [3]. Саме тому користувачей дозволено додавати до системи лише адміністраторам. Усі дані користувача, що використовуються для автентифікації, шифруються за допомогою токену. У ньому прописано ім’я користувача, роль, а також час закінчення дії токену. Після чого він підписується за допомогою секретного ключа, та шифрується за допомогою алгоритму HMAC256.

1.2 Аналіз аналогів

Жоден з аналогів як показано у таблиці 1.1 (де ГІ означає «за допомогою графічного інтерфейсу») не має достатнього механізму для пошуку проблемних місць у програмному продукті за рахунок графічного інтерфейсу. Також, як видно з таблиці, досить мало аналогів мають можливість зберігати логи, а це дуже важливо. Наприклад, при перегляді логів у реальному часі та визначенні проблемної ситуації необхідна можливість зберегти у файл логи, що буде відображати цю ситуацію, а також, інколи і причину її появи. Основні, найбільш цікаві аналоги описані нижче більш детально.

Logstash – це механізм збору даних з відкритим вихідним кодом з можливостями конвеєрної обробки даних в реальному часі. Logstash може динамічно ідентифікувати дані з різних джерел і нормалізувати їх, за допомогою обраних фільтрів [4]. Але найбільшим мінусом є те, що він працює тільки з набором продуктів Elastic.

Ще одним з аналогів є Fluentd. Fluentd – це збірник даних з відкритим кодом, який дозволяє уніфікувати збір та споживання даних для кращого використання та розуміння даних [5]. Але основним його недоліком є те, що в ньому відсутній пошук, фільтрація та сортування зібраних логів.

Таблиця 1.1 Порівняльна таблиця аналізу аналогів

Критерії/ПО	Graylog	Octopussy	Flume	Logstash	Fluentd	LOGalyze [9]	LogPacker [10]	Logwatch [11]	Syslog-ng [12]	Inav [13]	Створюваний програмний продукт
Пошук (ГІ)	-	-	-	-	-	-	-	-	-	-	+
Збір	+	+	+	+/-	+	+	+	+	+	+	+
Фільтрація (ГІ)	-	-	-	+/-	+/-	-	-	-	-	-	+
Сортування (ГІ)	-	-	-	-	-	+/-	+	-	-	-	+
Необхідність вивчання додаткових мов	+	+	+	+	+	+	+	+	+	+	-
Збереження логів	+	-	+	+	-	-	+	-	-	-	+
Перегляд логів у реальному часі	+	-	+	+	+	+	+	-	-	-	+

Apache Flume – це розподілене, надійне та доступне програмне забезпечення для ефективного збору, агрегації та переміщення великої кількості даних журналу [6]. Проте він також як і попередній не допомагає аналізувати логи. Він підтримується Apache Foundation, що каже про відсутність корпоративного плану.

Octopussy – рішення для управління журналами з відкритим вихідним кодом [7]. Він є найменш конкурентоспроможним створюваним програмним продуктом, бо не має навіть користувальницького інтерфейсу.

Найбільш конкурентоспроможним для даної роботи є аналог – Graylog, тому його розглянуто найбільш детально. Graylog2 – це безкоштовна open source система централізованого збору, зберігання і аналізу інформації, яка пишеться в syslog, graylog2 зроблений за концепцією DevOps [8]. Вона використовується для дуже великих проектів, виконує майже увесь необхідний функціонал. Також Graylog дозволяє зберігати у файл необхідну інформацію, але це буває не досить зручно адже логів зазвичай у ньому незчисленна кількість, тому зберегання у цьому випадку не є ефективним. Лише у випадку коли під час виводу логів у реальному часі з певного серверу помітили помилку, то цей аналог, як і створюваний програмний продукт дозволить зберегти це у окремий файл, щоб не загубити його. Тут слід зазначити, що одним з суттєвих мінусів є те, що для того, щоб відсортувати або відфільтрувати виробку необхідно прописувати регулярні вирази, що інколи досить складно. Однак, у створюваному програмному продукті все це можна зробити за допомогою інтерфейсу, що є швидше, та зручніше. Ще суттєвою відмінністю даного програмного продукту є можливість виставляти час, який логи повинні зберігатися, та частоту опитування серверів. Унікальність цього програмного продукту полягає у реалізації фільтрації і сортування логів за різними параметрами: датою, рівнем і тексту, а також вивід логів в реальному часі. Також він досить великоваговий у порівнянні зі створеним програмним продуктом.

При аналізі ринку програмних продуктів переглянуто основні аналоги. На основі цього, прийнято рішення про створення двох типів користувачів, один з яких включає в себе можливості перегляду логів, вибору серверу, управління списком серверів, налаштування облікового запису та авторизацію. Другий має усі можливості першого, а також може керувати користувачами, змінюючи їх ролі, додавати нових, та виконувати налаштування додатку за допомогою відповідних змінних.

2 МЕТОДИ ПОШУКУ ЛОГІВ

Для досягнення вищезазначеної мети, а саме є зменшення часу на пошук проблемних місць у програмному продукті за рахунок графічного інтерфейсу для збору фільтрації, та сортувань логів, що знаходяться на серверах, у файлах розташованих за певною адресою, обираються алгоритми які будуть використовуватися у програмному продукті. Методи аналізу зазвичай чутливі до масштабу даних; тип шкали вимірювання впливає на методи аналізу та розуміння даних [14], тому було вирішено порівнювати за допомогою блок-схем алгоритмів.

Основний функціонал даного програмного продукту полягає у перегляді логів, та пошуку там необхідної інформації. Виконувати пошук можна декількома способами.

Один з яких – переглядаючи кожен файл окремо (рис.2.1 а) – цей алгоритм досить затратний за часом, його складність складає n x m x t, де m – кількість файлів з логами, n – кількість директорій з файлами, t – час затрачений на пошук необхідної інформації. Для цього необхідно зайти до кожного файлу кожної директорії, та перевірити чи є там необхідна інформація. Однак, якщо вже є інформація про те у якому файлі лежить необхідний лог, то це значно зменшує час на пошук того логу у файлі.

Другий варіант – коли можна переглядати одразу усі файли з директорії (рис.2.1 б) – цей алгоритм у порівнянні з попереднім є менш складним. Його складність дорівнює n x t, де n – кількість директорій з файлами, t – час затрачений на пошук необхідної інформації. Цей алгоритм є дієвим лише коли є інформація про те у якій директорії зберігається необхідний файл. У іншому випадку час витрачений на пошук необхідної інформації буде завеликим.

Третій варіант – коли є інформація лише про сервер на якому зберігається необхідна інформація про помилку (рис.2.1 в). Для цього є варіант перегляданння логів у режимі реального часу, щоб коли з’явиться необхідний лог можна було одразу зберегти його у окремий файл і тоді інформація про помилку буде наприкінці цього файлу і не буде необхідності аналізувати логи з усього серверу.

Складність такого алгоритму дорівнює лише t, часу на пошук необхідної інформації з серверу.

Отже, проаналізувавши усі три варіанти перегляду логів було вирішено у роботі відтворити всі з них, бо у різних ситуаціях складність окремого з них може бути меншою ніж їх аналоги.

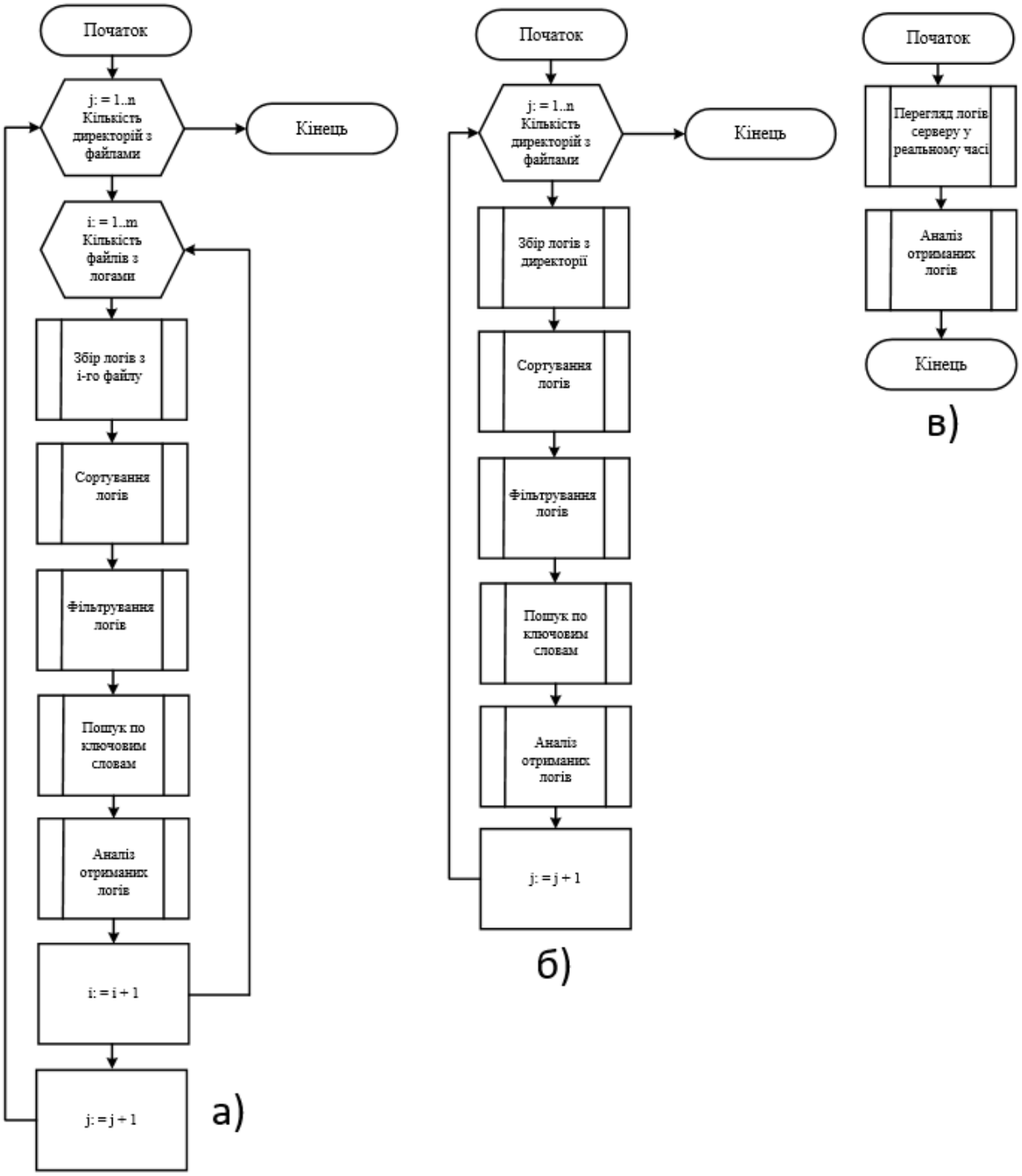

Рисунок 2.1 – Схема алгоритмів для перегляду логів

(де а – з файлу, б – з директорії, в – у реальному часі з серверу)

3 ВИЗНАЧЕННЯ ВИМОГ

3.1 Варіанти використання

Вимоги до програмного продукту представлені на діаграмі юзкейсів (рис. 3.1).

Можливості користувача включають в себе: перегляд логів, вибір сервера, управління списком серверів, налаштування облікового запису та авторизацію. Основна ідея продукту – це реалізація фільтрації і сортування логів за різними параметрами: датою, рівнем і тексту, а також вивід логів в реальному часі.

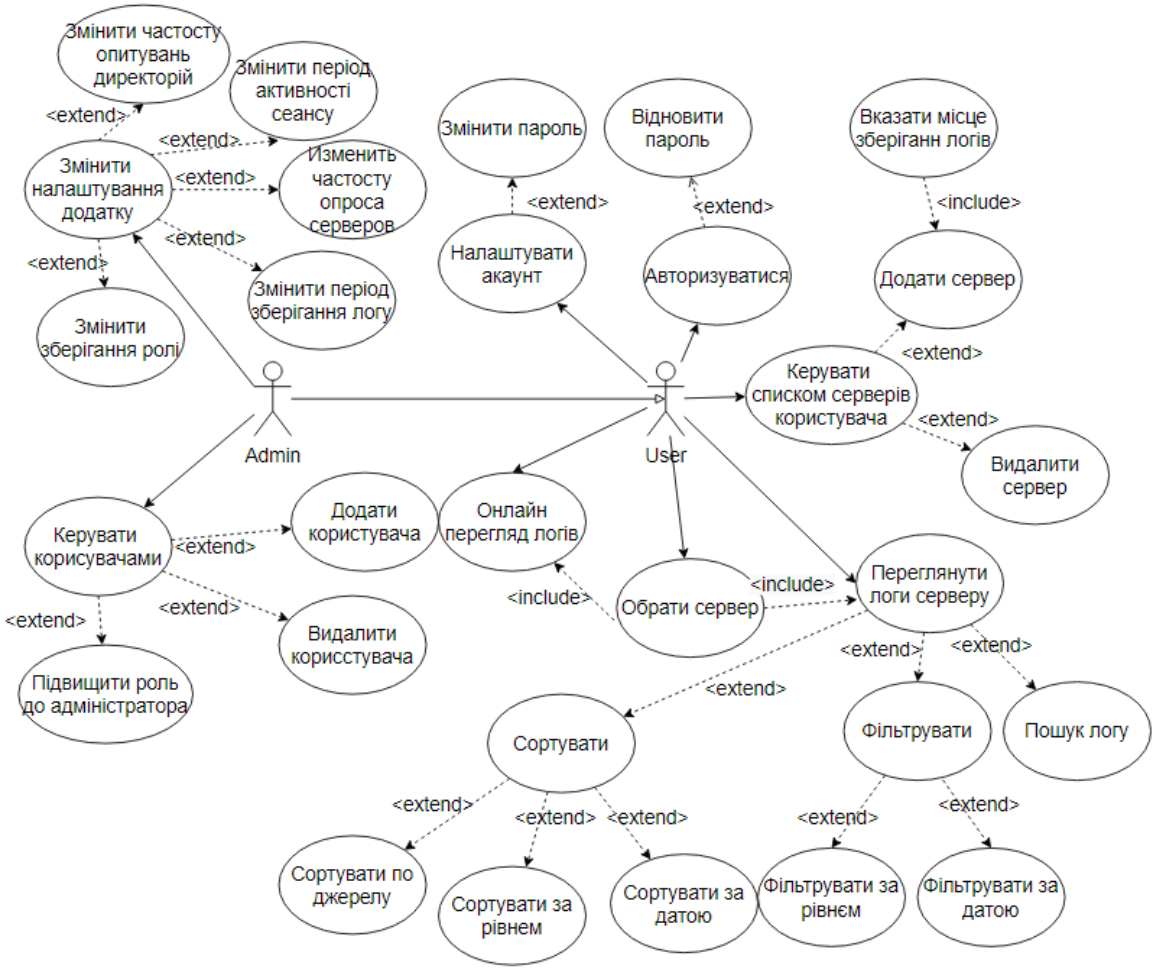

Рисунок 3.1 – Use Case діаграма

Адміністратор має повний спектр функціоналу користувача, та окрім цього, він ще може змінити налаштування програми. Наприклад, змінити частоту опитування серверів і період зберігання логів. Може управляти користувачами.

Створювати нових, підвищувати роль вже існуючих, видаляти, та змінювати налаштування.

Користувач при вході в систему вводить логін і пароль. Система перевіряє і видає повідомлення про успішну авторизації, або про помилку введених даних.

Якщо користувач забуває свій пароль, заходить у вікно відновлення пароля і вводить свій логін. Система перевіряє логін і якщо він вірний надсилає повідомлення на пошту з паролем, якщо немає – видає повідомлення про невірно введений логін і відмовляє користувачеві в можливості відновлення пароля.

Якщо користувач вирішує змінити пароль, він заходить у вікно зміни пароля і вводить старий і новий пароль. Система перевіряє старий пароль на правильність введення, та новий на розбіжність зі старим. Також пароль має декілька обмежень, такі як довжина, наявність літер великих та малих, та спец символів. І при правильно введених всіх даних змінюється пароль в системі.

Перегляд логів функціонує наступним чином. Система отримує запис з інформацією про сервер з БД, підключається до сервера, отримує запис логів від сервера. Система заповнює БД логами і виводить їх користувачеві.

Користувач вибирає один з логів і запитує інформацію про нього. Система видає всю доступну інформацію, якщо така є.

Додавання користувача в систему вимагає введення його логіна, паролю та адреси електронної пошти. Якщо логін і пошта не існують в БД, то система успішно зберігає нового користувача.

Видалення користувача відбувається за наступним алгоритмом. Адміністратор обирає зі списку існуючого користувача і підтверджує видалення. Система перевіряє наявність користувача в системі і видаляє всю пов'язану з ним інформацію.

Користувач додає новий сервер в систему. Вводить назву, IP адреса, логін і пароль сервера. Система перевіряє сервер на доступність. Викликає сценарій "Вказати місце зберігання логів" і додає сервер в список.

Користувач видаляє сервер з системи. Вибирає зі списку існуючий сервер і підтверджує видалення. Система перевіряє наявність сервера в системі і видаляє всю пов'язану з ним інформацію.

Користувач вводить адресу знаходження логів. Система перевіряє адресу на доступність і додає шлях в БД.

Користувач переглядає логи сервера і вибирає сортувати логи.

Система виводить меню вибору типу сортування в якому користувач вибирає сценарій для подальшого сортування.

Користувач вирішує впорядкувати логи за датою. Користувач вибирає впорядкувати за датою в контекстному меню. Відбувається запит на сортування даних за датою.

Користувач вирішує впорядкувати логи за рівнем логів. Користувач вибирає впорядкувати за рівнем в контекстному меню. Відбувається запит на сортування даних за рівнем.

Користувач вирішує впорядкувати логи по джерелу. Користувач вибирає джерело, в контекстному меню. Відбувається запит на сортування даних по джерелу. Користувач переглядає логи сервера і вибирає фільтрувати логи. Система виводить меню вибору правила фільтрації в якому користувач вибирає сценарій для подальшої фільтрації.

Користувач вирішує змінити місце або маску пошуку логів сервера. Користувач вибирає сервер, і активує функцію "Змінити настройки". Система відкриває форму зміни. І зберігає після підтвердження.

Користувач вирішує очистити логи. Користувач вибирає функцію "Очистити логи". Задає правило і підтверджує. Система очищає список і базу по заданому правилу.

Користувачеві потрібно знайти логи, які містять конкретну фразу. Користувач заповнює поле пошук і вибираємо "Пошук". Система виводить знайдені логи.

Адміністратору необхідно змінити налаштування програми. Адміністратор відкриває сторінку налаштувань де змінює необхідні дані. Система записує нові налаштування.

Таблиця 3.1 – Керувати користувачами – Додати користувача

Діючі лиця	Адміністратор, Система
Цілі	Адміністратор: Додати користувача до списку; Система: Додати користувача в базу.
Вхід	Унікальні логін і пошта, пароль.
Вихід	Список зареєстрованих користувачів в базі даних
Успішний сценарій: 0. Адміністратор в списку користувачів вибирає "Додати користувача". Система відкриває форму додавання користувача. 1. Адміністратор вводить логін, пароль, пошту. 2. Система перевіряє логін, пароль, пошту. 3. Система виводить повідомлення про успішне додаванні користувача	
Результат	Адміністратор додав нового користувача.
Розширення:	
2а	Логін вже існує в базі. Результат: Реєстрація нового користувача з обраним логіном неможлива. Система видає повідомлення ("Введений логін вже зареєстрований"). Повернення на етап 1.
2б	Введена пошта вже зайнята. Результат: Реєстрація нового користувача до обраної поштою неможлива. Система видає повідомлення ("Запроваджена пошта вже зареєстрована"). Повернення на етап 1.

