ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи на ступінь
вищої освіти магістр

на тему: «**РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ СТАНУ МЕРЕЖІ НА ОСНОВІ ТЕОРІЇ СТІЙКОСТІ»**

Виконав: студентка 6 курсу, групи ПДМ–61 спеціальності

121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Чепур М.К.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

ЧЕПУР МАРИНА КОСТЯНТИНІВНА
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка геоінформаційної системи моніторингу стану мережі на основі теорії стійкості»

Керівник роботи: Жебка В.В., д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом закладу вищої освіти від «11» жовтня 2021 року №170.

2. Строк подання студентом роботи

3. Вхідні дані до роботи

Геоінформаційні системи моніторингу мережі
Теорія стійкості;
Теорія графів;
Науково-технічна література з питань, пов'язаних з функціональною стійкістю мережі;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1 Системи моніторингу мережі
4.2 Вимоги та оцінка якості системи.
4.3 Опис проектування системи.
4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Існуючі системи моніторингу стану мережі
2. Відмовостійкість мережі
3. Формули для розрахунку зв’язності мережі
4. Граф мережі
5. Основний програмний модуль
6. Висновки

6. Дата видачі завдання_

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		Виконано
2	Вимоги до системи		Виконано
3	Дослідження аналогів систем моніторингу стану		Виконано
4	Аналіз математичної теорії		Виконано
5	Розробка геоінформаційної системи моніторингу мережі		Виконано
6	Вступ, висновки, реферат		Виконано
7	Розробка обов‘язкових демонстраційних матеріалів		Виконано
8	Попередній захист роботи		
9	Здача роботи		

Студент _____ Чепур М.К.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 69 с., 50 рис., 24 джерел.

ГЕОІНФОРМАЦІЙНА СИСТЕМА, МОНІТОРИНГ МЕРЕЖ, ІНФОРМАЦІЙНА МЕРЕЖА, СТІЙКІСТЬ МЕРЕЖІ, ТЕОРІЯ ГРАФІВ, НАДІЙНІСТЬ МЕРЕЖІ, ВІДМОВОСТІЙКІСТЬ, PYTHON

Мета дослідження: розробка геоінформаційної системи для моніторингу стану мережі на основі теорії стійкості.

Об'єкт дослідження: моніторинг стану мережі.

Предмет дослідження: геоінформаційна система моніторингу стану мережі.

Методи дослідження: теорія стійкості мережі, теорія графів, функціональна стійкість мережі.

У роботі проведено аналіз існуючих систем моніторингу мереж, який показав, що аналоги не відповідають заданим в магістерській роботі вимогам, тому розробка даної системи моніторингу є актуальної для роботи з інформаційними системами.

Основною проблемою даних аналогів є те, що вони не підтримують оновлення відображення системи в реальному часі, не підтримують автоматичного заповнення карти мережі, мають не зовсім зручний інтерфейс користувача та можливість вирішення поломки переводу подачі даних на інші вузли.

Реалізація системи була зроблена за допомогою високо-рівневої мови програмування Python, яку зручно використовувати для реалізації математичних задач. За допомогою Django фреймворку було створено веб-систему з необхідним функціоналом, яку можна відкрити на будь-якій платформі.

Клієнтське відображення мережі було створено за допомогою технологій веб-розробки, основним інструментом була використана мова програмування JavaScript, інтерфейс карти було описано на базі бібліотеки Leaflet.

Розроблена геоінформаційна система моніторингу мережі дозволяє відстежувати стабільність функціонування мережі та в разі виникнення поломки на вузлів мережі, повідомляє адміністратора мережі та робить перенаправлення передачі даних на інших робочий вузол.

Дану систему можна використовувати в сфері інформаційних та телекомунікаційних мереж для повідомлення про помилки та їх усунення. В подальшому систему можна модернізувати додавши використання штучного інтелекту, для того аби можливі помилки можна було передбачити до моменту їх виникнення.

Галузь використання – інформаційні мережі.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПК – персональний комп’ютер

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ГІС – геоінформаційна система

IP – internet protocol

SNMP – Simple Network Management Protocol

СКБД – система керування базами даних

MVC - Model-view-controller

ORM – Object-relational mapping

MTV – Model-template-view

ІС – інтегроване середовище

РІС – розгалуджена інформаційна система

ВСТУП

Мета дослідження: розробка геоінформаційної системи для моніторингу стану мережі на основі теорії стійкості.

Об'єкт дослідження: моніторинг стану мережі.

Предмет дослідження: геоінформаційна система моніторингу стану мережі.

Методи дослідження: теорія стійкості мережі, теорія графів, функціональна стійкість мережі.

В сучасному світі застосування телекомунікаційних та комп'ютерних мереж стрімко росте, в ході чого виникає все більше проблем з підтримкою стабільної роботи цих систем. Системи моніторингу мереж відстежує стан мережі, роботу пристроїв, робочих станцій та серверів, та повідомляє адміністраторів цих мереж про помилки, які могли виникнути в мережі.

Архітектура мережі складається із сукупності різноманітного обладнання та необхідного ПЗ, яке забезпечує передачу даних в середині мережі та підтримує стабільну роботу всієї системи. ПЗ моніторингу мережі повинно підтримувати роботу всієї системи та відстежувати будь-які простої в мережі.

Простої можуть бути викликані різними факторами, які незалежать від користувача, природніми або технічними. Неможливо передбачити всі можливі збої вручну, тому для цього використовують системи, які дозволяють автоматизувати даний процес та покращити майбутню роботу системи.

Для того аби забезпечити стійкість мережі до різноманітних збоїв, необхідно використовувати таку систему, яка дозволяє розподіляти ресурси всередині мережі в критичних ситуаціях та не дасть системі повністю вийти з ладу, натомість знайде необхідні обхідні шляхи для передачі даних по резервним маршрутам.

Системи моніторингу дозволяють відображати мережі у вигляді схем, також збирають, накопичують, зберігають, надсилають, перетворюють та оброблюють інформацію про мережу.

За допомогою накопичення даних про шляхи в мережі та збір інформації про всі минулі збої, в разі неполадки, система зможе залишатися відмовостійкою.

Найважливішою умовою роботи мережі є забезпечення її продуктивності в разі будь-яких непередбачуваних збоїв, як в середині так і зовні мережі.

Магістерська робота описує всі елементи функціональної стійкості мережі – надійність, розширюваність, керованість, живучість, продуктивність. Даний термін описує систему, яка здатна певний час виконувати всі необхідні функції, в критичних для роботи умовах.

Найкращим рішенням для інформаційної мережі є геоінформаційна система у вигляді додатку, яка забезпечує збір та аналіз географічних та просторових даних та дозволяє адміністратору системи робити інтерактивні запити, оброблювати просторові дані, відображати та аналізувати вихідну інформацію у вигляді карт.

На даний момент існують різні системи моніторингу з підтримкою відображення даних у вигляді геоінформаційної системи. Такі інструменти дають купу можливостей візуалізації даних у різних виглядах: діаграми, схеми, графіки, але мають певний комплекс недоліків, які не дають повного спектру можливостей, необхідних для максимальної ефективності роботи моніторингу системи.

Структура схеми системи моніторингу будується на основі теорії графів. В якості графа виступає неорієнтований зв’язний граф, який має певну вагу, я кості якої виступають коефіцієнти готовності кожного ребра.

Ці дані в подальшому використовуються для обчислення коректності роботи мережі, при виході з роботи певно ланцюгу мережі, передача даних передача даних перенаправляється на резервний ланцюг шляху.

В якості інструментів для розробки ГІС моніторингу мережі було обрано мову програмування Python та веб-фреймворк Django, реалізація відображення карти на сторінці створена за допомогою мови програмування JavaScript та бібліотеки на її базі – Leaflet, що дозволяє визначити всі необхідні елементи мережі в якості міток на мапі та задати їм необхідні шляхи, якими зв’язана мережа.

Розроблювана геоінформаційна система моніторингу повинна відповідати критеріям описаним нижче:

- оновлення даних мережі в реальному часі, без додаткової участі користувача;
- автоматична побудова графу мережевої карти, після вводу даних в систему (підтягування цих даних з БД);
- зручний інтерфейс користувача, який дозволить просто керувати необхідними функціями системи;
- кросплатформеність – підтримка системи моніторингу максимально можливої видів ОС;
- широкі можливості візуалізації графу мережі та даних про мережу.

1. ДОСЛІДЖЕННЯ АНАЛОГІВ СИСТЕМ МОНІТОРИНГУ СТАНУ МЕРЕЖІ

Розвиток сучасних телекомунікаційних та інформаційних технологій стрімко зростає, завдяки використанню глобальної мережі Інтернет, мобільного зв'язку, геоінформаційних систем. Це дозволяє інформації швидко та вчасно доходити від початкого користувача до кінцевого та полегшую. Також це поширює використання та різноманіття геоінформаційних систем, які допомагають.

В даній роботі мова йде про ГІС моніторингу, за допомогою яких можна відобразити стан мережі та відстежувати наявні в ній пристрої та з'єднання.

Найпримітивнішою програмою для моніторингу мережі є ping – системна утиліта, яка призначена для перевірки якості з'єднання та цілісності в мережах, основою якої є TCP/IP. Додаток фіксує отримані відповіді від вузла мережі, після того як надсилає запити протоколу вказаному вузлу.

Із безкоштовних програм представлених на сьогодняшній час, які дозволяють здійснювати моніторинг мережі представлені такі Cacti, Nagios, Icinga, LanState, Nedi, Ntop, Zabbix, Observium, TheDude.

1.1 Додаток для моніторингу та відображення графіків Cacti

Cacti – сучасний веб-додаток з відкритим кодом для моніторингу всередині мережі, який дозволяє створювати графіки. Інтерфейс дозволяє роботу с декількома користувачами, зі збереженням для кожного повноцінних функцій. Також його використовують постачальники веб-хостингу в якості віддаленого серверу або як віртуальний приватний сервер.

Можливості, які є у модуля Cacti:

- вбудована в систему підтримка SNMP;
- можливість відображення даних у вигляді дерева або списку;
- забезпечення автоматизації пристроїв;

- вбудовані шаблони графіків, пристроїв, джерел даних;
- великий вибір елементів графіка;
- вбудоване автоматичне заповнення даних для графіків;
- можливість переглянути аналіз продуктивності мережі;
- можливість використовувати сценарії написаних на Perl або PHP;
- спеціальні алгоритму для збору даних за нестандартний проміжок часу.

Cacti – веб-додаток, дозволяє моніторити стан серверів, дозволяє відстежувати як стан відвідувачів, так і стан ядер процесора. Як видно на рис. 1.1 всі ресурси відображаються у вигляді графіків. Вони прорисовуються через час, який користувач вказує в налаштуваннях.