Таблиця 3.2 – Керувати користувачами - Видалити користувача

Діючі лиця	Адміністратор, Система
Цілі	Адміністратор: Видалити користувача; Система: Видалити користувача з бази.
Вхід	Потрібно видалити користувача з системи.
Вихід	Список користувачів без видаленого.
Успішний сценарій: 0. Адміністратор в списку користувачів вибирає користувача і вибирає "Видалити користувача". Система виведе діалог підтвердження 1. Адміністратор підтверджує видалення. Система видаляє вибраного користувача і оновлює список.	

Продовження таблиці 3.2

Результат	Адміністратор видалив користувача
Розширення:	
1а	Користувач не існує в базі даних Результат: Відмова видалення користувача. Система видає повідомлення ("Користувач не існує в базі даних"). Повернення на етап 0.
1б	Адміністратор не підтвердив вилучення. Результат: Користувач не видалений. Повернення на етап 0.

Таблиця 3.3 – Змінити налаштування програми

Діючі лиця	Адміністратор, Система
Цілі	Адміністратор: Змінити налаштування програми; Система: Записати нові налаштування.
Вхід	Потрібно змінити налаштування програми.
Вихід	Нові налаштування програми.
Успішний сценарій: 0. Адміністратор заходить на сторінку загальних параметрів. Система виводить поточні налаштування. 1. Адміністратор змінює частоту опитування сервера, частоту перевірки активності і / або період зберігання логів. 2. Система записує нові значення.	
Результат	Установки програми змінені.
Розширення:	
0А	Немає доступу до БД. Система видає повідомлення ("База даних не доступна"). Результат: Повернення на головний екран.
2а	Перезаписати файл неможливо. Система виводить повідомлення "Файл з настройками неможливо переписати. Повернення на етап 0.

Таблиця 3.4 – Авторизація

Діючі лиця	Користувач, Система
Цілі	Користувач: авторизуватися в системі і почати працювати; Система: ідентифікувати користувача і його права.
Вхід	Логін і пароль користувача.
Вихід	Успішна авторизація.

Продовження таблиці 3.4

Успішний сценарій: 0. Користувач запускає систему. Система відкриває сесію користувача, пропонує ввести логін і пароль. 1. Користувач вводить логін і пароль. 2. Система перевіряє логін і пароль. 3. Система видає користувачеві повідомлення з приводу успішної авторизації ("Ви успішно увійшли в систему").	
Результат	Користувач успішно авторизований і може працювати з системою.
Розширення	
* а	Немає доступу до БД. Система видає повідомлення ("База даних не доступна"). Результат: користувач не може увійти.
1а	Користувач вибирає: «Нагадати пароль». Викликається сценарій «Нагадати пароль».
2а	Користувач з введеними логіном і паролем не найдений. Результат: відмова в авторизації. Система видає повідомлення ("Введені невірні дані"). Перехід на крок 2.

Таблиця 3.5 – Нагадати пароль

Діючі лиця	Користувач, Система
Цілі	Користувач: відновити пароль користувача; Система: змінити користувачеві пароль.
Вхід	Пошта забутого аккаунта.
Вихід	Аккаунт зі зміненим паролем.
Успішний сценарій: 0. Користувач натиснув "Нагадати пароль". Система відкриває вікно відновлення пароля, пропонує ввести логін. 1. Користувач вводить логін. 2. Система перевіряє логін і відправляє новий пароль через пошту. 3. Система видає користувачеві повідомлення з приводу успішну відправку нового пароля.	
Результат	Користувач успішно змінив пароль.
Розширення:	
* а	Немає доступу до БД. Система видає повідомлення ("База даних не доступна"). Результат: користувач не може змінити пароль.

Продовження таблиці 3.5

Розширення:	
2а	Користувач з введеними користувач не існує знайдений. Результат: відмова у відновленні пароля. Система видає повідомлення ("Користувач з введенням користувач не існує зареєстрований"). Перехід на крок 2.

Таблиця 3.6 – Налаштування облікового запису – Змінити пароль

Діючі лиця	Користувач, Система
Цілі	Користувач: відновити пароль користувача; Система: змінити користувачеві пароль.
Вхід	Поточний і новий пароль.
Вихід	Аккаунт зі зміненим паролем.
Успішний сценарій: 0. Користувач відкриває вікно "Налаштування облікового запису". Система відкриває сторінку з настройками. 1. Користувач уводить старий і новий пароль, повторює новий пароль. 2. Система перевіряє старий і новий пароль. 3. Система виводить повідомлення про успішну зміну.	
Результат	Користувач успішно змінив пароль і може продовжувати роботу
Розширення:	
2а	Старий пароль невірний. Результат: відмова в зміні пароля. Система видає повідомлення ("Поточний пароль введений не вірно"). Перехід на крок 1.
2б	Старий і новий пароль збігаються. Результат: відмова в зміні пароля. Система видає повідомлення ("Старий і новий пароль збігаються"). Перехід на крок 1.
2в	Поля нового пароля і повторення пароля не збігаються. Результат: відмова в зміні пароля. Система видає повідомлення ("Пароль повторений не вірно"). Перехід на крок 1.

Таблиця 3.7 – Керувати списком сервером – Додати сервер

Діючі лиця	Користувач, Система
Цілі	Користувач: Додати сервер в список; Система: Перевірити сервер і додати.

Продовження таблиці 3.7

Вхід	Назва, IP, порт, логін і пароль до сервера.
Вихід	Пул серверів з доданим сервером.
Успішний сценарій: 0. Користувач вибирає в списку серверів "Додати сервер". 1. Користувач вводить назву, IP, порт, логін і пароль до сервера і протокол підключення. 2. Система перевіряє доступність сервера. 3. Викликається сценарій "Вказати місце зберігання логів". 4. Система додає сервер в список.	
Результат	Система додає сервер в список.
Розширення:	
2а	Сервер не доступний. Результат: Сервер не може бути доданий. Система видає повідомлення ("Не вдалося підключитися до сервера"). Повернення на етап 1.
4а	У списку вже є сервер з такою назвою. Результат: Не можна додати сервер з таким ім'ям. Система видає повідомлення ("Введене назва вже зайнято"). Повернення на етап 1.

Таблиця 3.8 – Керувати списком сервером – Змінити настройки сервера

Діючі лиця	Користувач, Система
Цілі	Користувач: Змінити настройки сервера; Система: Змінити збережені настройки.
Вхід	Нові настройки сервера.
Вихід	Сервер з зміненими настройками.
Успішний сценарій: 0. Користувач вибирає в списку серверів сервер і активує "Змінити настройки логів". 1. Система отримує інформацію про сервер. 2. Викликається сценарій "Вказати місце зберігання логів". 3. Система змінює настройки.	
Результат	Система змінює настройки пошуку логів.
Розширення:	
1а	Доступ до бази даних відсутня. Результат: Перегляд логів неможливий. Система видає повідомлення ("База даних не доступна"). Повернення на етап 0.

Таблиця 3.9 – Керувати списком сервером – Видалити сервер

Діючі лиця	Користувач, Система
Цілі	Користувач: Видалити сервер зі списку; Система: Видалити сервер з бази.
Вхід	Необхідність видалити сервер зі свого пулу.
Вихід	Сервер видалений з пулу разом з усіма логами.
Успішний сценарій: 0. Користувач вибирає в списку сервер і вибирає "Видалити сервер". Система виведе діалог підтвердження. 1. Користувач підтверджує видалення. Система видаляє сервер зі списку і з бази.	
Результат	Сервер вилучений зі списку.
Розширення:	
1а	Сервер не існує в базі. Результат: Сервер не може бути видалений. Система видає повідомлення ("Сервер не існує в базі даних"). Повернення на етап 0.
1б	Використовувати не підтвердив вилучення. Результат: Сервер не видалений. Повернення на етап 0.

Таблиця 3.10 – Додати сервер – Вказати місце зберігання логів.

Діючі лиця	Користувач, Система
Цілі	Користувач: Вказати місце зберігання логів; Система: Додати в базу місце зберігання логів.
Вхід	Місцезнаходження папки з логами, маска для файлів-логів.
Вихід	Запис в базі даних про місце зберігання логів.
Успішний сценарій: 0. Користувач додає новий сервер. Система виводить вікно для введення адреси. 1. Користувач вводить адресу знаходження логів (адресний рядок). Система підтверджує доступність папки з логами. 2. Користувач задає маску файлів з логами. Система перевіряє чи існують такі файли. 3. Користувач підтверджує додавання логів за вказаним шляхом. Система додає шлях до логам в базу. 4. Перехід до пункту 5 прецеденту "Додати сервер".	
Результат	Користувач додав нове місце зберігання логів

Продовження таблиці 3.10

Розширення:	
1а	Користувач вводить неправильну адресу знаходження логів (адресний рядок). Система не підтверджує доступність папки за вказаним шляхом. Результат: Повідомлення про невірний адресу логів. Система видає повідомлення ("Введений шлях недоступний"). Повернення на етап 0.
2а	У папці немає логів з обраної маскою. Результат: Попередження що в даний момент в папці немає логів із заданим ім'ям. Система видає повідомлення ("За введеної масці в даний момент немає файлів.").
3а	Користувач не підтвердив додавання логів Результат: Повідомлення про помилку. Система видає повідомлення ("Операція скасована користувачем"). Повернення на етап 0.

Таблиця 3.11 – Подивитися логи сервера.

Діючі лиця	Користувач, Система
Цілі	Користувач: подивитися логи сервера; Система: вивести користувачеві логи сервера.
Вхід	Необхідність подивитися логи обраного сервера.
Вихід	Список всіх логів, з обраного сервера
Успішний сценарій: 0. Користувач вибирає сервер зі списку. Система виводить директорії сервера. 1. Користувач вибирає директорію для перегляду. 2. Система отримує запис логів з БД. 3. Система виводить логи користувачеві.	
Результат	Користувач бачить всі старі і нові надходять логи.
Розширення:	
1а	Доступ до бази даних відсутня. Результат: Перегляд логів неможливий. Система видає повідомлення ("База даних не доступна"). Повернення на етап 0.
1а	Доступ до папки неможливий. Результат: Перегляд логів неможливий. Система видає повідомлення ("Не вдалося відкрити директорію. Директорія видалена або недостатньо прав"). Повернення на етап 0.

Таблиця 3.12 – Подивитися логи сервера – Видалити логи.

Діючі лиця	Користувач, Система
Цілі	Користувач: Очистити історію логів; Система: Очистити базу даних і список від обраних логів.
Вхід	Необхідність видалити частину балок по заданому правилу.
Вихід	Список без віддалених логів.
Успішний сценарій: 0. Користувач переглядає логи і вибирає "Видалити логи". Система виводить форму вибору правила. 1. Користувач задає правило видалення логів. 2. Користувач підтверджує видалення. Система очищає список і базу даних.	
Результат	Список очищений від логів за обраним правилом.
Розширення:	
1а	Користувач закриває форму. Результат: Форма закривається, видалення не відбувається. Повернення на етап 0.

Таблиця 3.13 – Подивитися логи сервера – Пошук.

Діючі лиця	Користувач, Система
Цілі	Користувач: Знайти лог з введеної фразою; Система: Вивести користувачеві знайдені логи.
Вхід	Необхідність знайти логи з введеної фрази.
Вихід	Список знайдених логів.
Успішний сценарій: 0. Користувач переглядає логи і вирішує виконати пошук, він заповнює поле пошуку і натискає "Пошук". Система починає пошук. 1. Система перевіряє поле і виводить список логів.	
Результат	Виводяться знайдені логи.
Розширення:	
1а	Поле пошуку порожньо. Результат: Пошук не проводиться. Система видає повідомлення ("Поле пошуку не заповнено"). Повернення на етап 0.

Таблиця 3.14 – Подивитися логи сервера – подивитися інформацію про балці.

Діючі лиця	Користувач, Система
Цілі	Користувач: подивитися інформацію про конкретний балці; Система: вивести користувачеві обраний лог сервера.
Вхід	Необхідність подивитися інформацію про конкретний балці.
Вихід	Інформація про конкретний балці.

Продовження таблиці 3.14

Успішний сценарій: 0. Користувач переглядає логи і вибирає один з логів. Система виводить контекстне меню. 1. Користувач активує "Перегляд лога". Система виводить інформацію про балці.	
Результат	Користувач бачить обраний лог
Розширення:	
1а	Лог не доступний Результат: Перегляд обраний лог неможливо. Повернення на етап "Перегляд логів сервера". Система видає повідомлення ("Інформація про обраний балці недоступна").

Таблиця 3.15 – Переглянути логи сервера – Сортувати.

Діючі лиця	Користувач, Система
Цілі	Користувач: отримати відсортовані логи; Система: видати з бази відсортовані логи.
Вхід	список логів
Вихід	Список з відсортованими логами.
Успішний сценарій: 0. Користувач переглядає логи сервера і вибирає сортувати логи. Система виводить контекстне меню вибору типу сортування. 1. а. Пользователь вибирає сортування за датою. Перехід до прецеденту "Сортування за датою". 1. б. Користувач вибирає сортування за рівнем. Перехід до прецеденту "Сортування за рівнем". 1. ст. Користувач вибирає сортування за джерелом. Перехід до прецеденту "Сортування за джерелом". 2. Система відображає відсортовані логи.	
Результат	Користувач отримав відсортовані логи
Розширення:	
1.х.1	Користувач не вибрав сортування. Перехід до прецеденту "Переглянути логи сервера". Результат: Список НЕ відсортовані.

Таблиця 3.16 – Сортувати – Сортування за датою.

Діючі лиця	система
Цілі	Система: отримала відсортовані логи по даті;

Продовження таблиці 3.16

Вхід	Список логів.
Вихід	Користувач отримує список з відсортованими логами по правилу "Сортувати за датою".
Успішний сценарій: 0. Викликаний прецедент сортування логів за датою. 1. Система звертається до бази і запитує збережену процедуру для сортування логів. База даних виконує збережену процедуру. 2. База даних повертає результат. Система отримала результат.	
Результат	Система отримала відсортовані логи
Розширення:	
1а	База даних перевантажена. Результат: Система в очікуванні отримання відповіді на запит. Система не отримує відсортовані логи. Простоювання на прецеденті "Сортування за датою".

Таблиця 3.17 – Сортувати – Сортування за рівнем.

Діючі лиця	система
Цілі	Система: отримати відсортовані логи за рівнем;
Вхід	Список логів.
Вихід	Користувач отримує список з відсортованими логами по правилу "Сортувати за рівнем".
Успішний сценарій: 0. Викликаний прецедент сортування логів за рівнем. 1. Система звертається до бази і запитує збережену процедуру для сортування логів. База даних виконує збережену процедуру. 2. База даних повертає результат. Система отримала результат.	
Результат	Система отримала відсортовані логи
Розширення:	
1а	База даних перевантажена. Результат: Система в очікуванні отримання відповіді на запит. Система не отримує відсортовані логи. Простоювання на прецеденті "Сортування за рівнем".

Таблиця 3.18– Сортувати – Сортування за джерелом.

Діючі лиця	система
Цілі	Система: отримати відсортовані логи по джерелу;
Вхід	Список логів.

Продовження таблиці 3.18

Вихід	Користувач отримує список з відсортованими логами по правилу "Сортувати за джерелом".
Успішний сценарій: 0. Викликаний прецедент сортування логів по джерелу. 1. Система звертається до бази і запитує збережену процедуру для сортування логів. База даних виконує збережену процедуру. 2. База даних повертає результат. Система отримала результат.	
Результат	Система отримала відсортовані логи
Розширення:	
1а	База даних перевантажена. Результат: Система в очікуванні отримання відповіді на запит. Система не отримує відсортовані логи. Простоювання на прецеденті "Сортування за джерелом".

Таблиця 3.19– Переглянути логи сервера – Фільтрувати.

Діючі лиця	Користувач, Система
Цілі	Користувач: отримати відфільтровані логи; Система: показати користувачеві відфільтровані логи.
Вхід	Список логів.
Вихід	Список з відфільтрованими логами.
Успішний сценарій: 0. Користувач переглядає логи сервера і вибирає фільтрувати логи. Система виводить меню вибору типу фільтрації. 1. Користувач задає правило фільтрації логів. (Filtr.html) 2. Система відображає відфільтровані логи.	
Результат	Користувач отримав відфільтровані логи
Розширення:	
1а	Користувач залишив поза увагою жодного правила фільтрації. Система виводить повідомлення: "Не вибрані правила фільтрації". Результат: Список НЕ відфільтровані. Повернення на етап 0.
1б	Користувач закрив меню фільтрації. Перехід до прецеденту "Переглянути логи сервера" Результат: Список НЕ відфільтровані.

Таблиця 3.20 – Функціональні вимоги

Домен	Опис
LOGIN01	Користувач заходить в систему вводить логін і пароль. Система перевіряє і видає повідомлення про успішну авторизації або про помилку введених даних.
LOGIN02	Користувач забуває свій пароль, заходить у вікно відновлення пароля і вводить свій логін. Система перевіряє логін і якщо він вірний висилає повідомлення на пошту з паролем, якщо немає - видає повідомлення про невірно введеному паролі і відмовляє користувачеві в можливості відновлення пароля.
USER-SETTINGS01	Користувач вирішує змінити пароль. Заходить у вікно зміни пароля і вводить старий і новий пароль. Система перевіряє старий пароль на правильність введення та новий на розбіжність зі старим. І при правильно введених всіх даних змінює пароль в системі.
VIEW-LOG01	Користувач запитує перегляд логів. Система отримує запис з інформацією про сервер з БД, підключається до сервера, отримує запис логів від сервера. Система заповнює БД логами і виводить логи користувачеві.
VIEW-LOG02	Користувач вибирає один з логів і запитує інформацію про нього. Система видає всю доступну інформацію, якщо така є.
ADMIN-MANAGE01	Адміністратор додає користувача в систему. Вводить логін пароль і пошту. Якщо логін і пошта не існують в БД, то система успішно зберігає нового користувача.
ADMIN-MANAGE02	Адміністратор видаляє користувача з системи. Вибирає зі списку існуючого користувача і підтверджує видалення. Система перевіряє наявність користувача в системі і видаляє всю пов'язану з ним інформацію.
USER-SERVERS01	Користувач додає новий сервер в систему. Вводить назву, IP адреса, логін і пароль сервера. Система перевіряє сервер на доступність. Викликає сценарій "Вказати місце зберігання логів" і додає сервер в список.
USER-SERVERS02	Користувач видаляє сервер з системи. Вибирає зі списку існуючий сервер і підтверджує видалення. Система перевіряє наявність сервера в системі і видаляє всю пов'язану з ним інформацію.
USER-SERVERS-ADD01	Користувач вводить адресу знаходження логів. Система перевіряє адресу на доступність і додає шлях в БД.

Продовження таблиці 3.20

SORT01	Користувач переглядає логи сервера і вибирає сортувати логи. Система виводить меню вибору типу сортування в якому користувач вибирає сценарій для подальшого сортування.
SORT02	Користувач вирішує впорядкувати логи за датою. Користувач вибирає впорядкувати за датою в контекстному меню. Відбувається запит на сортування даних за датою.
SORT03	Користувач вирішує впорядкувати логи за рівнем логів. Користувач вибирає впорядкувати за рівнем в контекстному меню. Відбувається запит на сортування даних за рівнем.
SORT04	Користувач вирішує впорядкувати логи по джерелу. Користувач вибирає джерело, в контекстному меню. Відбувається запит на сортування даних по джерелу.
FILTR01	Користувач переглядає логи сервера і вибирає фільтрувати логи. Система виводить меню вибору правила фільтрації в якому користувач вибирає сценарій для подальшої фільтрації.
USER-SERVER S03	Користувач вирішує змінити місце або маску пошуку логів сервера. Користувач вибирає сервер, і активує функцію "Змінити настройки". Система відкриває форму зміни. І зберігає після підтвердження.
VIEW-LOG03	Користувач вирішує очистити логи. Користувач вибирає функцію "Очистити логи". Задає правило і підтверджує. Система очищає список і базу по заданому правилу.
VIEW-LOG04	Користувачеві потрібно знайти логи містять конкретну фразу. Користувач заповнює поле пошук і вибираємо "Пошук". Система виводить знайдені логи.
ADMIN-SETTING S01	Адміністратору необхідно змінити налаштування програми. Адміністратор відкриває сторінку налаштувань де змінює необхідні дані. Система записує нові настройки.

3.2 Нефункціональні вимоги

Далі наведено нефункціональні вимоги програмного продукту

А. Надійність

1. Якщо адмін додає нового користувача, то в базі від зберігається не менше ніж в 99% випадків.

2. Якщо адмін видаляє користувача, то з бази він видаляється не менше ніж в 99% випадків.

3. Якщо користувач вводить вірні дані і намагається авторизуватися, то у нього це виходить не менше ніж в 99% випадків.

4. Якщо користувач забуває свій пароль, то система нагадує його йому не менше ніж в 98% випадків.

5. Якщо користувач змінює свій пароль, то система змінює цей пароль не менше ніж в 98% випадків.

6. Якщо користувач додає новий сервер, то система видає йому вікно для введення місця зберігання логів не менше ніж в 99% випадків.

7. Якщо користувач змінює настройки сервера, то система зберігає ці настройки в не менше ніж 98% випадків.

8. Якщо користувач видаляє сервер, то система видаляє його з бази не менше ніж в 99% випадків.

9. Якщо користувач вказує місце зберігання логів, то система додає це місце не менше ніж в 98% випадків.

10. Якщо користувач переглядає логи сервера, то система видає йому список логів не менше ніж в 97% випадків. (Імовірність втрати 1 або більше логів)

11. Якщо користувач видаляє логи сервера, то система видаляє всі дані з бази не менше ніж в 98% випадків. (Імовірність видалити не всі пов'язані дані)

12. Якщо користувач переглядає інформацію про балці, то система видає йому цю інформацію не менше ніж в 99% випадків.

13. Якщо користувач сортує логи, то система видає відсортовані логи не менше ніж в 98% випадків.

14. Якщо користувач фільтрує логи, то система видає відфільтровані логи не менше ніж в 98% випадків.

15. Якщо користувач шукає логи, то система видає йому знайдені логи не менше ніж в 99% випадків.

Б. Продуктивність

1. Якщо адмін додає нового користувача, то в базі від зберігається не більше ніж за 1 секунду.

2. Якщо адмін видаляє користувача, то з бази він видаляється не більше ніж за 0,5 секунди.

3. Якщо користувач вводить вірні дані і намагається авторизуватися, то у нього це виходить не більше ніж за 1 секунду.

4. Якщо користувач забуває свій пароль, то система нагадує його йому не більше ніж за 1 хвилину.

5. Якщо користувач змінює свій пароль, то система змінює цей пароль не більше ніж за 1 секунду.

6. Якщо користувач додає новий сервер, то система видає йому вікно для введення місця зберігання логів не більше ніж за 1 секунду.

7. Якщо користувач змінює настройки сервера, то система зберігає ці настройки не більше ніж за 1 секунду.

8. Якщо користувач видаляє сервер, то система видаляє його з бази не більше ніж за 3 секунди.

9. Якщо користувач вказує місце зберігання логів, то система перевіряє і додає це місце не більше ніж за 7 секунд.

10. Якщо користувач переглядає логи сервера, то система видає йому список логів не більше ніж за 2 секунди.

11. Якщо користувач видаляє логи сервера, то система видаляє всі дані з бази не більше ніж за 2 секунди.

12. Якщо користувач переглядає інформацію про балці, то система видає йому цю інформацію не більше ніж за 0,5 секунди.

13. Якщо користувач сортує логи, то система видає відсортовані логи не більше ніж за 2 секунди.

14. Якщо користувач фільтрує логи, то система видає відфільтровані логи не більше ніж за 2 секунди.

15. Якщо користувач шукає логи, то система видає йому знайдені логи не більше ніж за 1 секунду.

В. Безпека

1. Якщо зловмисник спробує змінити системні дані, то система зареєструє зміни в контрольному журналі не менше в 98% випадків.

2. Якщо дані користувачів спробують зіпсувати сторонні зловмисники, у них це не вийде більш ніж в 99% випадків.

3. Якщо користувач спробує зламати систему, у нього це не вийде більш ніж в 99% випадків.

4. Якщо користувач спробує отримати інформацію про інших користувачів програмного продукту, у нього це не вийде більш ніж в 99% випадків.

Г. Юзабіліті

1. Якщо користувач вирішує додати логів розміром не більше 1Гб, то йому це вдається не більше ніж за 1,5 хвилини.

2. Якщо користувач вирішує додати новий сервер з кількістю даних більш 1Гб але менш 3 Гб, то у нього це виходить не більше ніж за 6 хвилин.

3. Якщо користувач вирішує додати 100 файлів розміром 25мб, то йому це вдається не більше ніж за 3,5 хвилини.

4. Якщо користувач вирішує додати 500 файлів розміром 25мб, то йому це вдається не більше ніж за 15 хвилин.

4 АРХІТЕКТУРА СИСТЕМИ

4.1 Послідовність дій користувача при першому використанні програмного продукту

Далі буде наведено послідовність дій користувача при першому використанні даного програмного продукту, та у подальшому коли йому необхідно додати новий сервер. Результати дій користувача наведено у текстовому описі, за основним сценарієм роботи програмного продукту, та наведена діаграма послідовності для цього сценарію на рис. 4.1.

Користувач авторизується у системі. (Вводить логін та пароль, та натискає кнопку увійти.)

Система зчитує дані, перевіряє їх та допускає користувача у програму.

Користувач додає новий сервер та вмикає його. Натискає кнопку додати новий сервер. Вводить адресу серверу та порт, логін, пароль для авторизації, та протокол за яким необхідно підключатися. Натискає кнопку активації.

Система збирає усі дані, перевіряє підключення до серверу, та показує, що підключення можливе. Сервер додається до опитувань на активність, та останній доступ користувача.

Користувач входить до серверу.

Користувач додає путь до директорії та активує її. Натискає кнопку додати нову директорію. Вводить путь до директорії, та натискає кнопку для активації директорії.

Система зчитує директорію, намагається отримати туди доступ, та відображає можливість доступу до директорії. Директорія додається до опитувань на існування, та останній доступ користувача.

Користувач обирає зі списку існуючих файлів необхідний та додає його. (Натискає кнопку додати новий файл. Відмічає у списку необхідний файл. Натискає кнопку додати.)

Система додає необхідний файл. Файли логів додаються до опитувань на існування, та збираються усі необхідні логи, запам’ятовуючи останній зчитаний рядок.

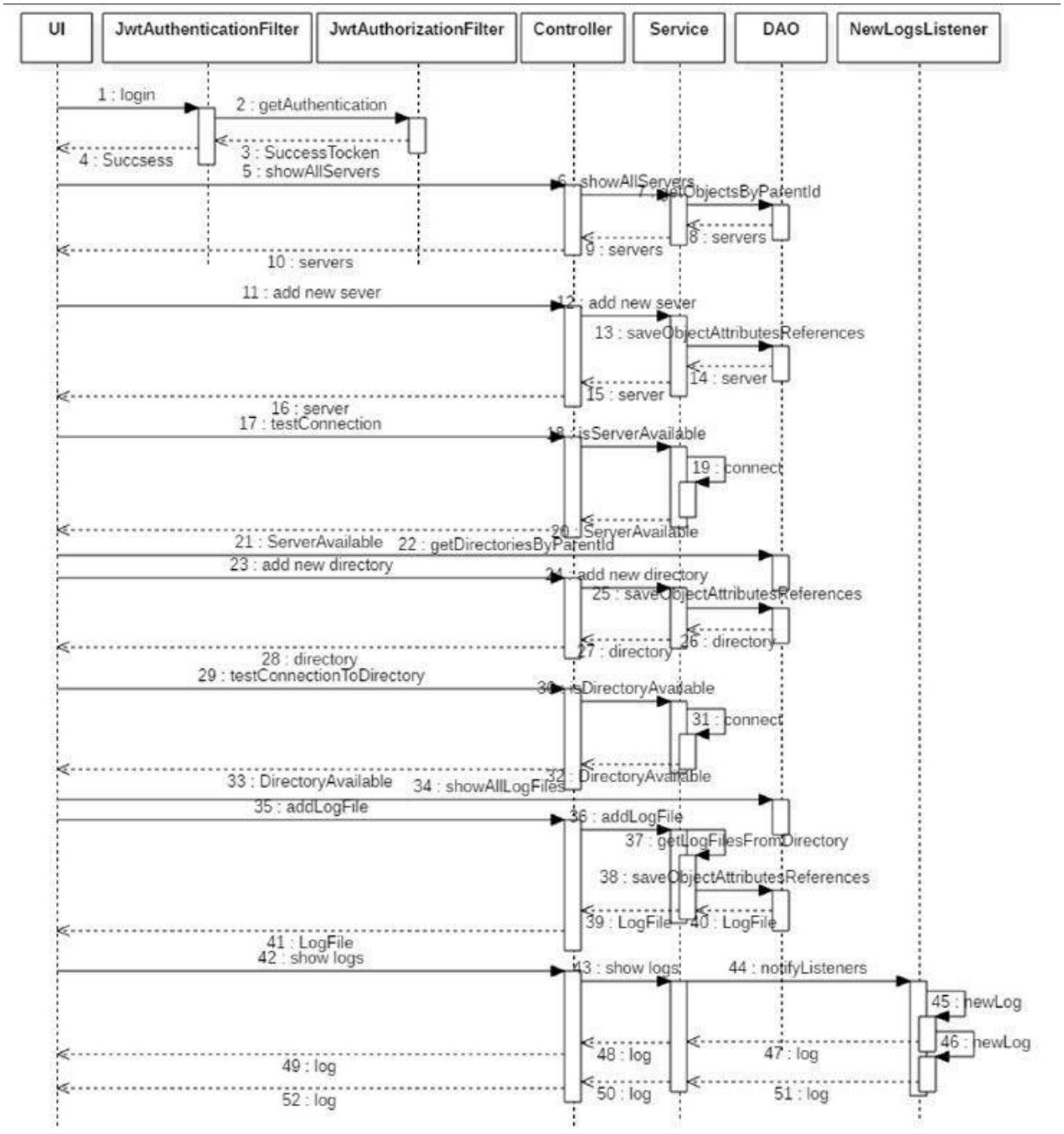

Рисунок 4.1 – Діаграма послідовності – Реакція системи на дії користувача при першому її використанні

Користувач переходить до сторінки логів онлайн. Натискає на кнопку перегляду логів онлайн.

Система відображає логи з усіх файлів директорії. (Збираються логи, які були щойно додані до файлів з логами.)

4.2 Послідовність дій користувача при найчастішому використанні програмного продукту

На цьому етапі наведено та описано реакцію системи на дії користувача при основних її використаннях (рис. 4.2). Дана послідовність дій полягає у відображені логів з усіх файлів директорії.

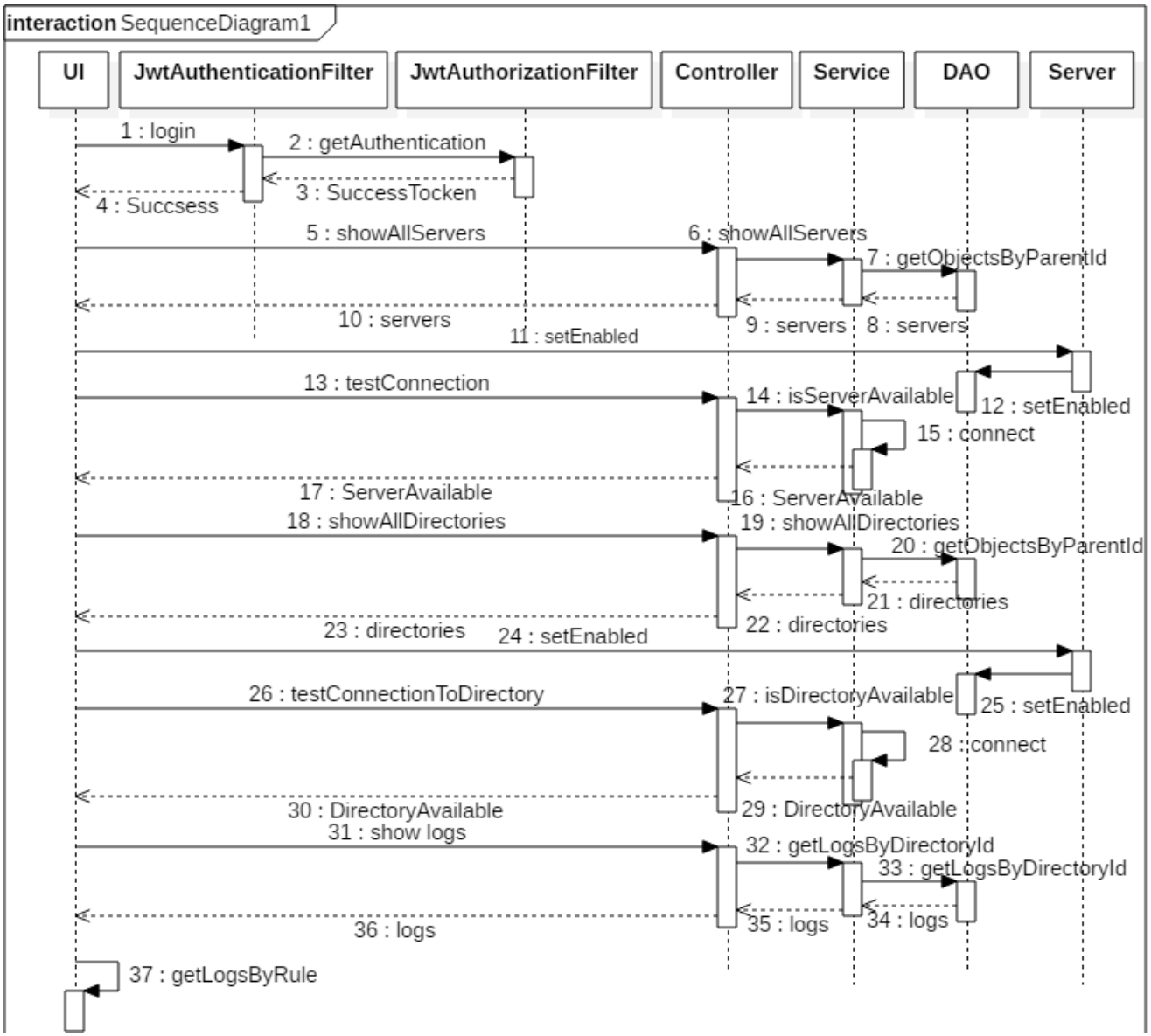

Рисунок 4.2 – Діаграма послідовності – Реакція системи на дії користувача при основних її використаннях

Логи сортуються, та фільтруються за трьома критеріями:

1. Дата, за яких період їх необхідно відображати логи, або вони відсортовані за цим критерієм.

2. Рівень логів, за якими необхідно відображати логи, або вони відсортовані за цим критерієм.

3. Текст, яких зустрічається у цих логах.

4.3 Послідовність дій користувача при рідкісному використанні програмного продукту

Наступним етапом наведено реакцію системи на дії користувача при рідкісному використанні, що дозволяють користувачеві з комфортом користуватися даним програмним продуктом.

1. Зміна паролю

Зміна паролю (рис. 4.3) відбувається після автентифікації.

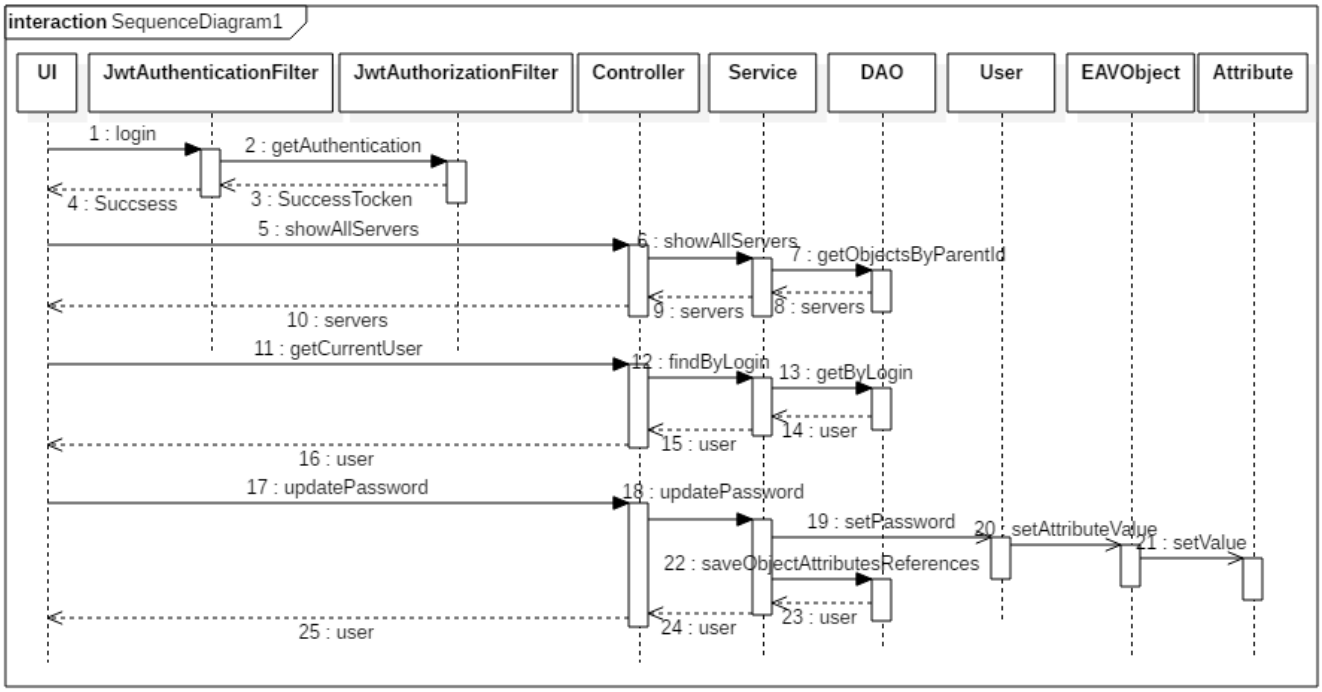

Рисунок 4.3 – Діаграма послідовності – Реакція системи на дії користувача підчас зміни паролю

Наступним етапом відображаються усі сервера (бо це основний сценарій даного програмного продукту). Далі переходимо до сторінки налаштувань, де можна побачити поштову адресу, логін та роль даного користувача. Далі необхідно ввести старий пароль, новий, та повторити його. Проходить встановлення нового паролю, та дані записуються у базу.