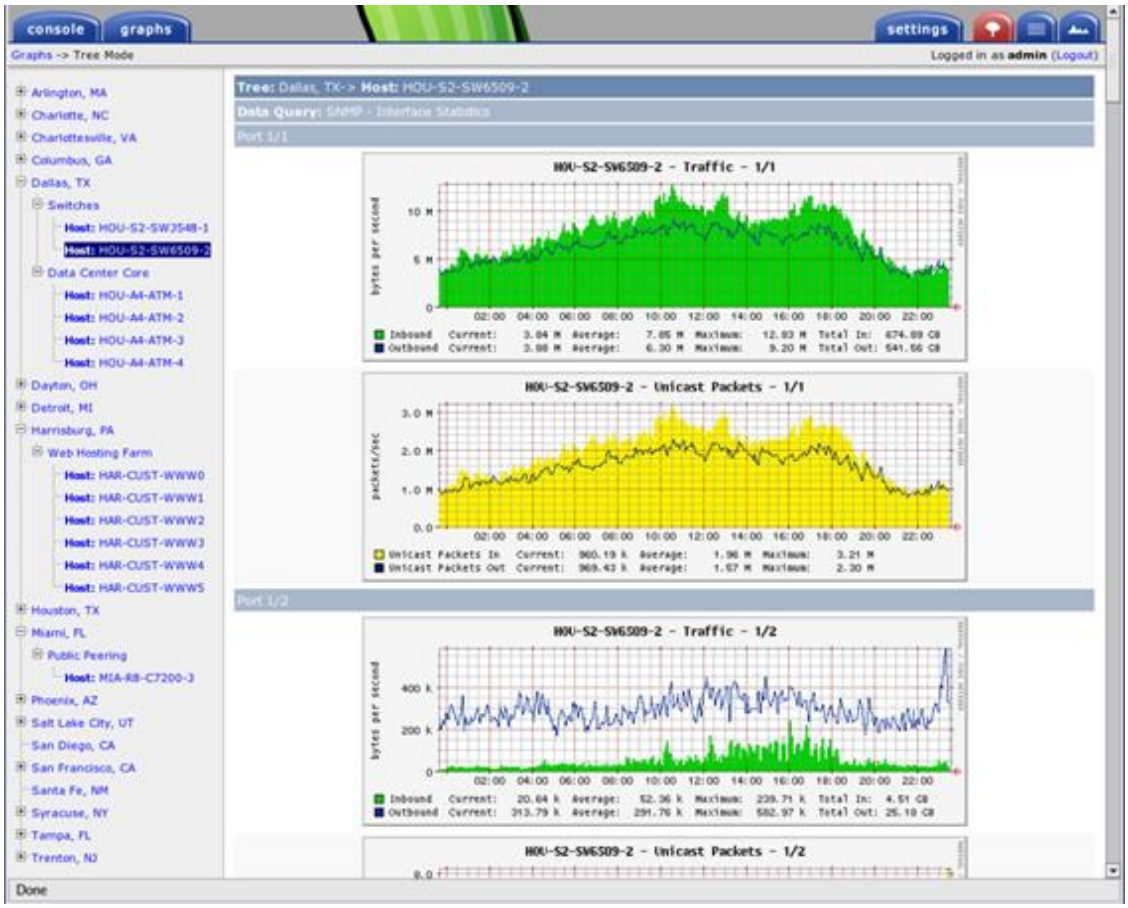


Рисунок 1.1 – Інтерфейс веб-додатку Cacti

1.2 Програмна система для моніторингу Nagias

Nagios – програмне забезпечення для ПК з відкритим кодом, написаний на мові С, що дозволяє моніторити роботи серверів та сповіщати користувачів про можливі неполадки та коли проблема вирішується. Спочатку додаток був розроблений для ОС Linux або будь-якого ОС на базі UNIX.

Система оповіщень у Nagios налаштовувана, через електронну пошту або смс-повідомлення. Присутня підтримка відображення всіх функцій пристроїв в мережі, з кольоровим кодуванням, що вказує на проблеми в мережі. Основним недоліком Nagios є те, що всі конфігурації налаштувань необхідно встановлювати через командну строку, що потребує довшого навчання нових адміністраторів.

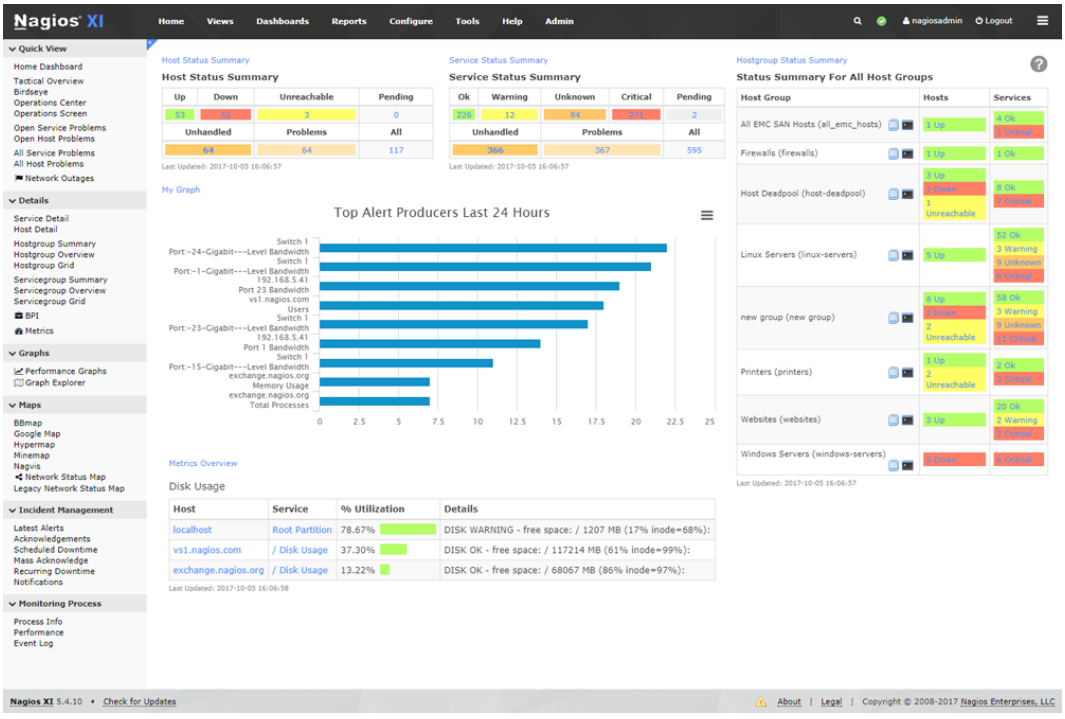


Рисунок 1.2 – Інтерфейс додатку на ПК Nagios

Після встановлення exe-додатку на ПК та авторизації. За замовчуванням, в налаштуваннях додатку встановлено моніторинг сервісів локального сервера, в

розділі Services, як представлено на рис 1.2. В Nagios встановлено купа утиліт для моніторингу стандартних системних серверів.

Набір функцій, які підтримує Nagios:

- виконує моніторинг всіх ресурсів хвоста;
- простий та зрозумілий інтерфейс;
- можливість написання скрипта, для перевірки служб мережі, використовуючи C++, Python, PHP, C#, Ruby, Perl;
- можливість віддаленого зашифрованого моніторингу через SSH SSL;
- можливість відображення графіків даних;
- підтримка Push-повідомлень;
- відображення історій та файлів журналів.

1.3 Вільна система моніторингу Icinga

Icinga – система моніторингу комп'ютерних мереж, яка є розгалудженням від системи Nagios, на відміну від Nagios Icigna є меньше важкою та більш продуктивною та зручною, в ній присутня багатопотоковість та модульна архітектура. Існує декілька видів веб-додатку Icinga, але головна відмінність від додатку Nagios є налаштування додатку, яка може бути відображена у вигляді веб-інтерфейсу, що є більш зручним на відміну від командної строки. Систему можна використовувати з купою програмних пакетів графічного відображення та моніторингу, наприклад PNP4Nagios, inGraph, Graphite, забезпечивши найкраще відображення мережі.

Інтерфейс Icinga, написаний на мові програмування Python, дозволяє переглянути порти серверів, які є в мережі, представлений на рис. 1.3. Система оптимізована на створення кількох агентів для моніторингу, вони направляють отримані результати перевірки на головний вузол мережі.

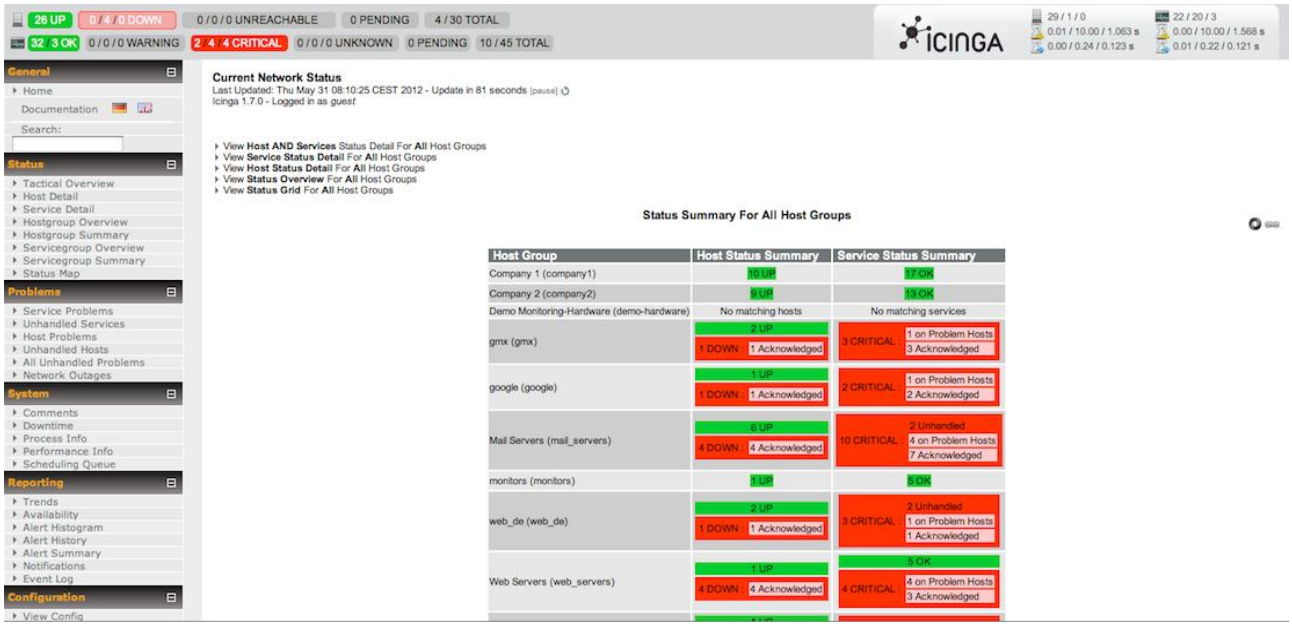


Рисунок 1.3 - Вільна система моніторингу Icinga

1.4 Система моніторингу та адміністрування мережі LANState

LANState – система ПЗ призначена для адміністрування та моніторингу мережі. За допомогою неї користувачі та адміністратори можуть спостерігати як змінюється стан мережі у вигляді графіка, також моніторити віддалені пристрої з головного ПК та керувати робочими станціями і серверами.

Особливості системи LANState:

- Підтримка можливості управлянні робочими станціями по мережі;
- Можливий доступ до всіх віддалених серверів та комп’ютерів;
- Інтерфейс представлений у вигляді веб-додатку;
- Автоматичне знаходження нових пристроїв (комп’ютерів) в мережі;
- Підтримка Push-сповіщень;
- Підтримка паралельної роботи з декількома картами;
- Не потребує встановлення ніяких компонентів на комп’ютери та сервери.