2. Відновлення паролю за логіном

Для відновлення паролю (рис. 4.4) необхідно перейти на спеціально відведену сторінку, та зазначити там свій логін від якого було втрачено пароль.

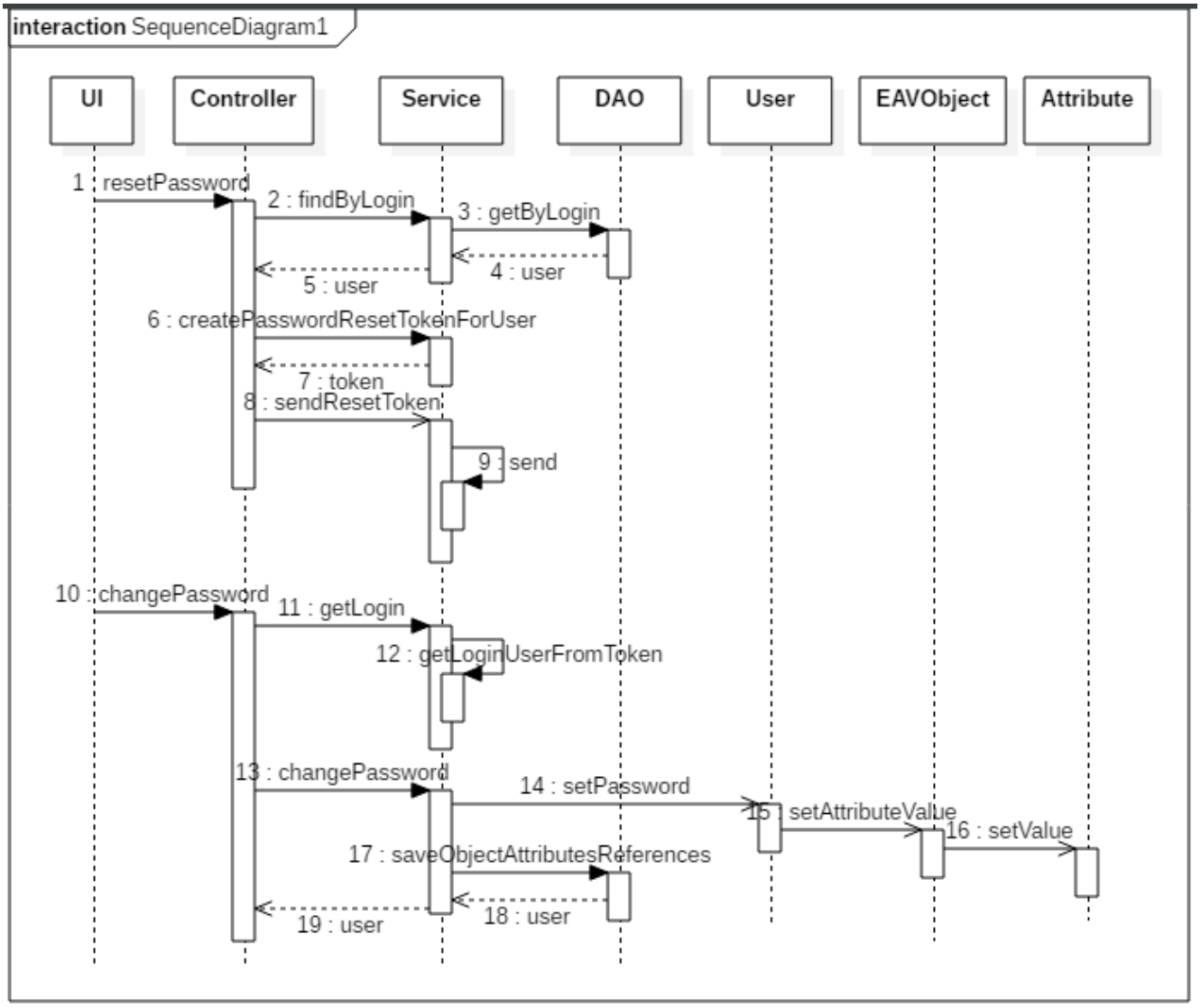

Рисунок 4.4 – Діаграма послідовності – Реакція системи на дії користувача підчас відновлення паролю

Програма виконує пошук користувача за логіном. Наступним етапом створюється спеціальний токін для відновлення паролю, та відсилається письмо з адресою, куди необхідно перейти для встановлення нового паролю.

Користувач переходить на надіслану адресу, дані з токіну збираються. Користувач встановлює новий пароль та повторює його. Дані зберігаються у базі даних.

4.4 Інтерфейс користувача

На рис. 4.5 – 4.9 зображено прототип інтерфейсу користувача, основних його сторінок. На цьому етапі для проектування інтерфейсу користувача вибраного модуля були використані спеціальний інструмент для UX/UI проектування - Wireframe. Створені рисунки прототипів можуть досить сильно відрізнятися від кінцевого результату, який буде відображено нижче. Так на рис. 4.5 можна побачити прототип початкової сторінки програмного продукту, де користувач може побачити назву додатку, та для чого створено цей додаток.

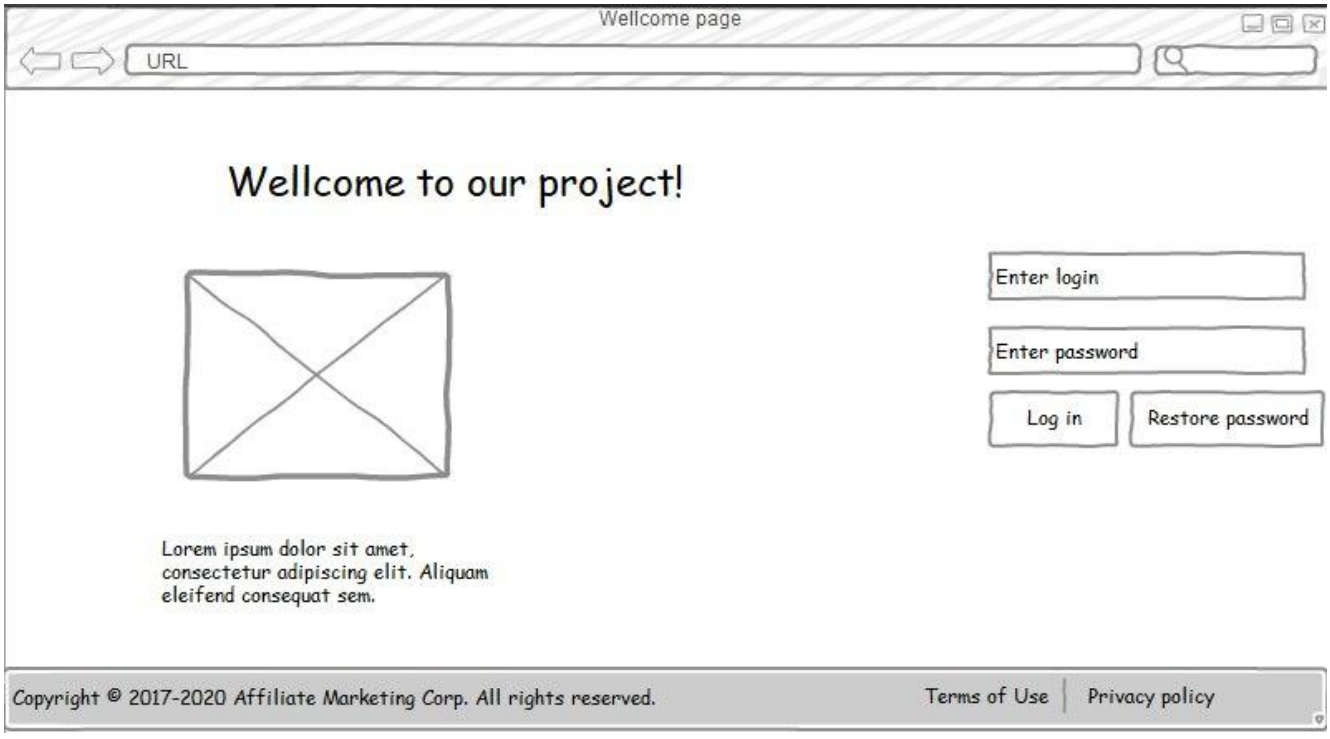

Рисунок 4.5 – Прототип інтерфейсу – початкова сторінка програмного продукту

Ще, на цій сторінці можна побачити, що є можливість ввести логін та пароль, та натиснути дві кнопки. Одна з яких дозволяє увійти в даний проект, а інша відновити існуючий пароль.

Рис. 4.6 – відображає список серверів, їх налаштування. Також на цій сторінці є можливість додавання нового серверу, кнопка переходу до панелі адміністратора, також дана сторінка має можливість додавати місце збереження логів, та випадаючий список. У останньому можна перейти до налаштувань аккаунту, зберегти всю інформації о серверах, перейти до сторінки для залишення відгуків, а також можливість виходу з додатку.

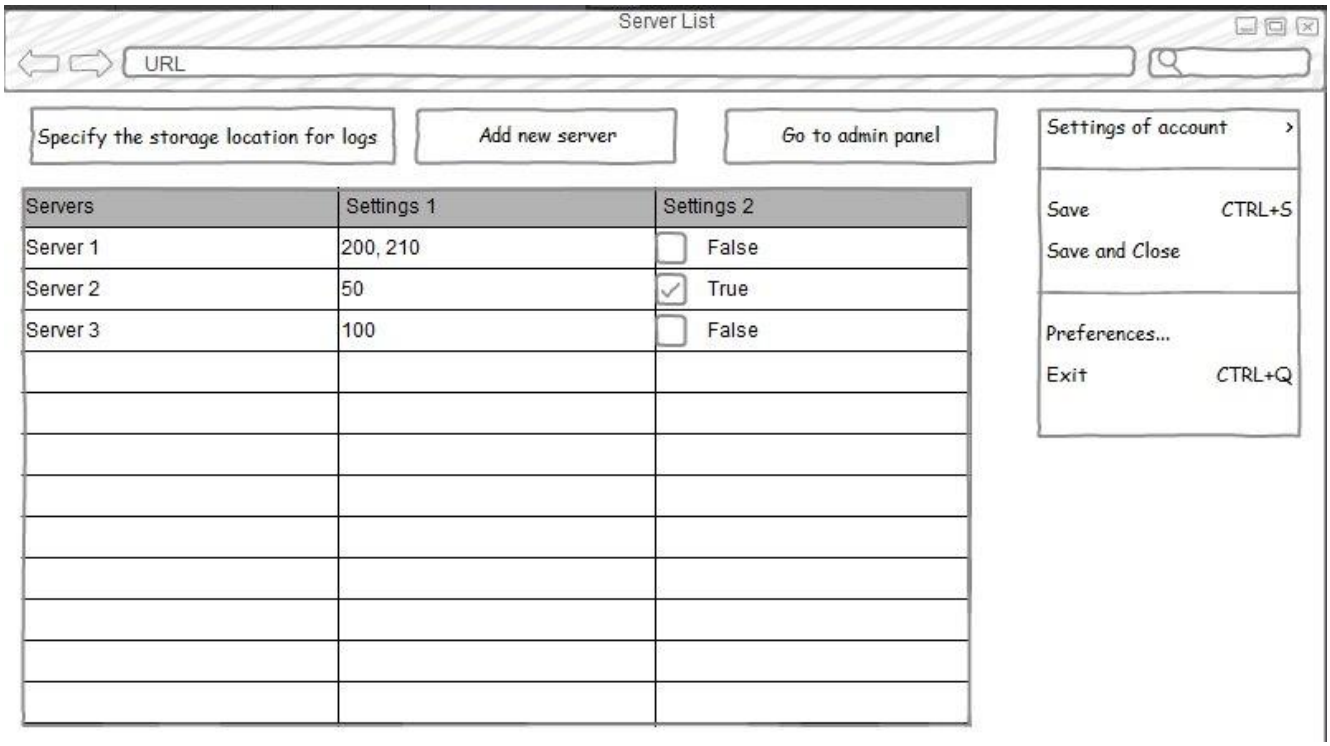

Рисунок 4.6 – Прототип інтерфейсу – сторінка списку серверів

Рис. 4.7 – відображає адмінастративну панель, на якій користувач – адміністратор має можливість переглянути усіх існуючих користувачів, зареєструвати нового (адміністратора, чи звичайного користувача), також змінити період зберігання логів, та період опитувань серверів.

Рис. 4.8 – дуже схож з рис. 4.6, основна відмінність у тому, що тут відображаються папки з серверів.

Рис. 4.9 – відображає сторінку перегляду логів.

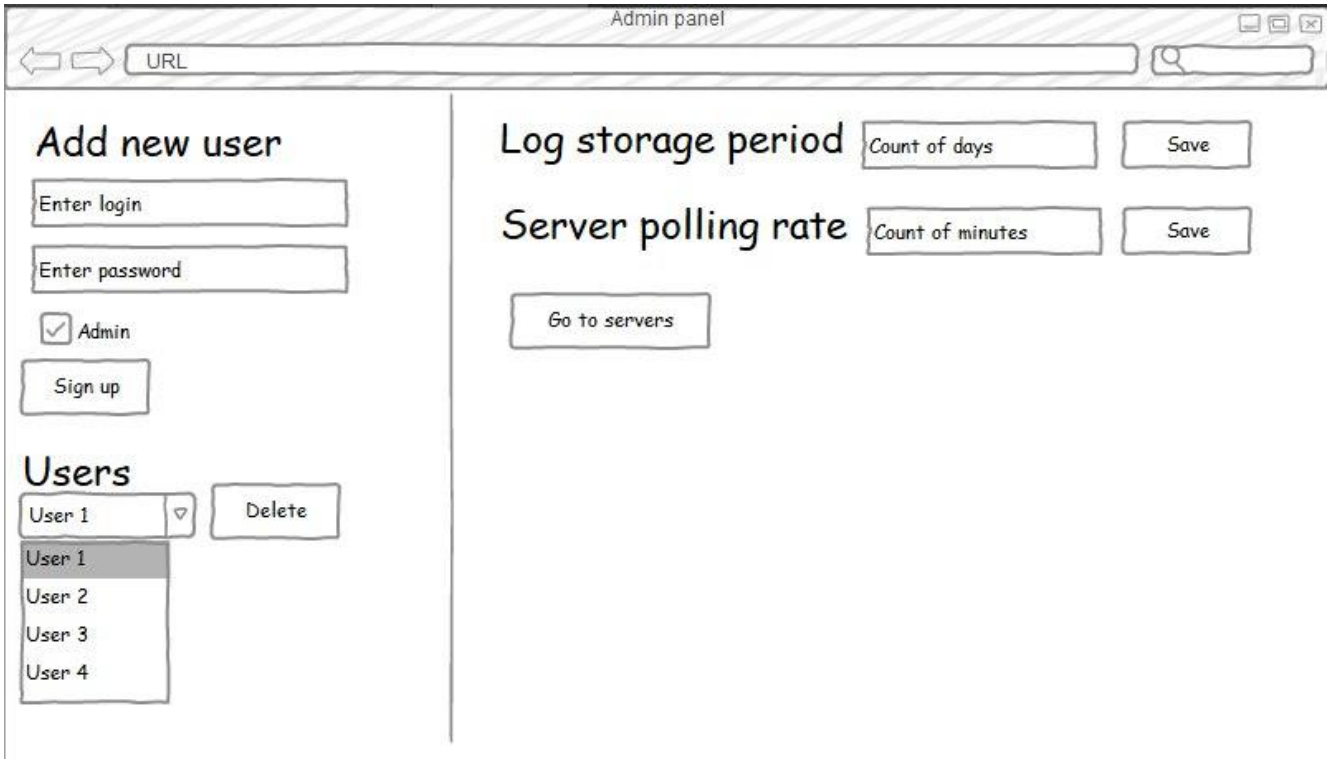

Рисунок 4.7 – Прототип інтерфейсу – сторінка панель адміністратора

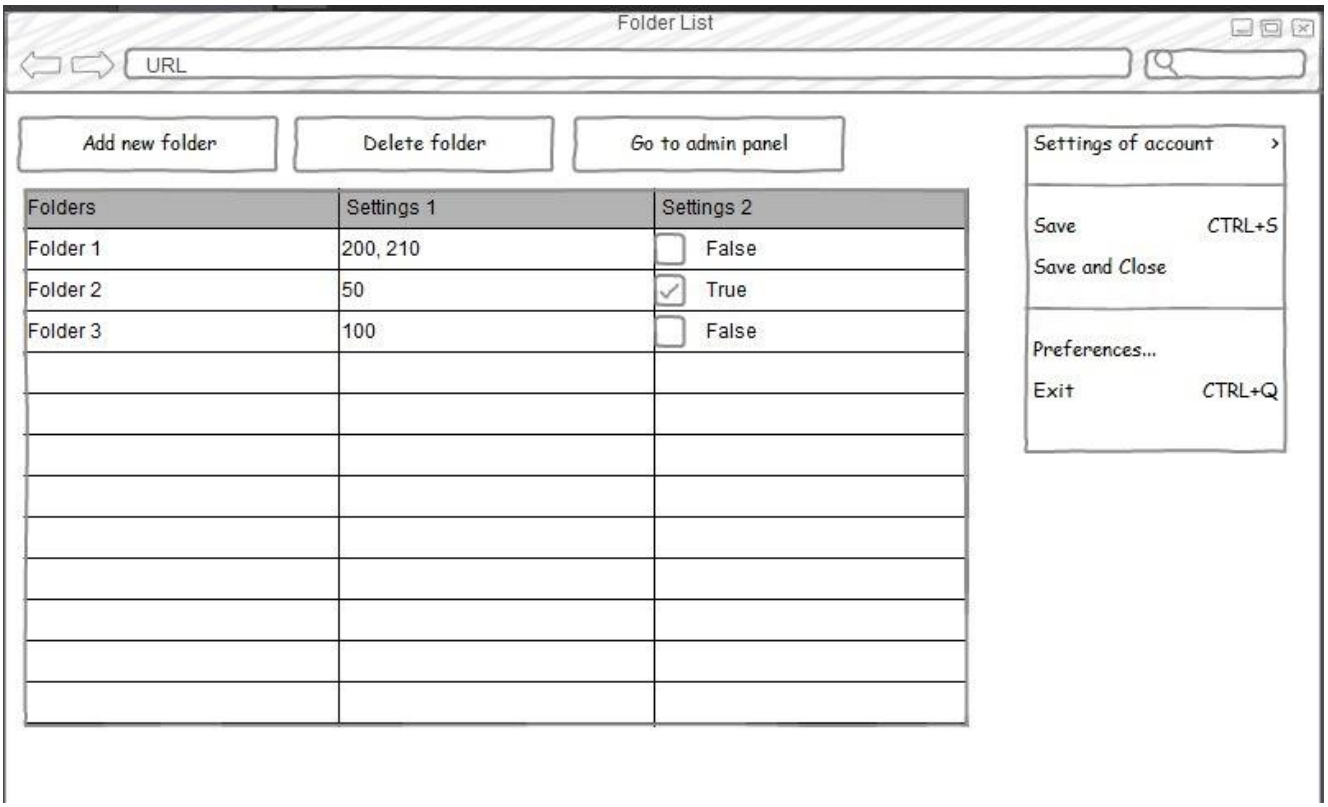

Рисунок 4.8 – Прототип інтерфейсу – сторінка списку папок

Окрім стандартного функціоналу, тут є можливість виконувати пошук за ключовими словами у логах. Сотрувати, та фільтрувати логи за датою, та рівнем.