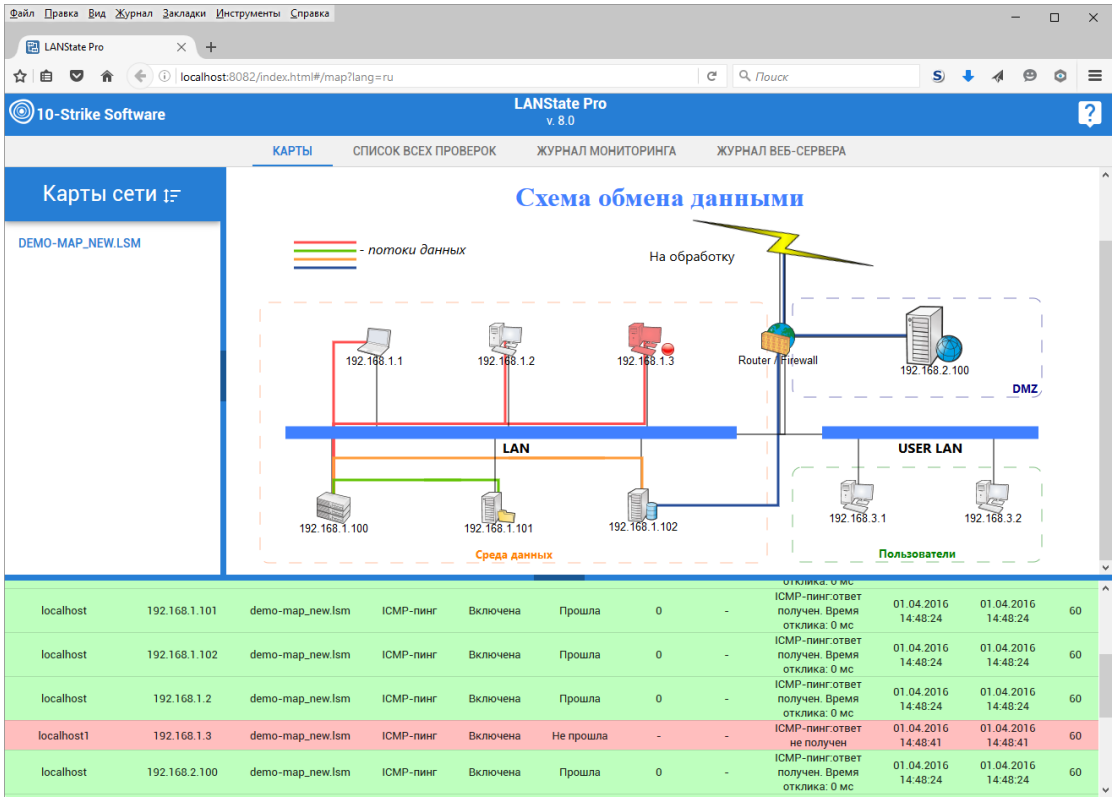


Рисунок 1.4 – Інтерфейс системи моніторингу LANState

Інтерфейс системи LANState зрозумілий та зручний, що дозволяє переглянути карту мережі, у вигляді схеми та дає доступ до всіх станцій мережі. Приклад інтерфейсу ПЗ представлено на рис. 1.4. Моніторинг всіх віддалених пристроїв мережі відбувається за допомогою опитування ПК, шо відбувається періодично.

1.5 Додаток для перегляду мережевої інфраструктури Nedi

Nedi – безкоштовне ПЗ, яке дозволяє переглядати інфраструктуру мережі у вигляді карти та відстежує кінцеві вузли, які були підключені та інвентеризує мережеві робочі станції.

ПО збирає всю отриману від вузлів інформацію (серійні номери, версії ПО та прошивок, часові параметри, налаштування модулів) з кожної робочої станції. Додаток використовує Cisco Discovery Protocol або Layer Discovery Protocol, щоб

знайти нові комунікатори і маршрутизатори, після чого підключається і збирає інформацію з них.

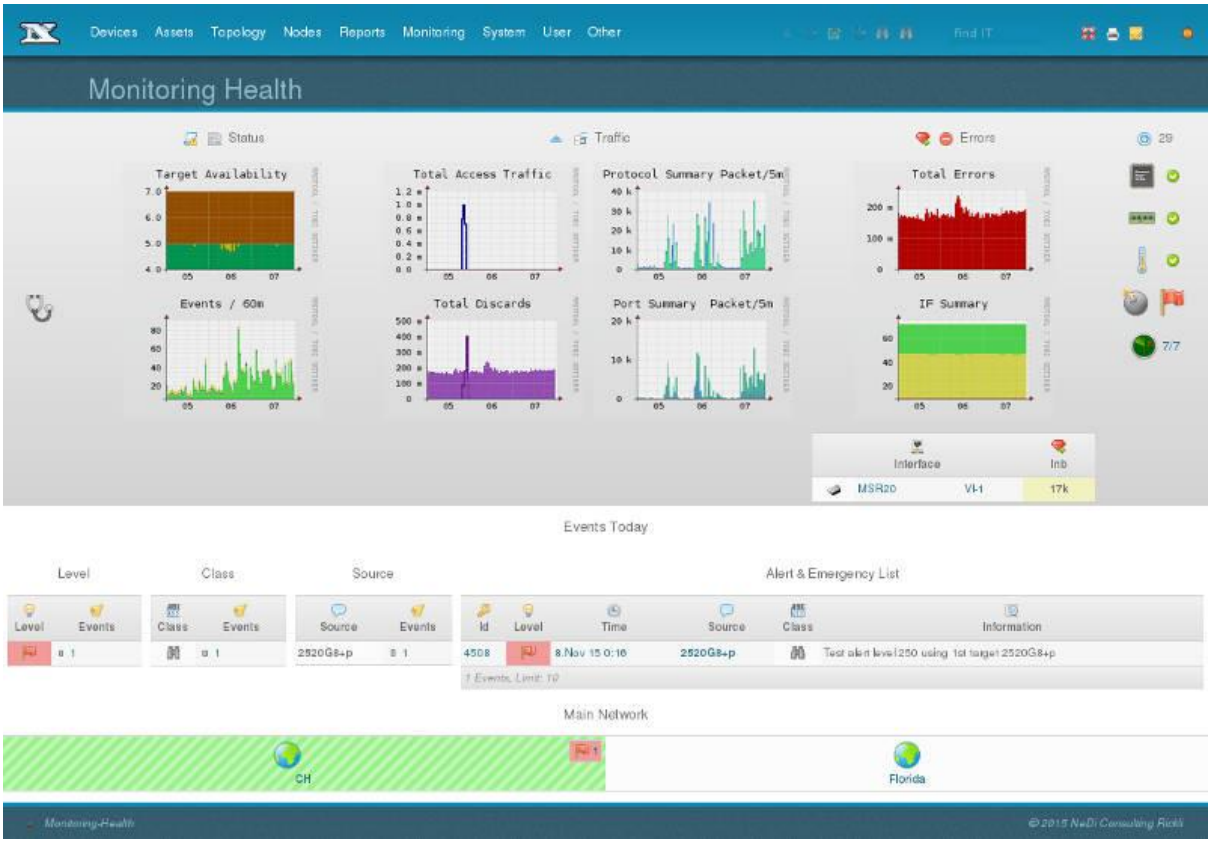


Рисунок 1.5 – Інтерфейс програми Nedi

Nedi підтримує безліч функцій у зрозумілому графічному інтерфейсі, що дозволяє керувати корпоративними мережами. Приклад інтерфейсу програми відображенно на рис 1.5. Також є підтримка можливості зіставлення фізичних MAC-адрес, відстеження змін у зібраних даних, відрисовка мережевих карт, можливість відображення помилок підключення, звітностей, використання комутаторів та маршуртизаторів. Через інтерфейс браузера можна робити пошук, щоб визначити комутатор, відносячи кожне обладнання до певної категорії, відповідно до локальної бази даних.

Особливості додатку Nedi:

- можливість керування та виявлення мережі;
- збереження резервних конфігурацій;
- IT reports;
- можливість керування принтером;
- відображення топології мережі;
- можливість проведення аналізу на основі Netflow та sFlow.

1.6 Комп'ютерне ПЗ Ntop

Ntop – безкоштовне ПЗ, яке дозволяє досліджувати комп'ютерну мережу, з набором інструментів, що дозволяють аналізувати пакети даних. Доступний для використання як для ПК так і в веб-вигляді, через HTML-сторінку. Режим використання консольний, тобто стан мережі відображається на терміналі.

Також за допомогою даного ПЗ можна розширювати можливості, використовуючи скриптову мову програмування Lua, так як Ntop включає в себе API-інтерфейс для цієї мови.

Інформація про потік даних, який проходить через мережу, і про з'єднання з нею доступні в додатку в режимі онлайн. Дана інформація відображається у вигляді графіків, таблиць та діаграм та зберігається у додатку для створення статистики.

Одною з переваг Ntop є те, що від має можливість контролю трафіка в мережі, в конкретно заданому місці. Якщо в мережі виникає проблема, то можна отримати часовий звіт про місце, де виникла проблема, та виявити місце яке відповідальне за цю помилку.

На рис. 1.6 представлено відображення даних у вигляді таблиць в програмі для моніторингу Ntop.

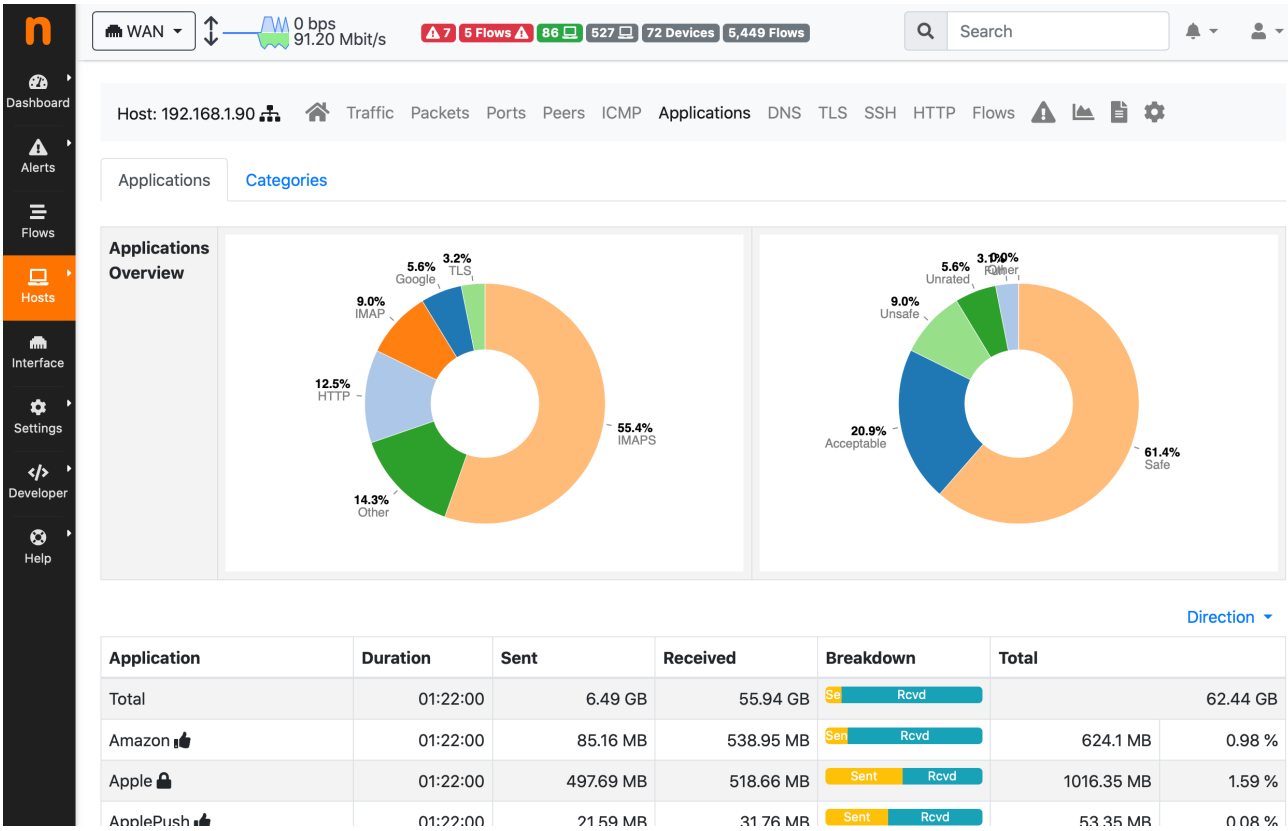


Рисунок 1.6 – Інтерфейс додатку Ntop

1.7 Вільна система для моніторингу Zabbix

Zabbix – повномасштабна система мережевого і системного моніторингу мережі, виконаний у вигляді веб-консолі. Архітектура системи складається із трьох основних частин:

- сервера (використовується для погодження виконання перевірок, складання запитів та формування статистики);
- агентів (утиліти, необхідні для виконання перевірок на стороні зовнішніх вузлів мережі);
- фронтенда (необхідний для конфігурації керування системою).

Аби розподілити навантаження на систему та зняти її з головного сервера, можливе розгортання групи проксі-серверів, котрі збирають дані про перевірку серії хостів. Мовою написання сервера та агентів системи моніторингу є C, а для реалізації фронтендової частини використано мову PHP. В якості БД можна використовувати наступні СКБД: СКБД MySQL, PostgreSQL, SQLite, DB2 і Oracle. Відображення логічної структури мережі можливе у вигляді карти, яку необхідно створювати в ручну, що є не зовсім зручно.

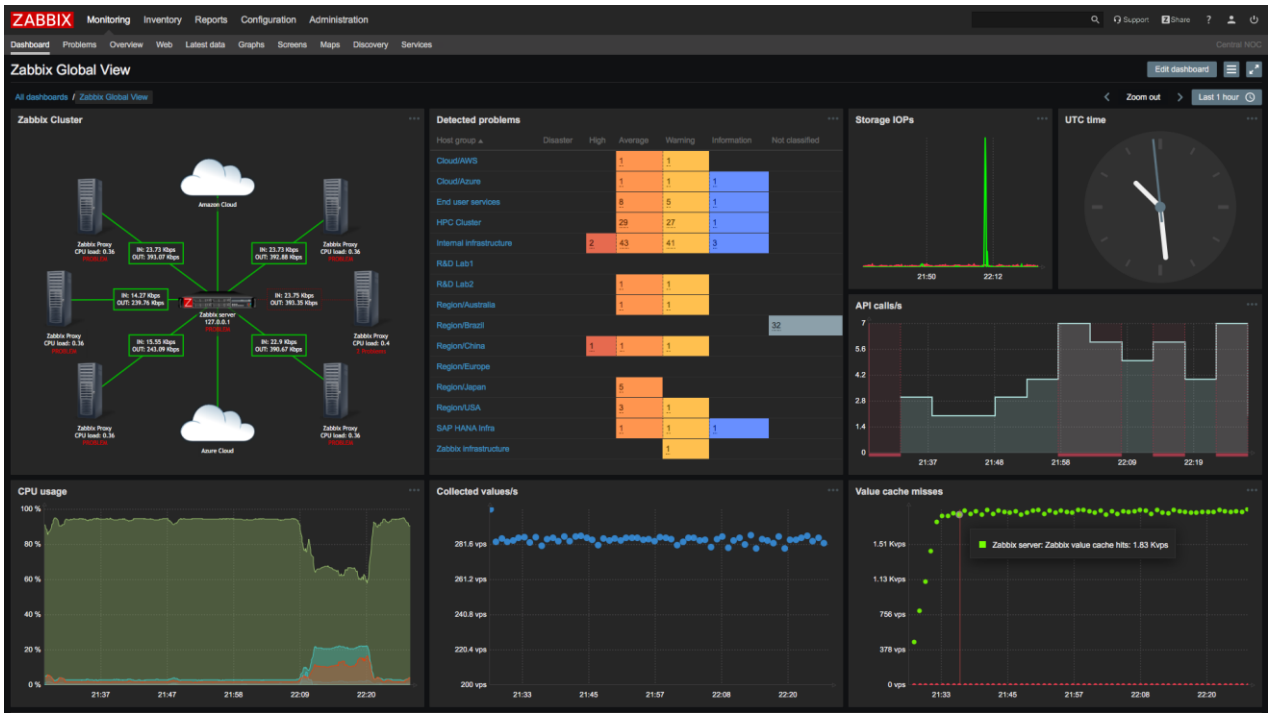


Рисунок 1.7 – Інтерфейс системи моніторингу Zabbix

На рис 1.7 представлено інтерфейс системи Zabbix, з відображенням даних у різних виглядах: діаграми, карта, схема. Відображення даних зручно редагувати та налаштовувати під необхідний користувачу вигляд.

Можливості, які представлені у системі моніторингу мережі Zabbix:

- підтримка автоматичного виявлення проблем;
- влаштований SLA моніторинг;

- можливість створювати карти мережі;
- влаштовані високопродуктивні агенти;
- влаштований веб-інтерфейс для налаштування та адміністрування;
- підтримка сценаріїв моніторингу;
- підтримка SNMP;
- автоматичне вивленння і різноманітні шаблони;
- влаштована система шаблонів.

Система доступна у якості віртуального пристрою для роботи декількох моніторів віртуальних машин.

1.8 Додаток моніторингу мережевого обладнання Observium

Observium – додаток для моніторингу, який пропонує як платну версію, так і безкоштовну, використовує протокол SNMP. Легко налаштовується та використовується лише за допомогою встановлення таких додатків як Apache, Php, MySQL та створення БД.

Налаштування мережі можна здійснювати як тільки відкриваєте додаток, після входу в графічний інтерфейс, задавши в систему хости, мережі та діапазони для автоматизації виявлення і даних SNMP для досліджування мережі. Після додавання пристроя в мережу, весь життєвий цикл буде автоматично зберігатися та відображатися в додатку.

Observium працює також як віртуальна машина, тому може бути основним інструментом отримання інформації про мережу. Додаток дозволяє отримати дані, в вигляді діаграм і графіків пропускної здатності мережі, наявної кількості IP-пакетів та фрагментації, як представлено на рис. 1.8. Також додаток відображає інформацію про оперативну пам’ять мережі, стан центрального процесора, журналу події або сховища даних.

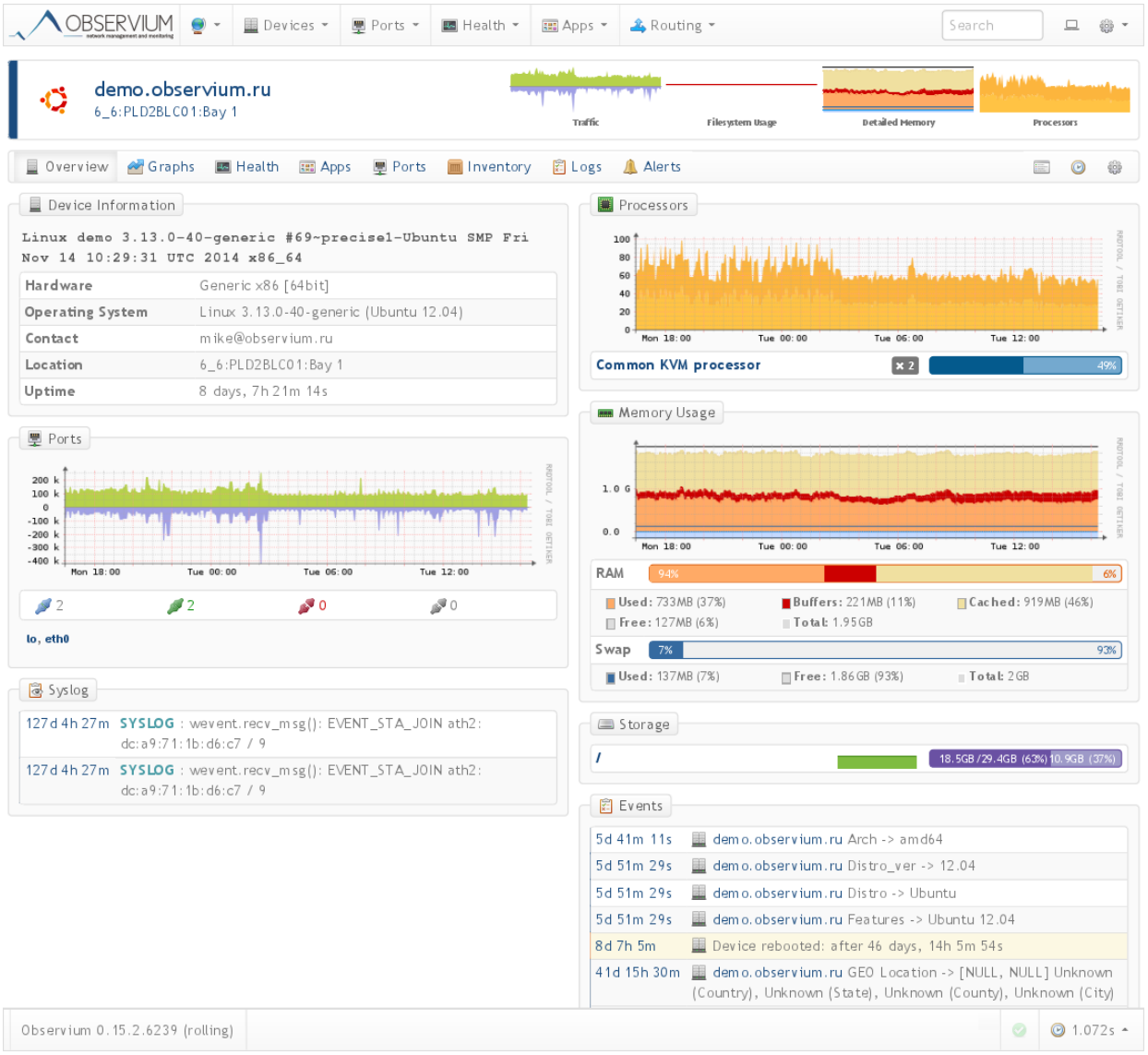


Рисунок 1.8 – Інтерфейс додатку Observium

1.9 Система моніторингу TheDude

Система моніторингу мережі TheDude дозволяє автоматично виявити мережу, намалювати відображення її у вигляді карти мережі, також попередити користувача при виникнені якоїсь проблеми в якійсь службі.