Рисунок 4.9 – Прототип інтерфейсу – сторінка перегляду логів

Наступним етапом буде наведено кінцевий результат тих самих сторінок після програмної реалізації продукту. Разом з цим буде зазначено лише відмінності між прототипом та кінцевим результатом. Усі скріншоти програмного продукту можна побачити на рис. 4.10 – 4.16.

Рис. 4.10 – на відміну від прототипу в оригіналі відсутній опис програмного продукту, що на мою думку виглядало досить дивно. Дуже мала кількість людей читає, та цікавиться даною інформацією. Саме тому, щоб виглядало лаконічніше залишилася лише назва додатку, та форма авторизації.

Рис. 4.11 – у даному випадку було додано кнопку оновлення, яка перевіряє чи змінився статус серверів. Також тут наглядно видно кнопки активації серверу, переходу до папок обраного серверу, перегляду (зміни налаштувань), та видаленню серверу зі списку.

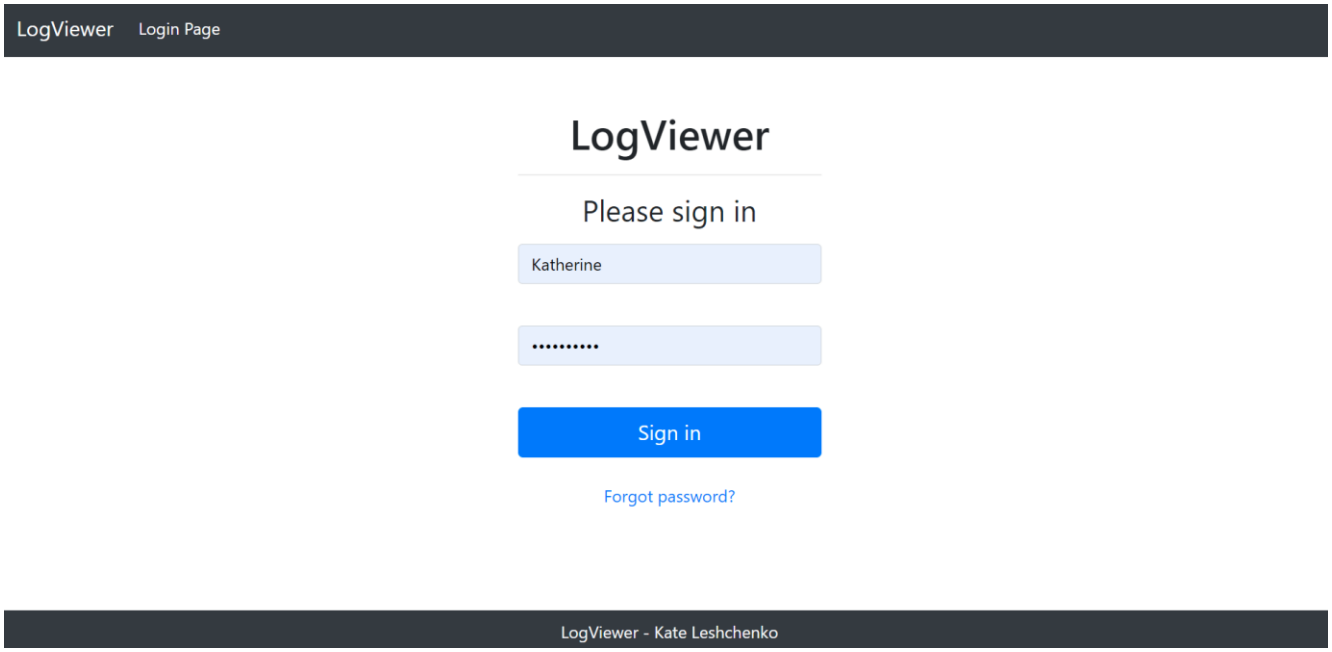

Рисунок 4.10 – Кінцевий вигляд продукту – початкова сторінка програмного продукту

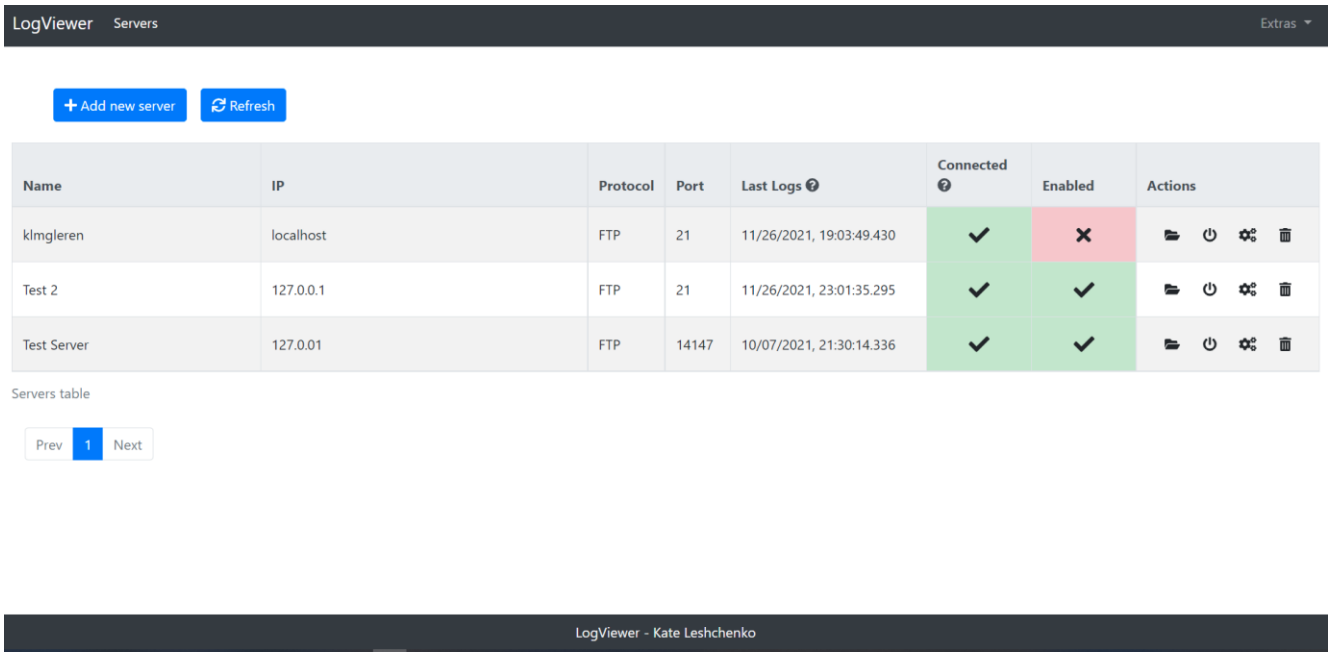

Рисунок 4.11 – Кінцевий вигляд продукту – сторінка списку серверів

Ще трохи змінився вигляд випадаючого списку. Зараз там є можливість перейти до зміни налаштувань аккаунту, вийти з даного програмного продукту, а також у адміністратора є можливість перейти на сторінку з користувачами, та глобальними налаштуваннями окремо.

Рис. 4.12 – панель адміністратора було вирішено поділити на дві сторінки перша з яких відображає список користувачів. На цій сторінці відображається логін, поштова адреса та роль. Також на цій сторінці можна додати, та видалити користувачів. За необхідністю можна змінити роль зі звичайного користувача на адміністратора.

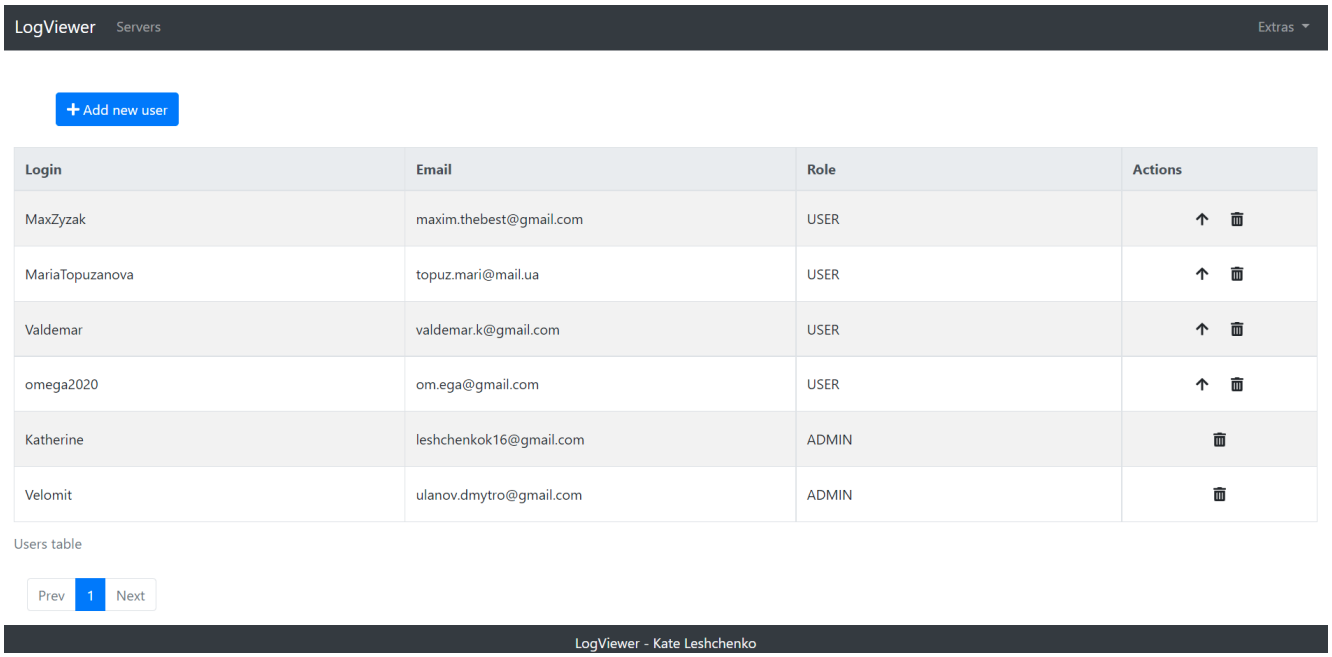

Рисунок 4.12 – Кінцевий вигляд продукту – сторінка списку користувачів (для адміністратора)

Рис. 4.13 – інша панель адміністратора, яка дозволяє змінювати данні оточення. Перше значення дозволяє змінити період активності вимкнених серверів, тобто час через який вони проходять опитування, а потім видаляються. Друге – час через який перевіряється зміна у файлах з логами. Третє – період через який логи будуть вважатися застарілими та видаляться з серверу. Четверте – час через який перевіряється чи постачалися на ці сервери логи. П’яте – час через який перевіряється чи постачалися у ці директорії логи.

Рис. 4.14 – сторінка на якій можна побачити інформацію про аккаунт, з якого було виконано вхід до системи, а також є можливість змінити пароль або якщо було забуте пароль то відновити його. Також можна видалити аккаунт з якого було виконано вхід.

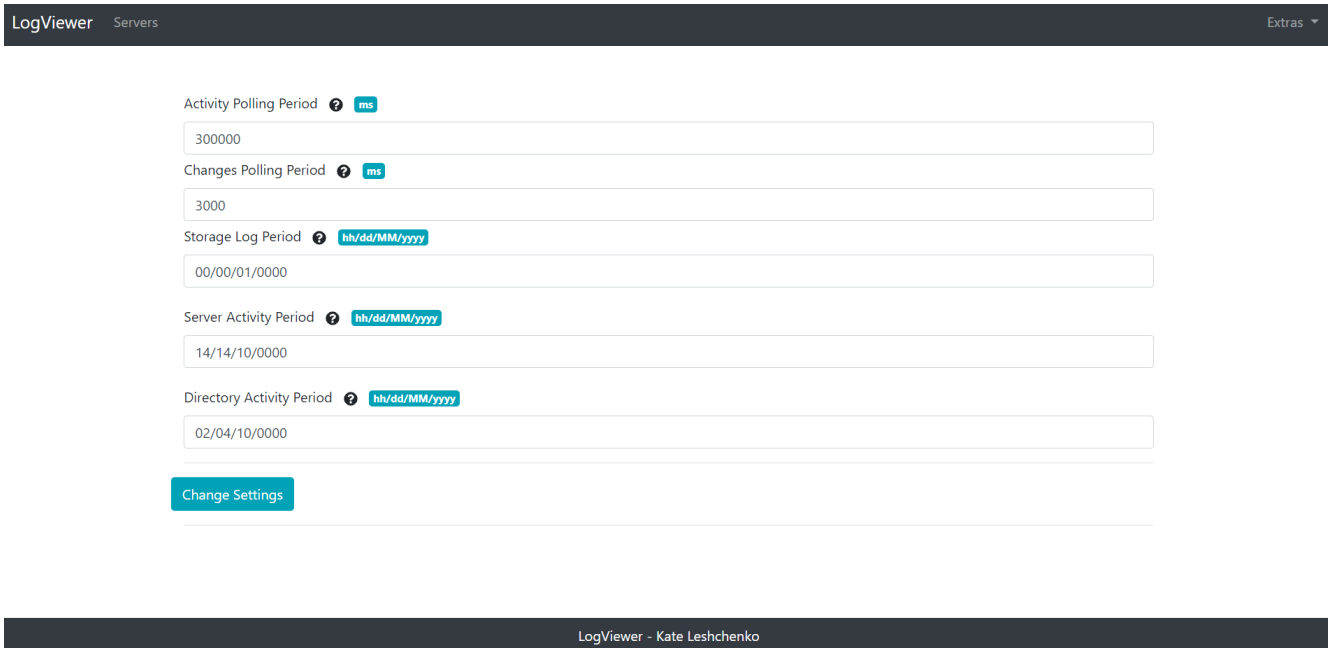

Рисунок 4.13 – Кінцевий вигляд продукту – сторінка списку глобальних налаштувань (для адміністратора)

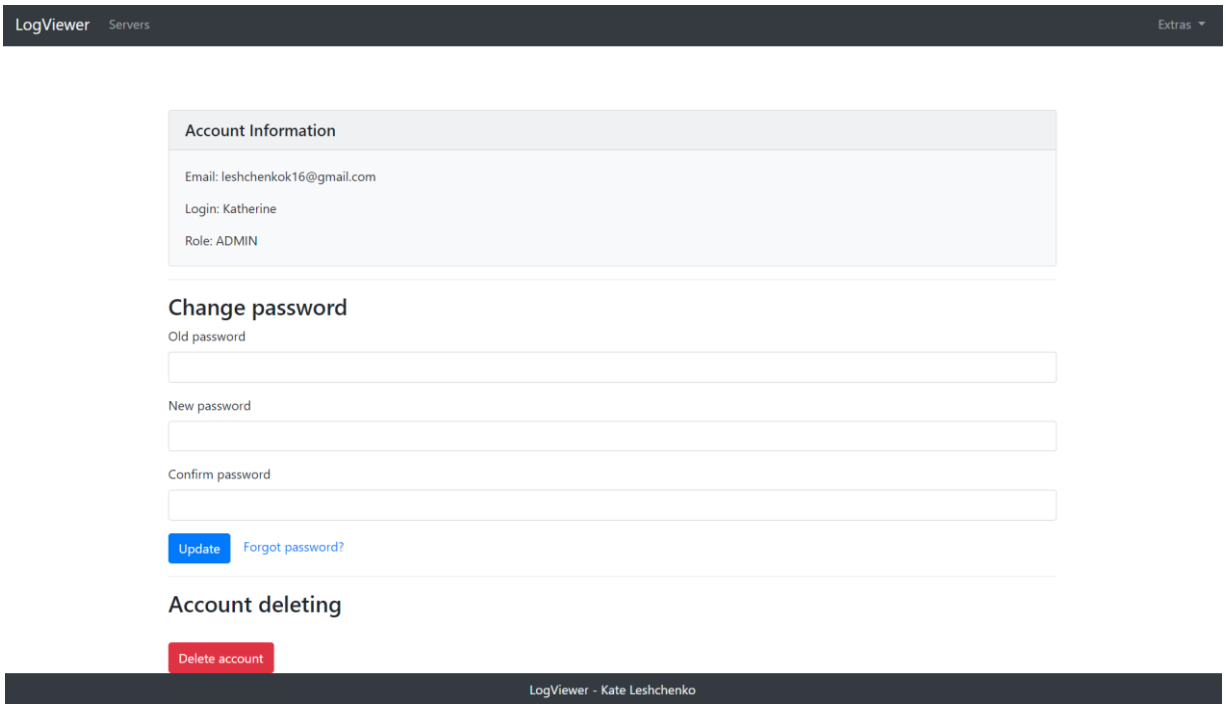

Рисунок 4.14 – Кінцевий вигляд продукту – сторінка зміни паролю

Рис. 4.15 – відображає список папок що знаходяться на обраному сервері. На цій сторінці можна побачити адресу папки, час останньої перевірки на активність. Також сторінка показує чи є необхідність перевіряти активність цієї папки, та чи доступна вона. З неї можна потрапити до відображення усіх логів цієї папки. Також на сторінку відображення усіх файлів з цієї директорії, оновити дані. Включити чи видалити з опитувань на активність. Останнім є можливість видалення цієї директорії.