Можливості системи TheDude:

- Push-сповіщення;

- Підтримка прорисовки власної карти;
- Можливість додавання пристроїв в мережу;
- Віддалення керування пристроями;
- Підтримка комбінації віддалений сервер та локальний клієнт;
- Кросплатформеність.

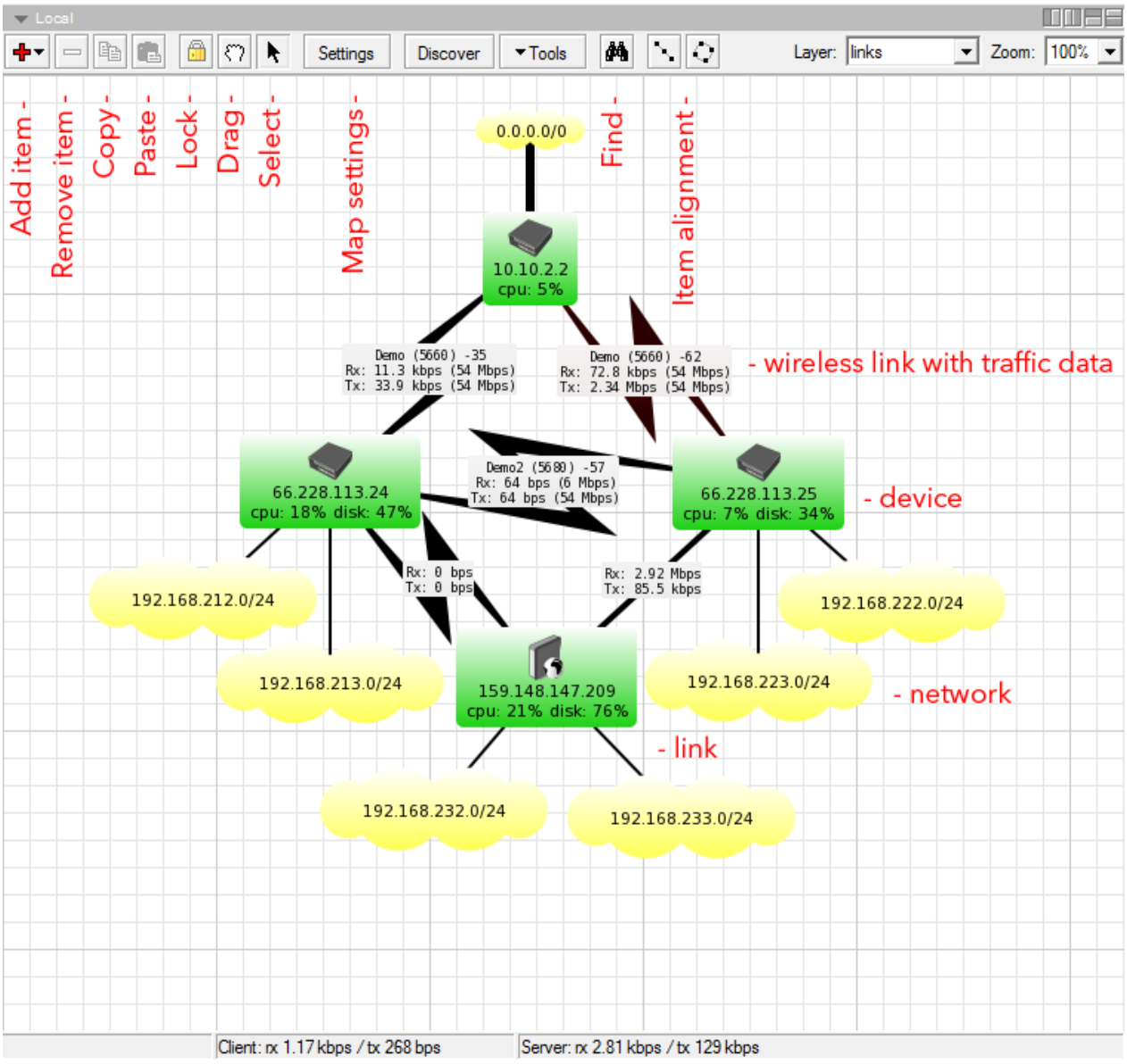


Рисунок 1.9 – Інтерфейс додатку TheDude

Система реалізована у вигляді desktop-додатку, для зручного використання, але реалізовано його не зовсім зручно для користувача. Також необхідно зазначити, що дана система створена для моніторингу пристроїв корпорації Mikrotic, якою і була створена, тож офіційно не заявлена робота з іншими пристроями.

1.10 Порівнювальна характеристика аналогів

В даному розділі було описано дев’ять систем моніторингу мережі, у кожного з яких є свої плюси та свої мінуси. В контексті магістерської роботи було виділено наступні критерії, які необхідні бути присутні в розроблюваному додатку. Порівняння цієї системи з аналогами представлено в таблиці 1.1. Умовне позначення розроблюваної системи – gisEnergo.

Таблиця 1.1 – Порівняння систем моніторингу

	Оновлення даних в реальному часі	Автоматична побудова графу мережевої карти	Зручний інтерфейс	Кросплатформеність	Широкі можливості відображення даних
gisEnergo	+	+	+	+	+
Cacti	-	+	+/-	+	+
Nagios	-	-	-	+	+
Icinga	+	-	+	+	+
LanState	+	+	+	-	-
Nedi	-	+	+	-	+

Продовження таблиці 1.1 – Порівняння систем моніторингу

Ntop	-	+	+	+	+
Zabbix	-	-	+	-	+
Observium	+	-	+	+	+
TheDude	+	+	-	-	+

В ході розгляду існуючих аналогів, розглянуто системи моніторингу мереж з візуальним представленням даних у вигляді карт, досліджено їхню функціональність, переваги та недоліки.

Додаток для моніторингу та відображення графіків Cacti не підтримує функцію оновлення даних в реальному чаті, але може автоматично побудувати карту на основі введених даних, інтерфейс додатку є плюс/мінус зручним, але не для всіх користувачів, також його можна запускати на різних платформах таких як Windows, Linux та інших UNIX-подібних системах, відображення даних доступне у різноманітних виглядах: карти, діаграми, схеми.

Програмна система для моніторингу Nagias не відповідає більшості критеріям, не підтримується оновлення даних в реальному часі, немає зручного користувацького інтерфейсу, та автоматичної побудови графу мережевої карти. Плюсами є підтримка програми на будь-якій платформі та широкий вибір відображення даних.

Вільна система моніторингу Icinga підтримує оновлення даних про мереже в реальному часі, але побудова графу у вигляді мережі в системі недоступна. Також система як і дві програми описані вище є кросплатформеною системою та має широкий вибір відображення даних, що є чудовою можливістю в купи зі зручним користувацьким інтерфейсом.

Система моніторингу та адміністрування мережі LANState на відміну від вище описаних систем моніторингу не підтримується на всіх операційних системах, а

працює виключно на Windows, також немає підтримки різноманітного відображення даних. Але на протиставу цьому система підтримує оновлення даних мережі в режимі онлайн та автоматично будує граф мережі та має інтрефейс для зручного користуванння.

Додаток для перегляду мережевої інфраструктури Nedi працює лише на операційній системі Linux, підтримки інших система немає, також немає підтримки оновлення даних в реальному часі, але при вводі даних про мережу система автоматично будує граф для візуального відображення. За допомогою зручного користувацького відображення можна зручно взаємодіяти з даними, які відображаються у різному графічному вигляді такому як карти, графи, схеми та діаграми та графіки.

Комп'ютерне програмне забезпечення Ntop пропонує користувачам великий вибір операційних систем, зручний користувацький інтерфейс та великий вибір можливостей відображення даних і також автоматизацію побудови мережевої карти у вигляді графу, але суттєвим недоліком є відсутність оновлення даних в реальному часі.

Вільна система для моніторингу Zabbix хоч і є потужною системою, яка має величезну кількість переваг, але її повноцінне використання лише на платформі Linux та UNIX-подібних систем та що немало важливо не підтримує автоматичну побудову мережевої карти і це потрібно робити в ручну, також немає підтримки оновлення даних в режимі онлайн. Із переваг даної системи те, що вона має чудовий користувацький інтерфейс та широкі можливості відображення даних у різних можливих варіантах.

Додаток моніторингу мережевого обладнання Observium відповідає більшості перевагам, які необхідні для роботи поставленої в заданому додатку, таким як зручне користувацьке відображення, підтримка роботи на більшості операційних системах, таким як Windows, Linux та інших UNIX-подібних системах, також відображення даних у різних виглядах необхідних для роботи і автоматичну побудову схеми

мережевої карти. Але недоліком додатку є те, що немає підтримки оновлення даних в реальному часі.

Система моніторингу TheDude не підтримує кросплатфомність, та може працювати лише на системі Windows, немає зручного користувацького інтерфейсу, на противагу цім недолікам має широкі можливості відображення даних, підтримує автомматичну побудову графу у вигляді мережевої карти і оновлення даних працює в режимі реального часу.

В ході магістерської роботи була розроблена геоінформаційна система моніторингу, яка відповідає всім поставленим критеріям і підтримує автоматичну побудову графу мережі у вигляді мережевої карти. Система була розроблена за допомогою веб-фреймворку Django, тобто це веб-додаток, який можна відкрити на будь-якій операційній системі, що дуже зручно для роботи з мережею. Система має широкі можливості відображення мережевих даних. Для побудови карти необхідно лише ввести дані про станції мережі та оновлення даних відбувається в режимі реального часу, що заощаджує час на оновлення та дозволяє вчасно визначити, якщо виникла якась помилка в роботі мережі.

2. ОПИС МАТЕМАТИЧНОЇ МОДЕЛІ

Теорія функціональної стійкості була проаналізована професором Маковим О.А., який розробив стратегію забезпечення роботи складних телекомунікаційних систем. Предметом дослідження професора були розгалуджені інформаційні системи та особливості їх функціонування, тому оптимально застосовувати дану теорію на розробляємій системі. Такі системи складають станцій (вузлів) та каналів для передачі даних між частинами системи, та можуть бути представлені у вигляді графів – складних структур даних, які допомогають представити різні системи у структурованому вигляді.

Функціональна стійкість – поняття стійкості мережі, при якому мережа залишається робочою, тобто може виконувати свої функції в умові різноманітних проблем виниклих через різні причини. В РІС функціональна стійкість складається з набору параметрів таких як надійність, живучість і відмовостійкість, що характеризують систему як працездатну.

2.1 Надійність мережі

Надійність мережі – це властивість мережі виконувати задані функції в тому об’ємі в якому вони були задані в конкретних умовах функціонування. Стійке функціонування мережі неможливе без надійної роботи розподілених систем.

Надійність складається з комплексу властивостей, яке залежить від умов експлуатації системи. Елементи, з яких складається надійність представлені в таб. 11.

Таблиця 2.1 – Властивості надійності

Безвідмовність	Властивість об'єкта безперервно зберігати працездатність протягом заданого інтервалу часу без вимушених перерв.