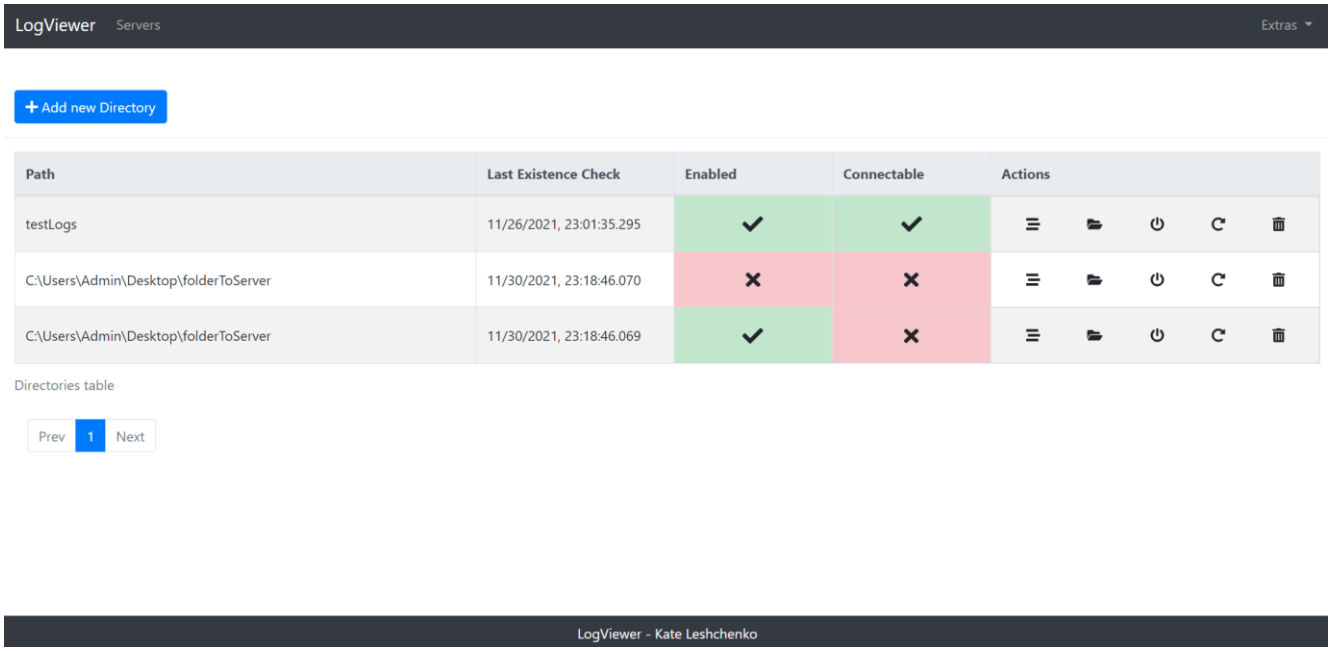

Рисунок 4.15 – Кінцевий вигляд продукту – сторінка списку папок

Рис. 4.16 – сторінка перегляду файлів директорії. Вона відображає файли з логами, які знаходяться у цій директорії, а отже додати можна лише ті, що вже там присутні. Цю сторінку було вирішено додати після обирання методів за якими будуть переглядатися логи, тому вона не відображена на прототипах. Також, тут можна переглянути назву файлу, час коли виконувалась остання перевірка можливості доступу джобою. Є можливість перейти до перегляду логів у реальному часі, а також до сторінки перегляду логів з можливістю сортування та фільтрації, та присутня можливість видалення файлів з логами.

Рис. 4.17 – відображає основну сторінку з логами, що розташовані у певній папці. Тут є можливість видаляти логи поодинці, групою, чи усі разом. Також

відображено основний функціонал системи. Таким чином, є можливість виконувати пошук логів, за присутнім у ньому тексті. Сортувати логи за рівнем та датою. За замовченням виконується сортування за датою. Ще є можливість виконувати фільтрацію за датою, тобто обирати період за який необхідно відобразити логи. Це може бути не лише дата, а ще й час. На цій сторінці можна виконувати сортування за рівнем, тобто обрати логи лише, наприклад, рівня ERROR, і тоді буде відображено які помилки виникли під час роботи того чи іншого програмного продукту. Ще невелика особливість є у тому що можна подивитися окремо ті логи, які прийшли без зазначення певного рівня до якого вони відносяться. Праворуч під усіма можливостями фільтрацій та сортувань присутня кнопка, яка дозволяє застосувати усі обрані налаштування.

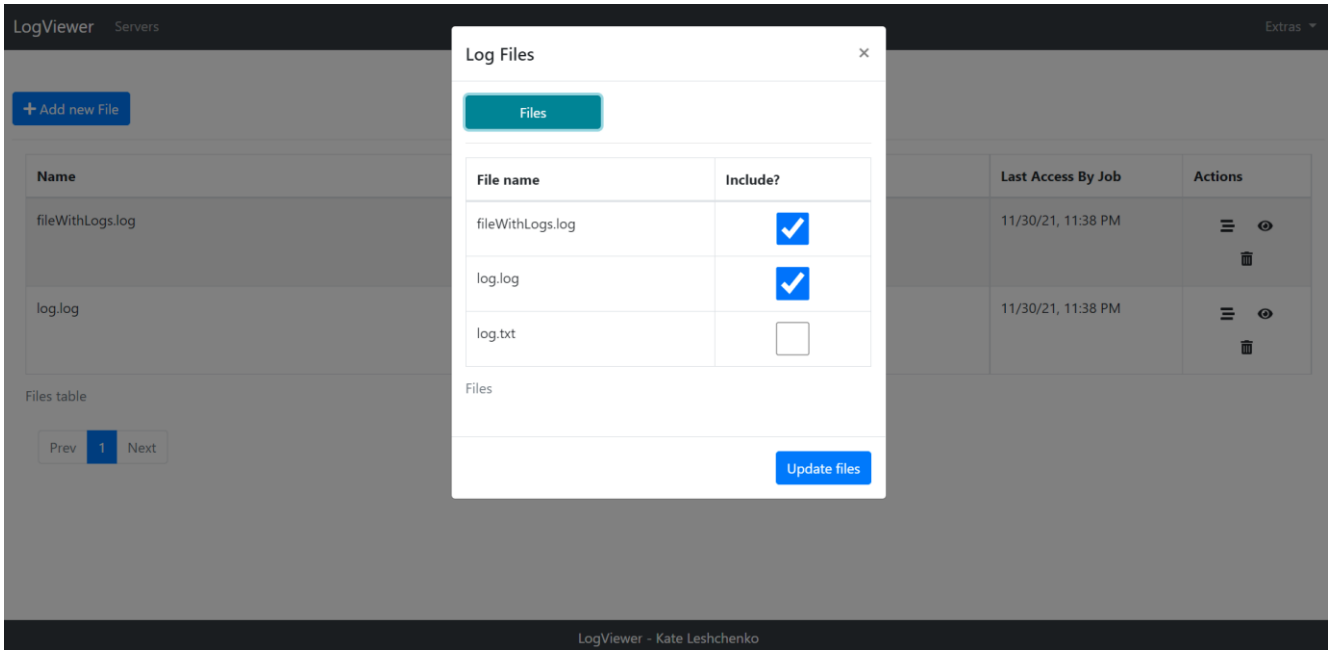

Рисунок 4.16 – Кінцевий вигляд продукту – сторінка списку файлів

Рис. 4.18 – останньою наведена сторінка перегляду логів у режимі реального часу. Тут відображаються логи з усього серверу, що додаються у реальному часі. Є можливість включити автоскол, щоб завжди було видно останні логи з серверу. Також якщо з’явилася важлива інформація, логи можна одразу зберегти у окремий файл, щоб у подальшому переглянути інформацію наприкінці нього.

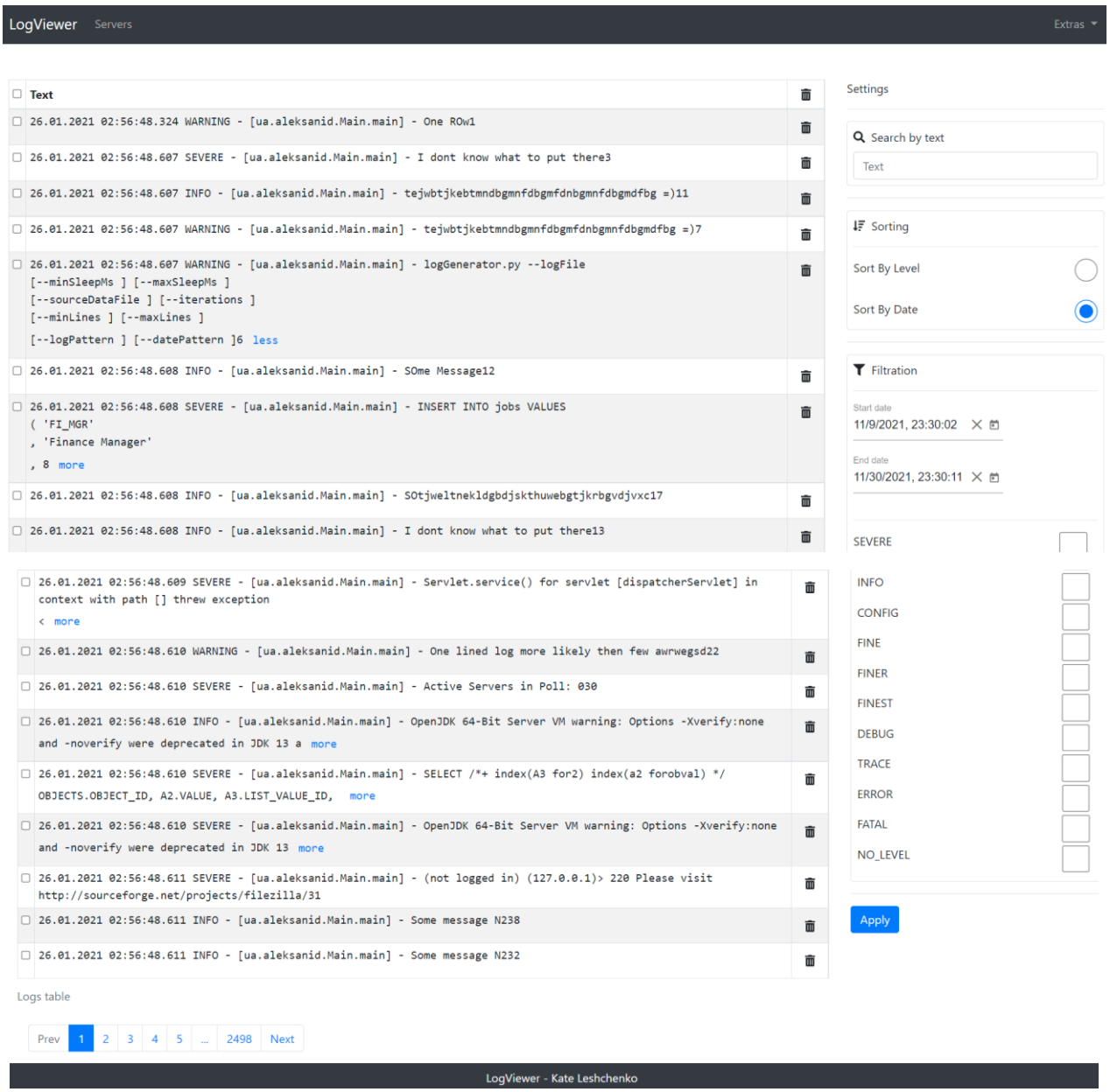

Рисунок 4.17 – Кінцевий вигляд продукту – сторінка перегляду логів

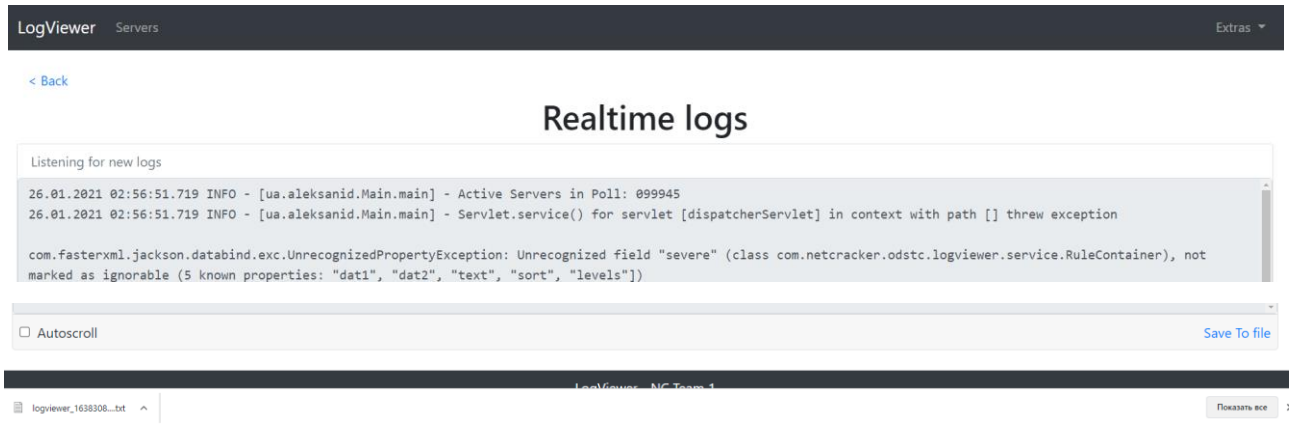

Рисунок 4.18 – Кінцевий вигляд продукту – сторінка перегляду логів у режимі реального часу

4.5 Структура програмного продукту

Структуру програмного продукту можна побачити на рис. 4.19.

Програма збирає логи з серверів і передає їх в базу даних. Користувач може зайти на сайт на якому є можливість: впорядкувати, відфільтрувати логи за рівнем, датою або знайти конкретний лог по тексту. Також була продумана можлива перевантаженість бази даних і для чого була створена задача на стороні баз даних. Її основна робота – це перевірка логів на те застаріли вони чи ні. У програмі є 2 пула серверів, які підключені і не підключені. Підключення сервера опитується через певний період часу, який задав адміністратор в глобальних налаштуваннях. Для цього була придумана та використана задача на стороні back–end. Принцип цієї задачі включає в себе перевірку сервера на те підключений він чи ні, після цього йде перехід в директорію і відбувається перевірка на наявність нової інформації в файлі. Чи не підключені сервера ми опитуємо через іншу задачу. Адміністратор задає період через який програма перевіряє сервера на можливість підключення до них.

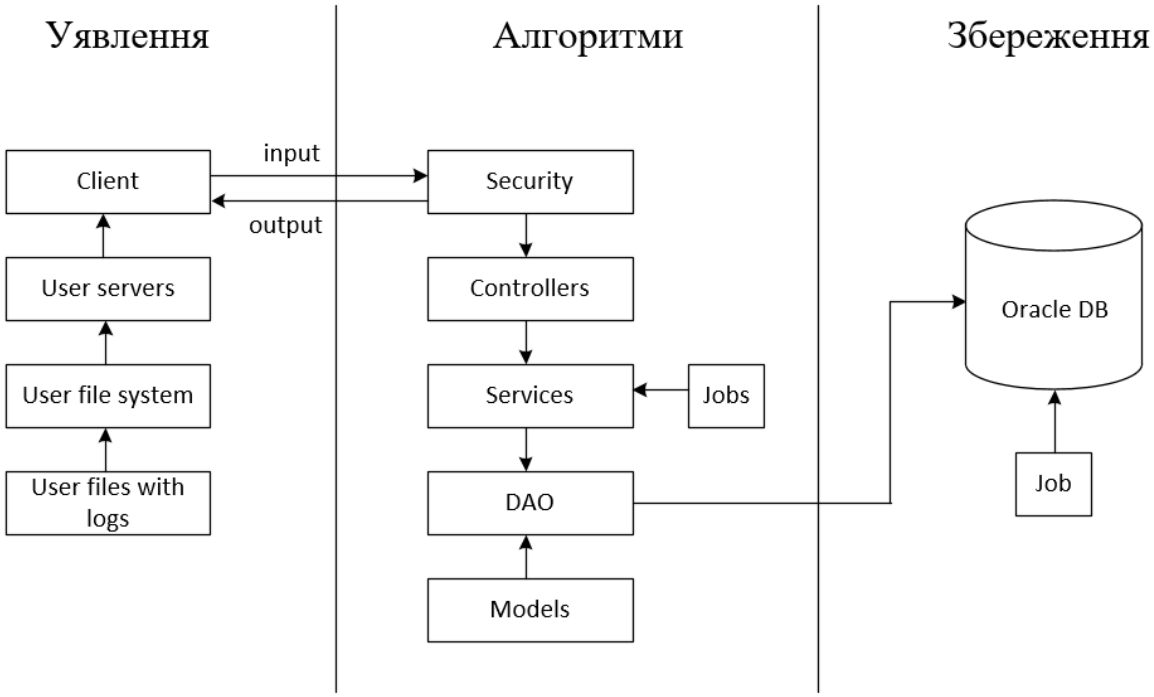

Рисунок 4.19 – Структура програмного продукту

5 ПРОГРАМНА РЕАЛІЗАЦІЯ

5.1 Проектування системи

Для розуміння структури розроблюваного програмного засобу створена діаграма класів вона відображена на рис. 5.1. Програмний засіб заснований на патерні MVC, DAO і класи моделей виконують роль Model, за перехоплення дій користувачів відповідають контролери. У кожного з них є своя зона відповідальності, всередині якої, відповідаючи FrontEnd запитам, що надходять. Контролери приймають вхідні дані, потім передають їх класам – сервісам, і повертають відповідний запит – результат. Запити, що перехоплюються контролерами є досить простими, наприклад – додавання користувача або зміна імені сервера.

Валідація даних, що надходять проводиться як на FrontEnd-е, так і на BackEnd-е. Для того, щоб в базу даних не потрапили некоректні дані існують сервіси, які і є прошарком між контролерами і DAO, і беруть на себе додаткову валідацію об'єктів.

Підключення до серверів. Підтримуються два протоколи SSH і FTP. Для підключення по SSH використана бібліотека JSCh. Для підключення по FTP використано FTPClient з бібліотеки Apache.commons. Net.

Процес збору логів. Збираються всі включені сервера з бази даних, направляються в пул потоків, які опитують сервера. До кожного встановлюється з'єднання. Після чого перебираються всі зазначені директорії і файли. У кожного файлу, є останній зчитаний рядок. Якщо щось з'явилося після цього рядка, то це означає що є нові логи, і тоді проглядаються нові рядки. Потім перевіряється чи є вони придатними під шаблони логами, і вони переводяться з рядка в об'єкт Log.

Самі опитування ділиться на дві частини. Збереження логів, і збереження проміжних станів. Кожен період, заданий адміністраторами, перевіряється результат збору логів. Якщо є нові логи, то вони зберігаються в БД. Далі перевіряється, чи закінчене повністю опитування. Якщо так, то зберігаються і проміжні дані серверів, директорій, файлів і планується нове опитування.

Збереження проміжних станів потрібні, щоб якщо під час опитування серверів, сталася помилка, наприклад, з доступом до сервера або директорії, зберегти це в БД.

За новими логами можна спостерігати в реальному часі. Це реалізовано за допомогою патерна Паблишер-Сабскрайбер. За ДАО закріплений слухач, якому після збереження логів, відправляється інформація про них, яку він передає на сокет з'єднання, і далі логи відправляються безпосередньо на Фронт-Енд.

У даному програмному продукті питання безпеки реалізовано на основі технології авторизації JWT.

JWT – JSON web token, замість куки для автентифікації використовує токен – json в закодованому вигляді[15].

Токени створюються сервером, який в подальшому використовує даний токен для підтвердження своєї особи. У токені ми прописуємо дані користувача, такі як ім'я, роль користувача і задаємо дату закінчення нашого токену. Після токен підписується за допомогою секретного ключа і шифрується.

Для входу в систему, користувач повинен відправити валідний логін і пароль, після чого запит обробляється spring security, і направляється до фільтру автентифікації, де сервер на основі цих даних шукає користувача в базі і автентифікує його. Якщо користувач знайшовся і автентифікувався, то на основі секретного ключа формується валідний jwt токен, і відправляється назад клієнту. Тепер якщо користувач хоче отримати доступ до даних сервера, він повинен відправляючи запити, кожен раз додавати в заголовок запиту цей токен.

На стороні UI клієнта, цю функцію бере на себе Angular, а саме AuthInterceptor, він дістає з локального сховища токен, і додає в заголовок.