Продовження таблиці 2.1 – Властивості надійності

Довговічність	Властивість об'єкта зберігати працездатність до настання граничного стану при встановленій системі технічного обслуговування та ремонтів.
Ремонтопридатність	Властивість об'єкта, що полягає у пристосованості до попередження та виявлення причин виникнення його відмов, пошкоджень та усунення їх наслідків шляхом проведення технічного обслуговування та ремонтів.
Відновлюваність	Властивість об'єкта бути пристосованим до попередження та виявлення причин виникнення відмов та їх усунення.
Стійкість	властивість об'єкта бути пристосованим до попередження та виявлення причин виникнення відмов та їх усунення
Режимна керованість	Якість об'єкта підтримувати нормальний режим у вигляді управління, тобто. збереження або відновлення (після порушення) нормального режиму його роботи.
Живучість	Властивість об'єкта протистояти обуренням, не допускаючи їх каскадного розвитку з масовим порушенням харчування споживачів.
Безпека	Властивість об'єкта не допускати ситуацій, небезпечних для людей та навколишнього середовища.

Протиріччям безвідмовності є відмова, існує багато типів відмов, найосновніші з них представлені в таб. 2.2 нижче.

Таблиця 2.2 – Види відмов

Повні	Функціонування неможливе
Катастрофічні	Відбувається раптова повна відмова
Випадкові	Виникаючі у фазі нормальної експлуатації об'єкта внаслідок взаємодії великої кількості незалежних один від одного факторів
Зв’язані зі старінням	відмови в кінці періоду експлуатації внаслідок втоми, зносу, старіння матеріалу
Часткові	за допустимі межі виходить один або кілька параметрів
Внезапні	відбувається стрибкоподібне зниження працездатності об'єкта, при зміні одного або кількох параметрів
Параметричні	визначальний параметр (температура, напруга, опір) безперервно змінюючись у часі, досягає граничних значень
Постепенні	відбувається поступове зниження рівня працездатності при зміні одного або кількох заданих параметрів
Деградаційні	відмови, зумовлені природними процесами старіння, корозії
Залежні	відмови, зумовлені відмовою інших елементів
Стійкі	для відновлення працездатності потрібен ремонт об'єкту
Самоусувні	відмови, що призводять до короткочасного порушення працездатності
Множинні	відмови двох і більше об'єктів або елементів з однієї причини (рідше з різних причин), за яких жодне настання відмови не є наслідком іншого

Продовження таблиці 2.2 – Види відмов

Каскадні	послідовні відмови двох і більше елементів, у яких відмова наступного елемента є наслідком попереднього
Приробіткові	обумовлені недостатньою якістю виробу, що проявляються у початковій фазі його експлуатації
Виробничі	виникають у результаті порушення чи відхилення від встановленого процесу виготовлення чи ремонту
Конструкційні	виникають у результаті порушення чи недосконалості встановлених правил чи норм конструювання та проектування
Явні	виявляються візуально або штатними методами та засобами контролю та діагностування відразу після їх появи
Неявні	не виявляються візуально чи штатними методами, засобами контролю та діагностики, для виявлення яких потрібні спеціальні методи чи заходи
Ринкові	виникають внаслідок зриву планових або договірних зобов'язань суб'єктів електроенергетичного ринку
Ресурсні	відмови за довговічністю, внаслідок яких об'єкт досягає граничного стану
Незалежні	не обумовлені відмовими інших об'єктів
Нестійкі	при яких для відновлення працездатності потрібне лише відключення (наприклад, успішне АПВ)
Експлуатаційні	виникають внаслідок порушення встановлених правил та (або) умов експлуатації

Продовження таблиці 2.2 – Види відмов

Перемежовані	збої одного характеру, що багаторазово виникають

Для визначення надійності мережі, між умовними вузлами a_s і a_t, при умові що задано купа шляхів, які можуть бути використанні для заданого зв'язку. Значення вірогідності безвідмовної роботи всіх ребер - p_{ij}.

Для знаходження значення надійності (ρ_{st}^k), заданого шляху (μ_{st}^k), використовується формула зазначена в [3]:

$$\rho_{st}^k = \prod_{\forall b_{ij} \in \mu_{st}^k} \rho_{ij} \quad (1)$$

Дані вузли з'єднують, за умови визначення показника надійності мережі, також враховується надійність всіх шляхів, за формулою [3]:

$$\rho_{st} = \rho(m_{st)} \quad (2)$$

Показник надійності мережі вираховується по різному, в залежності від того яка структура заданої мережі. Надійність задається наступними показниками послідовного з'єднання $p_1, p_2, \dots p_n$, n –кількість поєднаним між собою елементів. Дана послідовність, яка представлена на рис. 2.1 відповідає наступному співвідношенню [3]:

$$\rho = \rho_1 \ \rho_2 \dots \rho_n \quad (3)$$

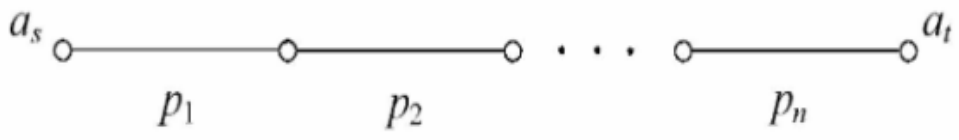


Рисунок 2.1 – Послідовне з'єднання елементів

Паралельне з’єднання елементів розраховується за допомогою виразу[3]:

$$\rho = 1 - \prod_{i=1}^{n}(1 - \rho_i) \quad (4)$$

2.2 Поняття живучості мережі

Живучість мережі – це здатність зберігати та відновлювати роботу основних задач мережі в тому обсязі в якому вони були задані з самого початку, при умові зміни налагодженої структури роботи через зовнішні фактори. Також можна сказати, що це здатність підтримувати функціонування і виконання необхідних основних функцій в умовах заданих характеристик якості.

Одним з головних показників живучості є запас живучості (d), тобто кількість дефектів -1. Дефект – це одиниця виміру збитку, який нанаситься інформаційній системі. Можна взяти С за критинчу кількість дефектів, тобто тоді коли система не зможе функціонувати. Тоді показник дефекту змінюється на d=C-1.

Критичні показники дефектності – це мінімальна кількість дефектів, так як коли вони виникаються система припиняє свою роботу.

Запас живучості можна визначити як максимальна кількість дефектів, так як коли вони є система може вистояти та працювати.

Математична теорія графів показники структурної живучості показуються у вигляді кількісної міри зв’язності структури графа, таких як – зв’язність, узагальнення, довжина графа, вузлова зв’язність та оцінка ймовірності формування працездатної структури в тому випадку коли з’являються несприятливі випадки.

Для розв’язання цих задач потрібно визначити число шляхів, які будуть залишатися в мережі при виникненні зовнішніх збоїв, що призведе до видалення певної гілки зв’язку мережі.

Оцінка ймовірності виключення певного шляху з врахуванням видалення будь-якої певної гілки, була описана в роботі [5]. Після цього описано вираз для к-ті шляхів

рангу r, які будуть залишені навіть після того як будуть видалені гілки з мережі l. Гілок залишиться $L = L_{max} - l$.

$$M_{r,l} = \frac{n(n-1)}{2} C_{r-1}^{r-1}(1 - \frac{2m_{r,L_{max}}}{n(n-1)C_{n-2}^{r-1}}), \quad (5)$$

тут $m_{r,L_{max}}$ – кількість шляхів рангу r, які знаходятся в зв'язку i-j, вузли мережі - n, L_{max} гілок.

Живучість системи визначається на трьох рівнях інформаційних систем: проектуання, функціонування та моделювання. Аналіз функціонального рівня живучості використовуються матричні, ймовірностні, графічні та теоритично-ігрові моделі.

Є ймовірність, що в інформаційних системах складність працездатних компонент забезпечена.

Характеристики аналізу живучості інформаційної системи:

- Ціль функціонування;
- Множина задач $Q = \{q_1 \dots, q_m\}$, вирішення цих задач забезпечено за допомогою інформаційних систем;
- Множина компонентів $\{S_1, S_2 \dots\dots, S_p\}$, які є частинами системи.

Компоненти системи під час її функціонування можуть знаходитися в цих станах:

- Працездатному;
- Непрацездатному;
- Частково працездатному.

2.3 Властивість відмовостійкості мережі

Відмовостійкість – властивість мережі підтримувати повністю або частково працездатність. В тих випадках коли відмовляють окремі елементи системи, які не

зв’язані з безпідставними зовнішніми діями. Для підвищення рівня відмовостійкості можливо підвищити рівень конфігурації апаратного та програмного забезпечення мережі або резервуванням інформаційних ресурсів.

Основна властивість відмовостійкості це те, що кінцевий користувач до останнього не знає про те, що система або деякі її компоненти вийшли з ладу. Це свідчить про те, що відмовостійка мережа автоматично змінює налаштування системи, коли відбувається якийсь збій. Для цього програмне забезпечення системи шукає резервні шляхи, які допоможуть мережі все ще працювати, не роблячи повне загасання мережі. Те які можливі негативні наслідки можуть впливати на відмовостійкість мережі показано на рис. 2.2.

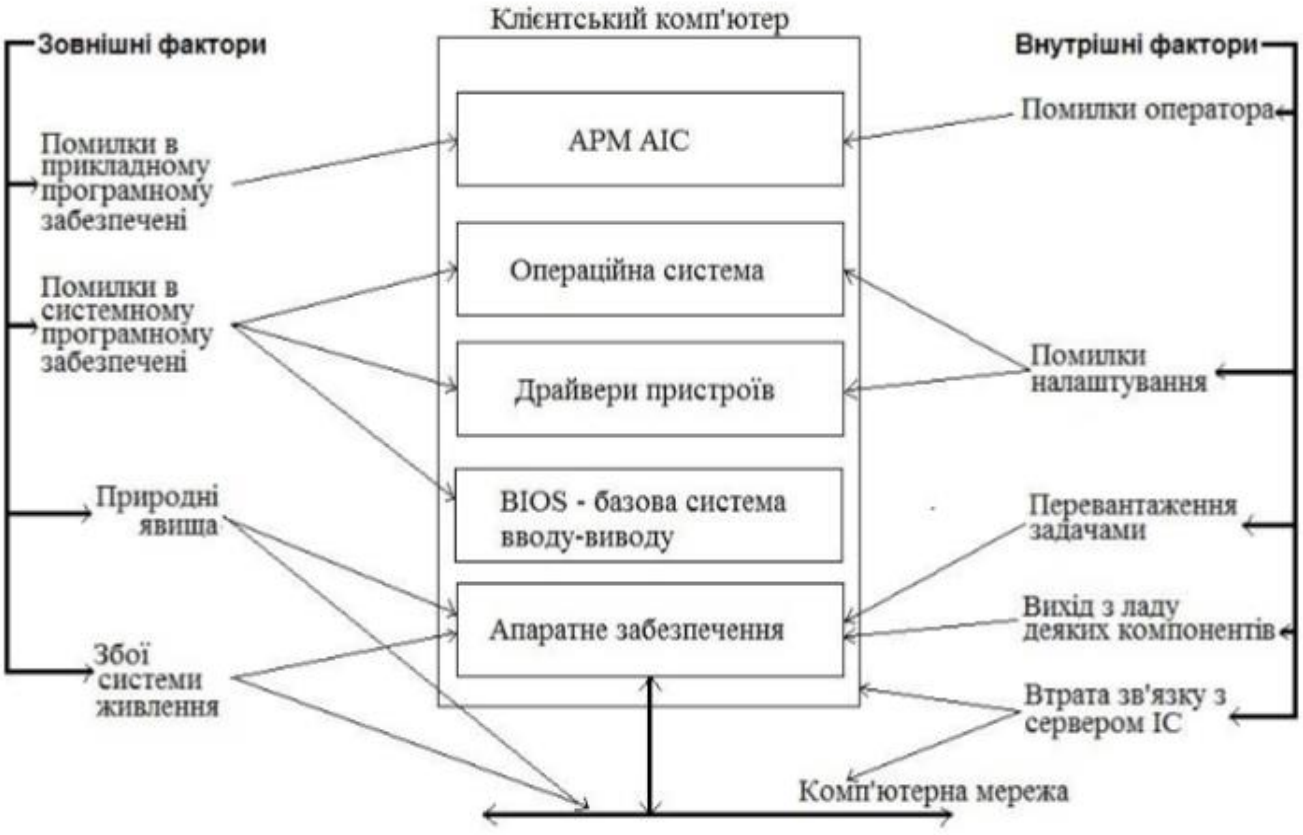


Рисунок 2.2 – Схема впливу негативних факторів

Негативні фактори (збої), які можуть якось впливати на відмовостійкість частини діляться на дві групи: внутрішні та зовнішні. До внутрішніх факторів відносяться помилки, які виникають через збої в програмному коді, а зовнішні помилки можуть бути викликаними через природні явища, або через перепади/збої в роботі енергосистем живлення, що і викликають відмови в роботі інформаційних систем.