Далі по ланцюжку фільтрів йде наступний, так званий, фільтр авторизації, який на стороні сервера буде діставати з кожного запиту токен, перевіряти його на валідність, перевіряти права доступу користувача, і на основі цих даних видавати дозвіл на отримання даних, або навпаки забороняти.

У даному програмному продукті є два типи користувача: Юзер і Адмін, які відрізняються своїми дозволами. Адмін може те ж саме, що і користувач, а також

має доступ до свого контролера для зміни загальних параметрів і маніпуляцією користувачами в базі.

По закінченню терміну дії токена, користувач буде вважатися не авторизованим. Тепер йому потрібно пройти заново процес автентифікації, для створення нового, валидного jwt токена.

Основою взаємодії програмного засобу з користувачем є інтерфейс користувача. Графічний інтерфейс користувача поділено на кілька частин. Основними сторінками програмного продукту є LogView і RealTimeView. LogView дає можливість сортувати і фільтрувати логи користувача. RealTimeView реалізує висновок логів в реальному часі, завдяки сокетсервісу, який використовує патерн паблішер-сабскрайбер.

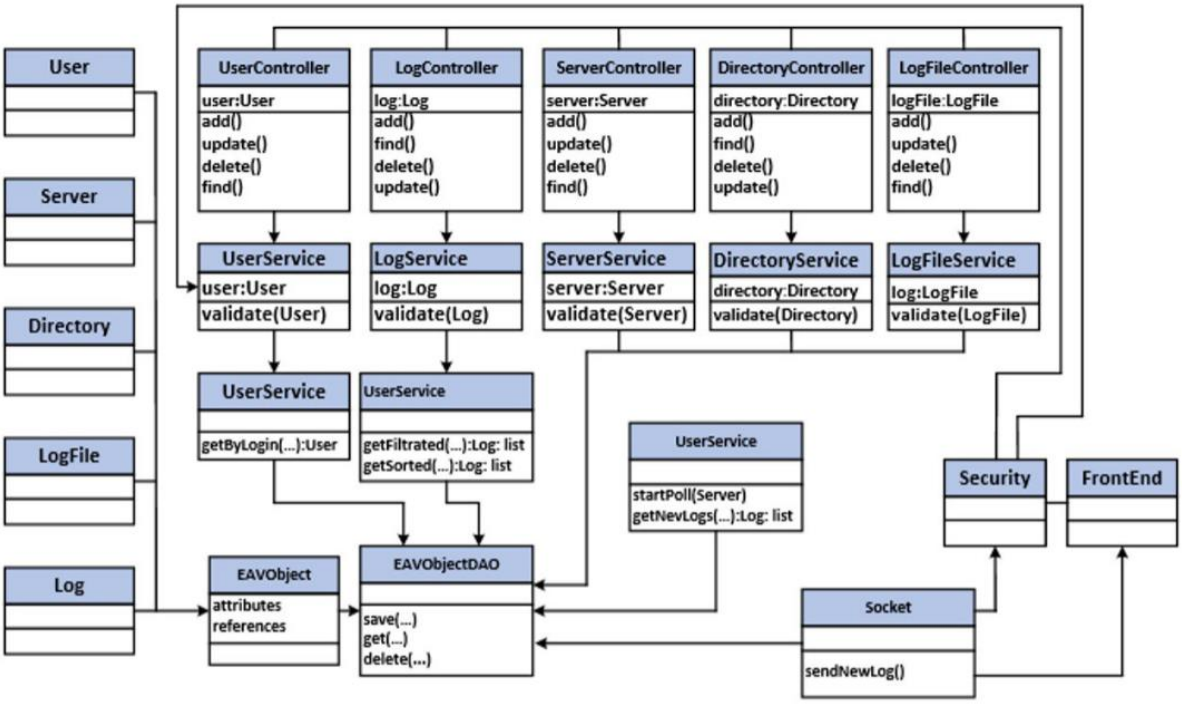

Рисунок 5.1 – Діаграма класів

Нижче наведено опис програмних основних програмних класів.

class RealtimeLogViewerApplication – основний клас для запуску роботи програми.

main(String[] args) – метод що запускає роботу програми.

class DataSourceConfig – клас конфігурацій бази даних.

getJdbcTemplate() – метод, що створює підключення бази даних.

getDataSource() – метод, що задає параметри бази даних.

class MailConfig – клас конфігурацій листа.

setMailSender() – метод, що задає усі необхідні параметри для відправлення листа.

class WebSecurityConfig – клас конфігурацій безпечної автентифікації.

WebSecurityConfig() – конструктор, що ініціалізує UserDetailsServiceImpl, та SecuritySettings.

bCryptPasswordEncoder() – метод, що ініціалізує клас, для кодування паролю.

configureGlobal() – метод, що виконує автентифікацію.

configure() – метод, що виконує кодування даних користувача.

class WebsocketConfiguration – клас для налаштувань протоколів обміну повідомленнями.

WebsocketConfiguration() – конструктор, що ініціалізує SecurityService.

configureMessageBroker() – метод, що задає параметри брокера повідомлень.

registerStompEndpoints() – метод, що регіструє кінцеву точку повідомлень.

configureClientInboundChannel() – метод, що задає налаштування каналу повідомлень для вхідних повідомлень.

SocketMessagingException(String message) – метод для повідомлення про помилки з каналом повідомлень.

class AttributeObjectContainerConverter – клас для конвертації даних бази (логів) у ієрархічну модель.

convertAttributeObjectContainerToHierarchyContainer() – метод, що виконує конвертацію атрибутів у ієрархічний контейнер.

setUpHierarchy() – метод, що виконує налаштування ієрархій об’єктів.

appendAttribute() – метод, що доповнює атрибути значеннями.

class LogDTO – POJO клас даних про логи, де зазначені ідентифікатор, рівень, саме повідомлення, та дата створення.

class AttributeObjectContainer – POJO клас даних об’єктів та атрибутів бази даних, що включає в себе ідентифікатори об’єктів, їх «батьків», типу, атрибутів, та листового значення, а також назву, назву, значення, та значення дати.

class HierarchyContainer – клас, що визначає ієрархію даних у базі.

HierarchyContainer() – конструктор, що ініціалізує EAVObject.

HierarchyContainer() – конструктор без параметрів, що створює пустий список «дітей».

Також клас включає в себе POJO методи для «дітей», «батьків», та самих об’єктів.

Також перевизначені методи для порівняння даних equals та hashCode.

class RuleContainer – POJO клас з даними для фільтрації та сортування за текстом, датами, та рівнем.

enum SortType – клас (перечислення) для визначення типу сортування

class DirectoryController – клас контролер, для обробки запитів клієнта щодо директорій.

DirectoryController() – конструктор, що ініціалізує DirectoryService.

getDirectoriesByParentId() – метод, що відловлює запит клієнта на отримання директорії по ідентифікатору «батька».

addDirectory() – метод, що відловлює запит клієнта на додавання нової директорії.

updateDirectory() – метод, що відловлює запит клієнта на оновлення директорії.

deleteDirectoryById() – метод, що відловлює запит клієнта на видалення директорії по ідентифікатору.

getDirectoryById() – метод, що відловлює запит клієнта на отримання директорії по ідентифікатору.

testConnectionToDirectory() – метод, що тестує можливість підключення до директорії.

class LogController – клас контролер, для обробки запитів клієнта щодо логів.

LogController() – конструктор, що ініціалізує LogService.

logs() – метод, що відловлює запит клієнта на отримання усіх логів.

getLogByFileId() – метод, що відловлює запит клієнта на отримання логів по ідентифікатору файла.

add() – метод, що відловлює запит клієнта на додавання нового лога.

deleteById() – метод, що відловлює запит клієнта на видалення одного логу по ідентифікатору.

deleteByIds() – метод, що відловлює запит клієнта на видалення декількох логів за ідентифікаторами.

findById() – метод, що відловлює запит клієнта на пошук логу за ідентифікатором.

class LogFileController – клас контролер, для обробки запитів клієнта щодо файлів з логами.

LogFileController() – конструктор, що ініціалізує LogFileService, та DirectoryService.

showAllLogFiles() – метод, що відловлює запит клієнта на отримання усіх файлів з логами.

getLogFilesFromDirectory() – метод, що відловлює запит клієнта на отримання усіх файлів з логами з конкретної директорії за її ідентифікатором.

getLogFileById() – метод, що відловлює запит клієнта на отримання файлу з логами по ідентифікатору.

addLogFile() – метод, що відловлює запит клієнта на додавання нового файлу з логами.

addLogFiles() – метод, що відловлює запит клієнта на додавання декількох файлів з логами.

deleteById() – метод, що відловлює запит клієнта на видалення файлу з логами по ідентифікатору.

class ServerController – клас контролер, для обробки запитів клієнта щодо серверів.

ServerController() – конструктор, що ініціалізує ServerService, та UserService.

showAllServers() – метод, що відловлює запит клієнта на отримання усіх серверів.

add() – метод, що відловлює запит клієнта на додавання нового серверу.

update() – метод, що відловлює запит клієнта на оновлення даних серверу.

updateLastAccessByJob() – метод, що відловлює запит клієнта на оновлення останнього доступу задачі (частина коду, яка повторюється через певний (заданий) період часу) до серверу за ідентифікатором.

findById() – метод, що відловлює запит клієнта на пошук серверу за ідентифікатором.

deleteById() – метод, що відловлює запит клієнта на видалення серверу по ідентифікатору.

testConnection() - метод, що тестує можливість підключення до серверу.

class UserController – клас контролер, для обробки запитів клієнта щодо користувачів.

UserController() – конструктор, що ініціалізує SecurityService, та UserService.

getUsers() – метод, що відловлює запит клієнта на отримання усіх користувачів.

create() – метод, що відловлює запит клієнта на створення нового користувача.

update() – метод, що відловлює запит клієнта на оновлення даних користувача.

updatePassword() – метод, що відловлює запит клієнта на оновлення паролю користувача.

resetPassword() – метод, що відловлює запит клієнта на скидання паролю користувача.

changePassword() – метод, що відловлює запит клієнта на змінення паролю користувача.

checkPassword() – метод, що відловлює запит клієнта на перевірку паролю користувача.

findById() – метод, що відловлює запит клієнта на пошук користувача за ідентифікатором.

getCurrentUser() – метод, що відловлює запит клієнта на отримання поточного користувача.

deleteById() – метод, що відловлює запит клієнта на видалення користувача по ідентифікатору.

updateSettings() – метод, що відловлює запит клієнта на оновлення глобальних налаштувань.

getConfig() – метод, що відловлює запит клієнта на оновлення конфігурацій.

class ContainerDAO – клас для роботи з базою контейнерів.

ContainerDAO() – конструктор, що ініціалізує JdbcTemplate.

getActiveServersWithChildren() – метод, що отримує з бази активні сервера з «дітьми».

getNonactiveServers() – метод, що отримує з бази усі неактивні сервера.

getActiveServersWithNonactiveDirectories() – метод, що отримує з бази усі активні сервера з неактивними директоріями.

getHierarchyContainerFromQuery() – метод, що отримує з запиту ієрархічний контейнер.

validateServerContainer() – метод, що валідує контейнери серверів.

validateChildrenNotEmpty() – метод, що видаляє сервера з пустими «дітьми».

class EAVObjectDAO – клас для роботи з основними запитами до бази.

EAVObjectDAO() – конструктор, що ініціалізує JdbcTemplate.

getObjectsByParentId() – метод, що отримує об’єкти за ідентифікатором «батьків» з бази.

getObjectsByObjectTypeId() – метод, що отримує об’єкти за ідентифікатором типу об’єктів з бази.

getObjectByIdAttrByIds() – метод, що отримує об’єкт та атрибути за ідентифікаторами з бази.

getObjectById() – метод, що отримує об’єкт за ідентифікатором з бази.

getAttributeByValue() – метод, що отримує об’єкт за значенням з бази.

getAttributeByDateValue() – метод, що отримує об’єкт за датою з бази.

getAttributeByListValue() – метод, що отримує об’єкт за списковим значенням з бази.

saveObjectAttributesReferences() – метод, що зберігає об’єкт з атрибутами та зв’зками у базі.

saveObjectsAttributesReferences() – метод, що зберігає об’єкти з атрибутами та зв’зками у базі.

saveObjects() – метод, що зберігає об’єкти у базі.

saveObject() – метод, що зберігає один об’єкт у базі.

saveAttributes() – метод, що зберігає атрибути у базі.

saveReferences() – метод, що зберігає зв’язки у базі.

deleteById() – метод, що видаляє об’єкт з бази за ідентифікатором.

nextObjectId() – метод, що отримує ідентифікатор наступного об’єкту з бази.

getReference() – метод, що отримує зв’язок за ідентифікатором з бази.

getAttributes() – метод, що отримує атрибути за ідентифікатором з бази.

getObjectsByQuery() – метод, що отримує об’єкти за значенням з бази.

getObjectsByIds() – метод, що отримує об’єкти за ідентифікаторами з бази.

getAttribute() – метод, що отримує атрибут за значенням з бази.

class LogDAO – клас для роботи зі специфічними запитами до бази для логів.

LogDAO() – конструктор, що ініціалізує JdbcTemplate.

getLogByDirectoryId() – метод, що отримує логи за ідентифікатором директорії з бази.

getLogByFileId() – метод, що отримує логи за ідентифікатором файлу з бази.

convertEnumListToIntList() – метод, що конвертує листове значення Enum у Int.

class UserDao – клас для роботи зі специфічними запитами до бази для користувачів.

UserDao() – конструктор, що ініціалізує JdbcTemplate.

getByLogin() – метод, що отримує користувача за логіном з бази.

getUsers() – метод, що отримує усіх користувачів з бази.

class AttributeMapper – клас, що виконує зіставлення кожних рядків даних атрибутів.

mapRow() – метод, що виконує зіставлення кожних рядків даних атрибутів.

class AttributeObjectMapper – клас, що виконує зіставлення кожних рядків даних атрибутів та об’єктів.

mapRow() – метод, що виконує зіставлення кожних рядків даних атрибутів та об’єктів.

class LogDTOMapper – клас, що виконує зіставлення кожних рядків даних логів.

mapRow() – метод, що виконує зіставлення кожних рядків даних логів.

class ObjectMapper – клас, що виконує зіставлення кожних рядків даних об’єктів.

mapRow() – метод, що виконує зіставлення кожних рядків даних об’єктів.

class ReferenceMapper – клас, що виконує зіставлення кожних рядків даних зв’язків.

mapRow() – метод, що виконує зіставлення кожних рядків даних зв’язків.

class UserMapper – клас, що виконує зіставлення кожних рядків даних користувачів.

mapRow() – метод, що виконує зіставлення кожних рядків даних користувачів.

class Config – POJO клас з даними екземплярів, періоди опитування активності, опитування змін, зберігання логів, активності директорій, та серверів.

Config() – конструктор без параметрів, що визначає формат дати.

Config() – конструктор з параметром, що визначає формат дати, та ідентифікатор об’єкту.

class Directory – POJO клас з даними путі, включення, можливості підключення, останнього доступу користувача, та перевірки наявності.

Directory() – конструктор без параметрів, що визначає тип об’єкту самостійно.

Directory() – конструктор з параметром, що визначає тип об’єкту за вхідним параметром.

Directory() – конструктор з параметрами, що задає путь до директорії, що вона увімкнена та доступна, дату останньої перевірки, та доступу користувача.

class Log – POJO клас з даними тексту, рівня, та дати створення логів.

Log() – конструктор без параметрів, що визначає тип об’єкту самостійно.

Log() – конструктор з параметром, що визначає тип об’єкту за вхідним параметром.

Log() – конструктор з параметрами, що задає рівень логу, дату створення та ідентифікатором «батьків».

class LogFile – POJO клас з даними імені, останнього оновлення, та останнього рідка файлу з логами.

LogFile() – конструктор без параметрів, що визначає тип об’єкту самостійно.

LogFile() – конструктор з параметром, що визначає тип об’єкту за вхідним параметром.

LogFile() – конструктор з параметрами, що задає назву, останній рядок, та останнє оновлення файлу.

class Server – POJO клас з даними ip-адреси, логіну, паролю, протоколу, порту, включення, підключення, остання перевірка задачі, та користувача до серверу.

Server() – конструктор без параметрів, що визначає тип об’єкту самостійно.

Server() – конструктор з параметром, що визначає тип об’єкту за вхідним параметром.

Server() – конструктор з параметрами, що задає ip-адресу, порт, логін, пароль, протокол, що сервер включено та він доступний, дату останньої перевірки задачею, та доступу користувача.

class User – POJO клас з даними поштової адреси, логіну, паролю, ролі, ким створений користувач.

User() – конструктор без параметрів, що визначає тип об’єкту самостійно.

User() – конструктор з параметром, що визначає тип об’єкту за вхідним параметром.

User() – конструктор з параметрами, що задає поштову адресу, логін, пароль, роль, та ким створений.

Та перевизначені для порівняння користувачів методи equals та hashCode.

Ще є ряд класів що визначають винятки для деяких випадків. Також описано моделі списків, та моделі EAV бази даних. Класи для реалізації безпеки, безпечної

автентифікації, та авторизації користувачів. Класи, що відтворюють підключення до SSH та FTP серверів.

Ще є ряд класів сервісів, що виконують усе те, що виловлюють класи контролери Клас, що виконує «прослуховування» на присутність нових логів.

5.2 Проектування бази даних

Проектування бази даних. Для бази даних використовували EAV модель. Модель Сутність-Атрибут-Значення (Entity-attribute-value model) – це модель даних, призначена для опису сутностей, в яких кількість атрибутів (властивостей, параметрів), що характеризують їх, потенційно величезна, але то кількість, яке реально буде використовуватися в конкретній суті, відносно мало[16]. Схему бази даних даного програмного продукту наведено на рис. 5.2.