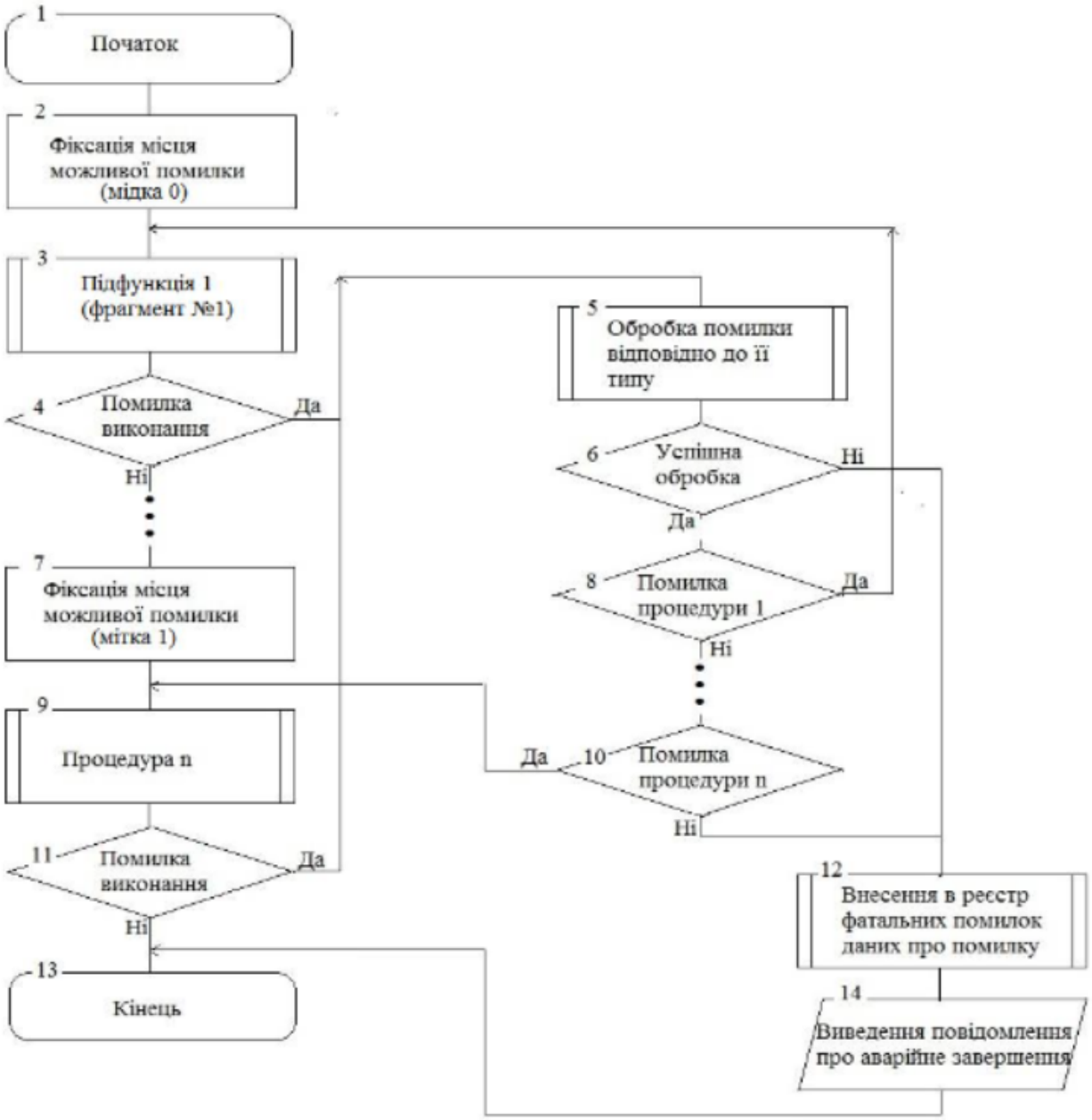


Рисунок 2.3 – Алгоритм обробки помилок в системі

ПЗ, яке відноситься до клієнтських частин інформаційних систем та викликає помилки в роботі, фіксуються разом з тими параметрами з якими вони були використані, і далі заносяться в автоматичні реєстри системи, аби в майбутньому їх можна було використовувати для аналізу і подальшого усунення причин викликаних ними проблем.

Для цього всі процедури, в яких можуть бути присутні подібні критичні помилки, які можуть порушити роботу системи, розроблюються посилаючись на певний шаблон (алгоритм побудови), структура якого представлена на рис. 2.3.

На початку виконання тієї частини, де може бути гіпотетична помилка, в реєстр помилок записується інформацію про неї, для того щоб в подальшому її можна було відстежити та передбачити такого роду помилки і усунути до їх виникнення. Алгоритм припускає такі варіанти розвитку подій, які представлені в таб.2.3. Таким чином відмовостійкість можна забезпечити, виконуючи вище описані дії.

Таблиця 2.3 – Результати виконання алгоритму

Результат виконання програми	Що відбувається
Фрагмент заданої програми виконався без помилок (успішно)	Інформація про нього в реєстрі помилок видаляється, і алгоритм переходить до іншого фрагменту;
Сталася помилка (була успішно усунена обробником помилок).	Запис про цю помилку також може бути видаленим з реєстру помилок
Сталася помилка (не була усунена)	Інформація про помилку залишається в реєстрі помилок

2.4 Стійкість мережі

Функціональна стійкість системи означає, що вона може виконувати свої функції за заданий в конфігурації час, навіть, при умові, що на неї відбувається вплив потоку збоїв (зовнішніх або внутрішніх).

Математична модель РІЗ можна представити у виді неорієнтованого графу [7]:

$$G(V,E), v_i \in V, e_{ij} \in E, i,j = \overline{1,n}, \quad (6)$$

Дану математичну модель також можна описати у вигляді матриці суміжності[7]:

$$A = || a_{ij} ||, \quad i,j = \overline{1,n}, \quad a_{ij} = \begin{cases} 1, & e_{ij} \in E \\ 0, & e_{ij} \notin E \end{cases}, \quad (7)$$

де V –множина вузлів мережі в кількості n, а E – множина ліній по яким відбуваються передача даних в мережі. Тобто можна сказати, що для кожного вузла існує, як мінімум один шлях для передачі даних. Дану систему можна представити у вигляді незв’язного графу, що дає можливість оцінити її функціональну стійкість.Канали зв’язку даної системи мають пропускну спроможність і це дозволяє передавати по ній великий обсяг інформації.

Зовнішній вигляд графу і його властивості дає можливість передбачити до якого виду структури можна віднести систему, вони представлені в таб.2.4.

Таблиця 2.4 – Структури мережі

Структура	Ознака структури
Функціонально стійка структура	Граф даної структури повинен бути однокомпетентним і не мати ніяких мостів або вузлів з’єднання
Функціонально нестійка структура	Граф цієї структури повинен бути багакомпонентним і незв’язним

Кількісну оцінку для визначення ступеня функціональної стійкості структури, розглядають показники представлені в таб. 2.5.

Таблиця 2.5 – Показники функціональної стійкості структури

Показник	Властивості
Вершинної зв'язності $\chi(G)$	Мінімальна кількість вершин, які при видалені з відповідними їм ребрам, робить граф незв'язним та одновершинним
Реберної зв'язності $\lambda(G)$	Мінімальна кількість ребер, які при видалені, роблять граф незв'язним
Ймовірність зв'язності $P_{ij}(t)$	Ймовірність, що передача повідомлення з вузла і в вузол j, буде пердавати не більше часу t

Описані вище показники виділяють такі особливості:

- Поточна структура характеризується вершинною і реберно зв'язністю, що не залежить від того наскільки надійні вузли комутації або лінії зв'язку;
- Цілі значення з'являються у показників $\chi(G)$ та $\lambda(G)$ та пов'язується між собою наступним співвідношенням [7]:

$$\chi(G) \leq \lambda(G) \leq \min_{V_i \in V}\{deg(v_i)\} \quad (8)$$

- За допомогою $P_{ij}(t)$ (ймовірності зв'язності) з'являється можливість врахувати, яка буде надійність обладнання зв'язку, який буде тип каналу для передачі даних, чи будуть в системи резервні канали або маршрути.
- Тільки одна пара вершин може бути охарактерезована ймовірністю зв'язності.

Характеристику зв’язності для всіх пар вершин можна визначити за допомогою матриці ймовірностей [7]:

$$P = || P_{ij} ||, i, j = \overline{1, n} \quad (9)$$

Таблиця 2.6 – Критерії функціональної стійкості структури

Критерій	Ознака	Можливості
Структурний	Показники вершинна і реберна зв’язності відповідають умові $\chi(G) \geq 2 \cup \lambda(G) \geq 2$	Порівняння системи з різними структурами, без складних обчислень.
Ймовірнісний	Ймовірність зв’язності всіх пар відповідають умові $P_{ij}(t) \geq P_{ij}^{3},\ i \neq j,\ i, j = \overline{1, n}$ n – к-сть вершин графа	Враховує можливі технічні параметри каналів зв’язку та порівнювати системи зі схожими топологіями.

З використанням точних розрахунків, дані критерії дозволяють визначити яка буде функціональна стійкість системи [7], які представлені в таб.2.6.

Існує область, де система може бути одночасно функціонально стійкою та функціонально нестійкою та називається – границя функціональної стійкості структури.

Ознакою границі функціональної стійкості є те, що граф зв’язний та складається з мостів ($N_e \geq 1$) або з’єднання між собою вузли ($N_v \geq 1$). Міст – це ребро зв’язного двокомпонентного графа, яке з’єднує між собою підграфи та якщо видалити його граф стане однокомпонентним.

Система, яка перебуває на границі стійкості, може бути працездатною та виконувати необхідні функції, але як тільки один з існуючих мостів (або моста з’єднання) перестає працювати, система стає не робочою.

3. РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ МЕРЕЖІ

3.1 Інструментарій для розробки

Основним інструментом для створення ГІС моніторингу мережі було обрано сучасну мову програмування Python – це інтерпретована мова програмування високого рівня, перша версія мови з'явилася у 1990 році для розподіленої операційної системи Amoeba.