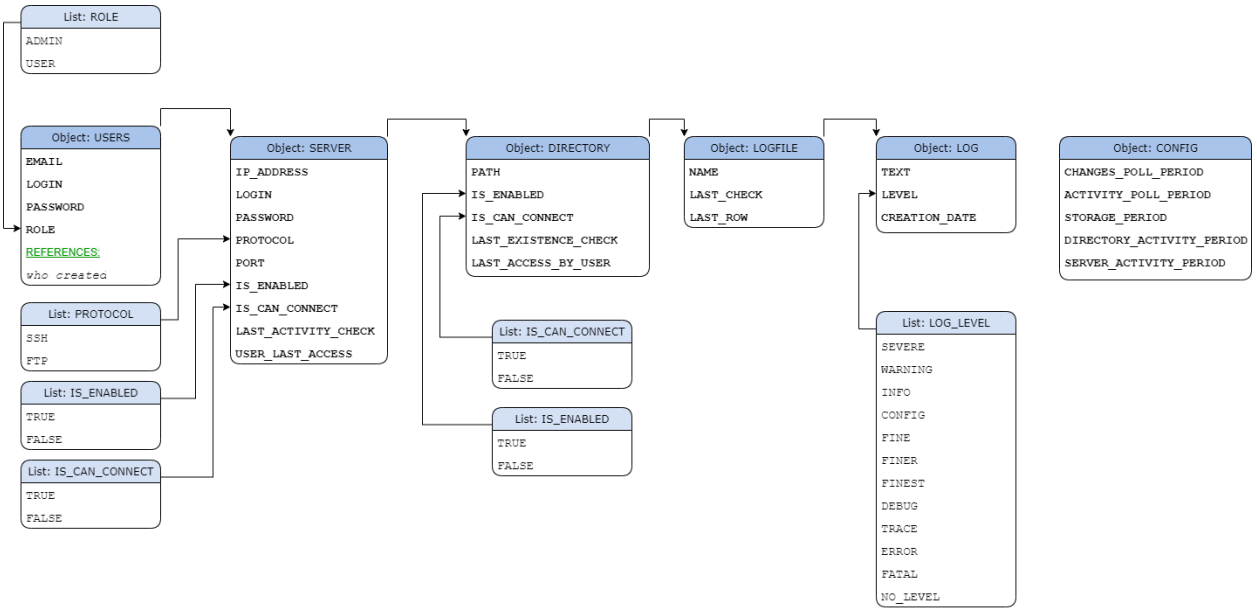

Рисунок 5.2 – Схема бази даних

Для відображення предметної області було обрано шість сутностей. Перша – Юзер, для відображення користувачів програми. Вона містить стандартні поля, логін, пароль, роль, хто створив. Наступна – Сервер, належить користувачеві, та містить основні поля для роботи з віддаленим сервером (IP, логін, пароль для авторизації). На сервері розташовані директорії – основні атрибути якої: шлях до

директорії, включена вона, або виключена. У директоріях – Лог Файли – по суті звичайний файл в будь-якої файлової системи. Містить ім'я, останній зчитаний рядок і коли він останній раз був прочитаний. І власне сам лог, що містить текст, рівень і дату створення. Рівні представлені списком значень. Список брався з бібліотек JUL, log4j, Slf4j. Також в силу того, що час логів треба зберігати з точністю до мілісекунд, була трохи змінена структуру в атрибутах Date на timestamp.

Ще для бази даних були створені дві процедури. Одна для заповнення тестовими даними, яка застосовувалася під час тестування. І друга, для очищення логів. Яка запускається щодня, і з огляду на поточні настройки, очищає логи.

5.3 Опис усіх елементів бази даних

Дані бази даних представлено у табличному вигляді (табл. 5.1 – 5.13), з описаними ключами, атрибутами та їх призначенням.

Таблиця 5.1 – Об’єкт USERS

Назва атрибуту	Тип даних	Призначення
EMAIL	varchar(50)	Унікальна адреса пошти, куди доставляються посилання на відновлення паролю
LOGIN	varchar(20)	Логін для входу користувача
PASSWORD	varchar(20)	Пароль для входу в систему
ROLE	list value	Роль, яка визначає можливості користувача.
who created	references	Посилання на адміністратора, який зробив даного користувача (або адміністратора)

Таблиця 5.2 – Список Role

Назва атрибуту	Значення	Призначення
Admin	‘ADMIN’	Роль яка має повний спектр функціоналу ролі User, та ще деякі можливості для загального налаштування усього програмного продукту
User	‘USER’	Роль яка має усі можливості для аналізу логів

Таблиця 5.3 – Об’єкт SERVER

Назва атрибуту	Тип даних	Призначення
IP_ADDRESS	varchar(50)	Унікальна адреса за якою необхідно підключатися до серверу
LOGIN	varchar(20)	Логін для підключення до серверу
PASSWORD	varchar(20)	Пароль для підключення до серверу
PROTOCOL	list value	Протокол, за яким необхідно підключатися до серверу
PORT	int(4)	Порт за яким знаходиться сервер
IS_ENABLED	list value	Значення, що показує необхідність підключення до серверу
IS_CAN_CONNECT	list value	Значення, що показує можливість підключення до серверу
LAST_ACTIVITY_CHECK	datatime	Дата та час останньої перевірки активності серверу
USER_LAST_ACCESS	datatime	Дата та час останнього доступу користувача до даного серверу

Таблиця 5.4 – Список PROTOCOL

Назва атрибуту	Значення	Призначення
SSH	‘SSH	SSH – мережевий протокол який необхідно використовувати для збирання логів з серверу
FTP	‘FTP	FTP – стандартний мережевий протокол який необхідно використовувати для збирання логів з серверу

Таблиця 5.5 – Список IS_ENABLED

Назва атрибуту	Значення	Призначення
True	‘TRUE’	Необхідне підключення до серверу за для збирання логів
False	‘FALSE’	Даний сервер відключено

Таблиця 5.6 – Список IS_CAN_CONNECT

Назва атрибуту	Значення	Призначення
True	‘TRUE’	До даного серверу є можливість підключення
False	‘FALSE’	До даного серверу відсутня можливість підключення

Таблиця 5.7 – Об’єкт DIRECTORY

Назва атрибуту	Тип даних	Призначення
PATH	varchar(150)	Унікальна для даного серверу адреса за якою необхідно збирати файли логів
IS_ENABLED	list value	Значення, що показує необхідність підключення до директорії
IS_CAN_CONNECT	list value	Значення, що показує можливість підключення до директорії
LAST_EXISTENCE_CHECK	datatime	Дата та час останньої перевірки на існування директорії
USER_LAST_ACCESS	datatime	Дата та час останнього доступу користувача до даної директорії

Таблиця 5.8 – Список IS_ENABLED

Назва атрибуту	Значення	Призначення
True	‘TRUE’	Необхідне підключення до директорії за для збирання логів
Назва атрибуту	Значення	Призначення
False	‘FALSE’	Дану директорію видалено зі списку активних

Таблиця 5.9 – Список IS_CAN_CONNECT

Назва атрибуту	Значення	Призначення
True	‘TRUE’	До даної директорії є можливість підключення
False	‘FALSE’	До даної директорії відсутня можливість підключення

Таблиця 5.10 – Об’єкт LOGFILE

Назва атрибуту	Тип даних	Призначення
NAME	varchar(100)	Унікальна назва файлу для даної директорії
LAST_CHECK	datatime	Дата та час останньої перевірки існування файлу з логами
LAST_ROW	int	Номер останнього зчитаного рядка

Таблиця 5.11 – Об’єкт LOG

Назва атрибуту	Тип даних	Призначення
TEXT	varchar(25000)	Текст логу

Продовження таблиці 5.11

Назва атрибуту	Тип даних	Призначення
LEVEL	list value	Рівень логу
LAST_ROW	int	Номер останнього зчитаного рядка

Таблиця 5.12 – Список LOG_LEVEL

Назва атрибуту	Значення	Призначення
Severe	‘SEVERE’	Рівень логу – Severe
Warning	‘WARNING’	Рівень логу – Warning
Info	‘INFO’	Рівень логу – Info
Config	‘CONFIG’	Рівень логу – Config
Fine	‘FINE’	Рівень логу – Fine
Finer	‘FINER’	Рівень логу – Finer
Finest	‘FINEST’	Рівень логу – Finest
Debug	‘DEBUG’	Рівень логу – Debug
Trace	‘TRACE’	Рівень логу – Trace
Error	‘ERROR’	Рівень логу – Error
Fatal	‘FATAL’	Рівень логу – Fatal
No_Level	‘NO_LEVEL’	Рівень логу – відсутній

Таблиця 5.13 – Об’єкт CONFIG

Назва атрибуту	Тип даних	Призначення
CHANGES_POLL_PERIOD	int	Час через який видаляється роль
ACTIVITY_POLL_PERIOD	int	Період активності сеансу
STORAGE_PERIOD	int	Період зберігання логів
DERECTORY_ACTIVITY_PERIOD	int	Період активності директорії, через цей час вона перестає бути активним
SERVER_ACTIVITY_PERIOD	int	Період активності серверу, через цей час він перестає бути активним

5.4 Програмне тестування

Для даного програмного продукту було вирішено написати автотести, які будуть контролювати чи зламана та чи інша частина. Тестування виконувалось з використанням бібліотеки JUnit тестів. JUnit – бібліотека для модульного

тестування програм Java [17]. Тестами було покрито більш ніж половина програми, це відображено на рис. 5.3. Мною було обрано найважливіші класи від яких залежить робота усього програмного продукту. За умовою коли один з них перестане виконувати свою функцію, то це потягне за собою інші проблеми у даному програмному продукті.

Отже, для тестування було обрано головні класи роботи з базою даних, вони були протестовані на 80%. В роботі було протестовано усі сервіси, що покриває собою усі контролери, тому їх тестування майже не проводилось. Тестування сервісів покриває майже весь основний функціонал програмного продукту. Також проводилось тестування на коректність підключення до серверу.

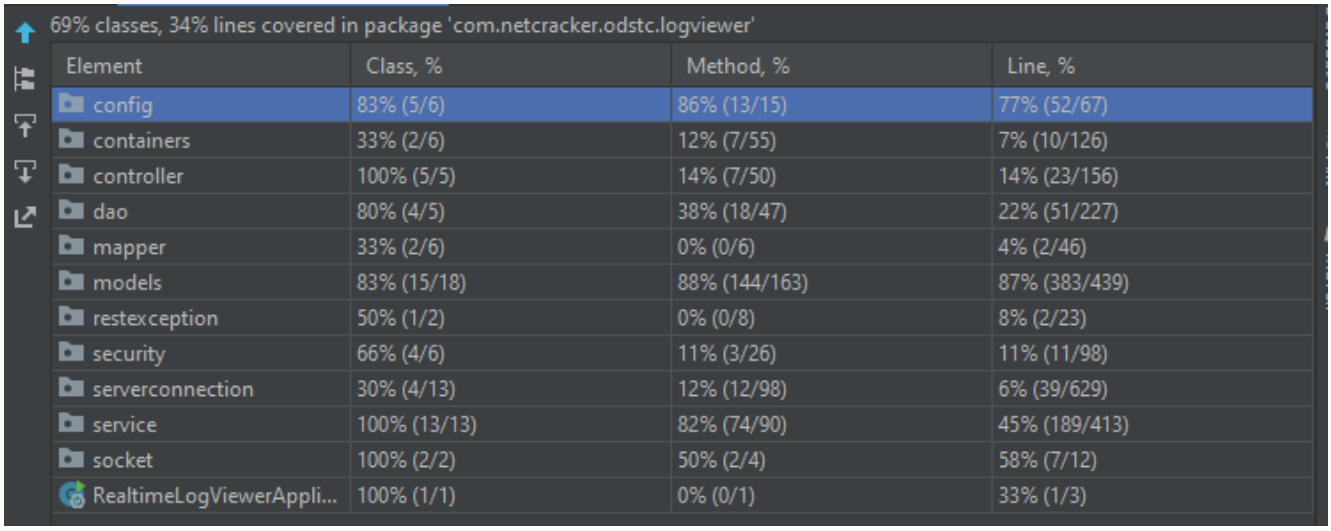

Рисунок 5.3 – Результати покриття тестами програмного коду

6 РОЗРАХУНОК ЕФЕКТИВНОСТІ ПРОГРАМНОГО ПРОДУКТУ

Для визначення ефективності програмного продукту було запрошено 5 програмістів, які вже працюють досить довгий час у цій сфері, 5 студентів 5 курсу спеціальності система програмна інженерія, та 5 студентів 1 курсу спеціальності інженерія програмного забезпечення. Нижче у табл. 6.1 (де мд – означає «мій програмний продукт», а ан – «програма аналог») зазначено номер кожного зі студентів і кількість затраченого часу на виявлення проблеми за допомогою мого програмного забезпечення, та основного аналога, з визначенням рангу для кожного зі значень. Розрахунок виконувався за допомогою критерія Вілкоксона-Манна-Уітні.

Учасники мали знайти у виданому їм програмному продукті логи, що вказують на проблему, яку необхідно вирішити. Після проведення тестувань результати показали, що у середньому час на пошук зменшився більш ніж у 3 рази. Сума рангів створеного нами програмного продукту склала 195, а аналогу – 321,5. Рангова сума створюваного програмного продукту менше майже у 2 рази ніж рангова сума аналогу.

Емпіричне значення критерія вийшло 23,5. Критерій значення складає 64. Так як критерій значення більший за емпіричне значення, то різниця між програмними продуктами є суттєвою. Отже, є необхідність у створенні, та використанні даного програмного продукту.

Таблиця 6.1 Таблиця розрахунку ефективності програмного продукту за критерієм Вілкоксона-Манна-Уітні

1 курс (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)	5 курс (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)	Програмісти (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)
1	14	14	30	26	6	7	5,5	25	21,5	11	3	2,5	10	10

Продовження таблиці 6.1

1 курс (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)	5 курс (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)	Програмісти (№)	Час, хв (мд)	Ранг (мд)	Час, хв (ан)	Ранг (ан)
2	17	17,5	33	28	7	14	14	29	24,5	12	2	1	8	7,5
3	17	17,5	38	29	8	8	7,5	31	27	13	5	4	15	16
4	21	20	41	30	9	11	11	27	23	14	7	5,5	18	19
5	13	12	29	24,5	10	9	9	25	21,5	15	3	2,5	14	14

ВИСНОВКИ

Дана кваліфікаційна робота присвячена аналізу існуючих рішень, розробці алгоритмів за обраної темою, проектуванню системи та баз даних, та тестуванню програмного продукту, за темою «Програмний продукт для пошуку проблемних місць у програмних продуктах за допомогою графічного інтерфейсу».

На початковому етапі була визначена мета та актуальність роботи, проведений опис системи та аналіз існуючих аналогів. Основний з яких – Graylog2 описано більш детально. У порівнянні зі створюваним програмним продуктом час на пошук зменшується в середньому у п’ять разів, а тому і процес усунення помилок буде легшим та швидшим. Раніше на пошук необхідних логів потрібно було витратити 20 хвилин. Зараз за допомогою даного програмного продукту, це можна виконувати і за 2 хвилини, тобто швидше майже в 10 разів.

В роботі було визначено та описано варіанти використання. Під час визначення яких була створена та описана Use Case діаграма, на якій можна побачити весь основний функціонал програмного продукту. Також було визначено нефункціональні вимоги, яким повинен відповідати створений програмний продукт.

На наступному етапі виконувався опис послідовності дій користувача і реакції системи на ці дії. У роботі відображено 4 основні послідовності. Усі вони відображені за допомогою діаграм послідовностей та описані словесно.

Першим відображено послідовність дій користувача при першому використанні програмного продукту, тобто сценарій за яким користувач повинен діяти при першому запуску програмного продукту, або у подальшому коли необхідно буде додавати нові сервери.

Другим відображено основний сенс програмного продукту, а отже як користувачеві необхідно отримати логи з вже відомих серверів, та відсортувати і відфільтрувати їх.

Третій та четвертий показують додаткові сценарії системи, які відображають зміну, та відновлення паролю.

Під час проектування програмного продукту створена діаграма класів, де зазначено основні, найнеобхідніші методи та атрибути. Описано кожну частину програми, що вона в себе включає, та для чого буде використовуватися. Також на цьому етапі виконано опис інтерфейсу користувача у порівнянні з прототипами, які були створені за допомогою спеціального інструменту – Wireframe. Також відображено структуру програмного продукту.

Наступним етапом розроблена та описана база даних. Визначені усі необхідні таблиці, та описано, що вони у собі зберігають. Також створена схема бази даних, з включенням усіх використаних таблиць, списків, та атрибутів – усі з яких описані.

Тестування виконувалося за допомогою бібліотеки JUnit. Також відображено покриття тестами, що за відсотками склало 80% протестованих бекенд класів, що собою покриває весь основний функціонал програмного продукту.

На цьому ж етапі визначена ефективність програмного продукту за критерієм Вілкоксона-Манна-Уітні. Емпіричне значення критерія склало 23,5, а критерій значення – 64. Емпіричне значення менше майже у три рази, що показує необхідність у створенні даного програмного продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лещенко К.О. , Пригожев О.С. «Перегляд логів у реальному часі» Матеріали Одинадцятої Міжнародної наукової конференції студентів та молодих учених «Сучасні інформаційні технології - 2021» «Modern Information Technology - 2021» (13-14 травня 2021 р., м.Одеса) / МОН України; Державний університет «Одеська політехніка»; Ін-т комп'ют. систем. – Одеса : Наука і техніка, 2021. – 82 с.

2. Лещенко К.О., Пригожев О.С., Лещенко О.І. «Програмний продукт для пошуку проблемних місць у програмних засобах за допомогою графічного інтерфейсу» Матеріали Одинадцятої Міжнародної науково-практичної конференції «Технічне регулювання, метрологія, інформаційні та транспортні технології» «Technical regulation, metrology, information and transport technologies» (22 - 23 жовтня 2021 р., м.Одеса) / МОН України; Державний університет інтелектуальних технологій і зв'язку. – Одеса, 2021. – 134 с.

3. Khlevna, Iu. L. & Koval, B. S."Development of the Automated Fraud Detection System Concept in Payment Systems". Applied Aspects of Information Technology. Publ. Nauka i Tekhnika. Odessa: Ukraine. 2021; Vol.4 No.1: 37–46. DOI: https://doi.org/10.15276/aait.01.2021.3

4. Logstash. Сайт розробника. URL: https://www.elastic.co/guide/en/logstash/current/introduction.html (дата звернення 12.09.2021).

5. Fluentd. Сайт розробника. URL: https://www.fluentd.org/architecture (дата звернення 8.09.2021).

6. Logstash Reference [7.15]. Logstash Introduction Apache Flume. URL: https://en.wikipedia.org/wiki/Apache_Flume (дата звернення 5.09.2021).

7. Octopussy. Сайт розробника. URL: https://www.octopussy.pm/ (дата звернення 1.09.2021).

8. Стаття опису Graylog2. URL: https://habr.com/ru/post/132116/ (дата звернення 10.09.2021).

9. LOGalyze. Сайт з репозиторієм програмного коду. URL: https://github.com/iNavFlight/inav/wiki (дата звернення 10.10.2021).

10. LogPacker. Сайт розробника. URL: https://medium.com/@LogPacker/logpacker–new–log–management–platform–18c66704ca6b (дата звернення 12.09.2021).

11. Вікіпедія хостингу Logwatch. URL: https://wiki.archlinux.org/title/Logwatch (дата звернення 3.09.2021).

12. Syslog–ng. Сайт розробника. URL: https://www.syslog–ng.com/ (дата звернення 6.09.2021).

13. Inav. Сайт з репозиторієм програмного коду. URL: https://github.com/iNavFlight/inav/wiki (дата звернення 12.09.2021).

14. Liubchenko, V., Komleva, N., Zinovatna, S. & Pysarenko, K. “Framework for Systematization of Data Science Methods”. Applied Aspects of Information Technology. Publ. Nauka i Tekhnika. Odessa: Ukraine. 2021; Vol.4 No.1: 80–90. DOI: https://doi.org/10.15276/aait.01.2021.7 (дата звернення 10.09.2021).

15. JWT — как безопасный способ аутентификации и передачи данных. Павел Кочетов. URL: https://vc.ru/dev/106534-jwt-kak-bezopasnyy-sposob-autentifikacii-i-peredachi-dannyh (дата звернення 10.11.2021).

16. Виталий Сороко, Гродно, belarus. “Использование архитектуры EAV в Opensource–проектах”. URL: https://designpatternsphp.readthedocs.io/ru/latest/More/EAV/README.html (дата звернення 27.09.2021).

17. Сайт з описом відмінностей між версіями бібліотеки Junit. URL: http://java-online.ru/blog-junit.xhtml (дата звернення 31.10.2021).

Додаток А. ЛІСТИНГ ПРОГРАМИ