Python підтримує декілька парадигм програмування – структурну, функціональну, об'єктно-орієнтовану, аспектно-орієнтовану та рефлексивну.

До початку минулого року існувало дві версії мови, які суцільно відрізнялися, мали відмінності у синтаксисі та різні особливості функціоналу – Python 2.0 і Python 3.0. Вони розвивалися окремо один від одного, так як при появі версії 3.0, вона була не до кінця сумісною з версією 2.0. Але через 12 років після появи Python 3.0, підтримку другої версії мови було припинено через проблеми з безпекою, що могло призводити до отруєння веб-кешу. Офіційне закриття версії 2.0 також було гарантом того, що всі програмні продукти та модулі можна легко перенести з цієї версії на постійну версію 3.0.

Мова підтримує строгу динамічну типізацію даних – типізація, при якій спочатку змінній присвоюється значення, і вже пісня цього визначається тип призначеного значення та збирача сміття – автоматичне видалення даних з оперативної пам'яті, які не використовуються, під час виконання програми . У Python динамічна семантика та високі рівні структур даних, що дозволяє швидко розробляти програми та поєднувати вже готові рішення.

Розробники вложили в мову Python ідею філософії програмування названу «The Zen of Python». Ідеї цієї філософії можна отримати, навіть в інтерпритаторі самої мови, використавши команду `import this`. Автором філософії виступив Тім Пейтерс.

На рис. 3.1 представлена головна сторінка сайту Python, де можна знайти детальну документацію з необхідними пояснення синтаксису та семантики мови. Також можна скачати актуальну версію мови, знайти старі версію, якщо необхідно створити модулі для підтримки вже існуючих програм.

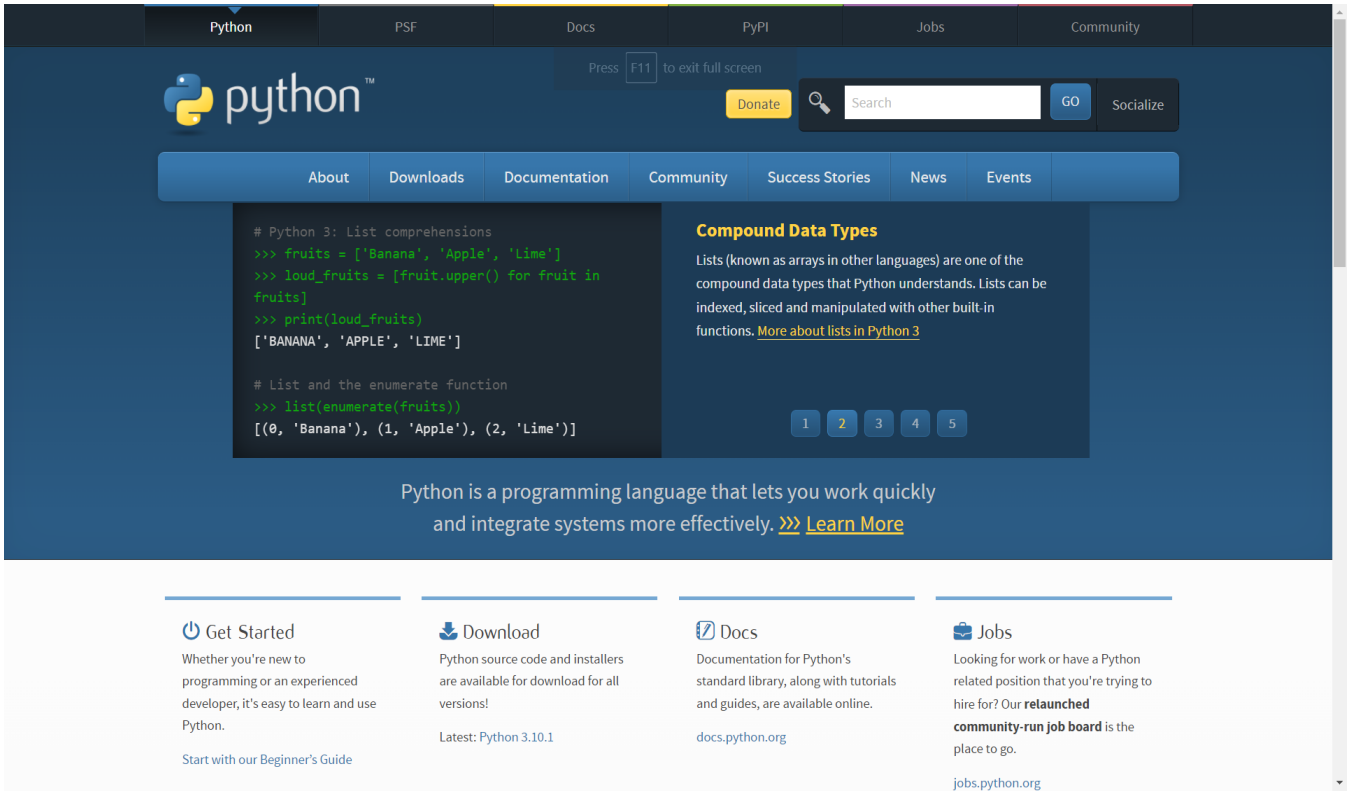


Рисунок 3.1 – Головна сторінка Python

Переваги мови програмування Python:

- синтаксис коду чистий від «зайвих» символів, що дозволяє написати код швидше та з використанням меншої кількості символів;
- можливість переносу програми на різні платформ;
- влаштоване середовище має необхідні функції для налаштування та тестування не громіздких програм (IDLE);

- багате різноманіття стандартний бібліотек, які полегшують написання програм;
- можливість написання власних бібліотек, які в подальшому можуть використовувати інші користувачі;
- режим діалогового вікна (термінал) для швидкого тестування та відлагодження необхідних функцій;
- простота використання для розв'язання математичних задач, так як непотрібні глибокі знання мови для написання математичних рішень;
- open-source та підтримка редагування коду користувачами;
- масштабованість мови, що дозволяє постійно розширювати функціонал систем та можливість підтримки великої кількості користувачів.

Головним недоліком є те, що програми написані на Python не можна компілювати, тобто код виконується построково, а не перетворюється в двійковий код, що робить виконання програми більш повільним на фоні виконання додатків, написаних на компільованих мовах програмування.

Python є ідеальною мовою програмування для написання скриптів та розробки програмах у більшості галузях: системи Google. Так як мова інтерпритована, її можна легко використовувати на майже всіх платформах і вільно розповсюджувати. Також плюсом інтерпитованості мови є те, що її інтерпритатор можна розширити функціями та типами даних, використовуючи мову C або C++. Python також можна самостійно розширювати та створювати додаткові модулі та бібліотеки для прикладних програм, які в майбутньому необхідно наладити.

Також для розробки використовується відкритий, високорівневий Django веб-фреймворк, за допомогою якого можна створювати серверну частину складних веб-систем.

Архітектура фреймворку Django дуже схожа на архітектуру шаблону MVC (Model-view-controller), дану модель шаблону вводять самі розробники – MTV (Model-

template-view), лише з заміною деяких понять. Так в MTV controller заміняється на view, а те що було в MVC view, заміняється на template, model залишається на тому ж місці, що і в моделі MVC. Таким чином, архітектура Django - MTV залишає подібну MVC структуру, вводячи деякі нововедення та має наступний вигляд, який представлено на рис. 3.2 у вигляді структури обміну даними.

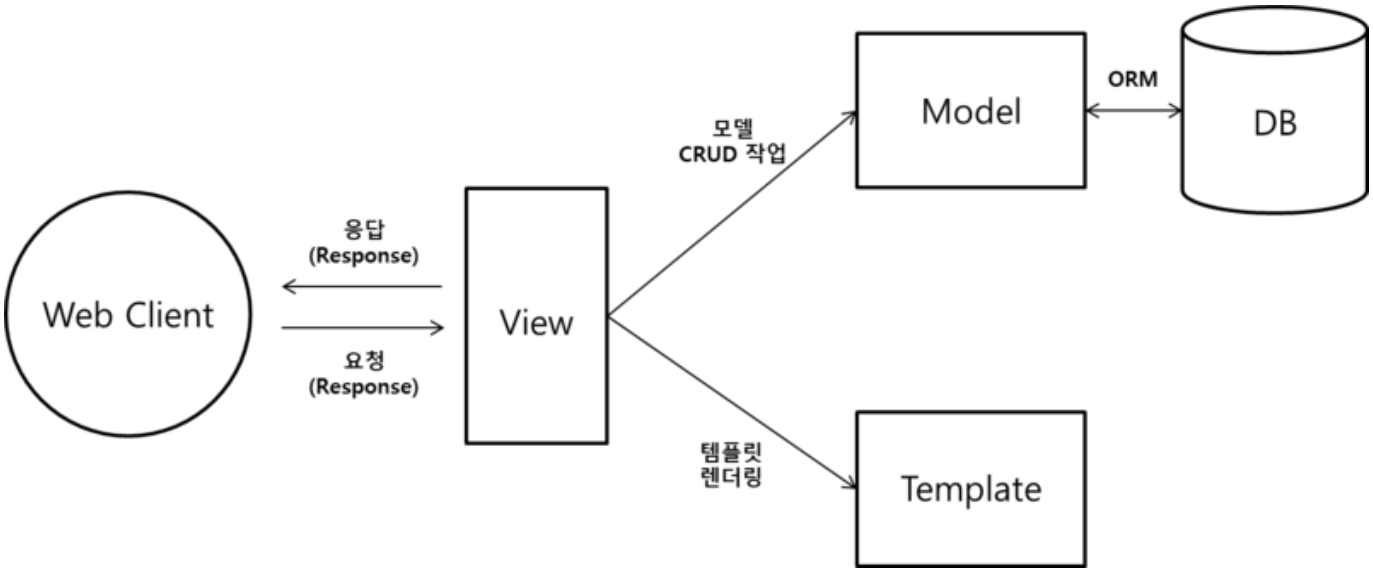


Рисунок 3.2 – Представлення MTVмоделі

Ідеєю архітектури MTV в тому, що всі частини системи постійно в зв’язку один з одним: БД, візуальне відображення та серверна частина, яка займається всіма обчисленнями, за допомогою цього реалізується коректна робота всієї системи.

Можливості веб-фреймворку Django, за допомогою яких можна легко створювати складні системи:

- швидкодія розроблюваних систем;
- підтримка об’єктно-реляційного відображення (ORM);
- API доступу до БД;
- вбудована в систему кешування;

- можливість підключення реалізованих модулів для авторизації аутентифікації;
- вбудована бібліотека модулів для роботи з формами;
- підтримка інтерфейсу адміністратора;
- підтримка актуальної документації в налаштування інтерфейсу;
- вбудована система шаблонів;
- архітектура MTV;
- безпечність;
- проста реалізація розроблюваних систем;
- структура створюваних додатків, які в подальшому можна переносити на інші проекти.

Для візуальної складової системи використовуються ряд технологій:

- HTMl – мова розмітки тексту, за допомогою неї реалізується структура відображення сторінки інтерфейсу;
- CSS – таблиця стилей, за допомогою неї можна редагувати візуальне відображення структури сторінки;
- JSON – технологія, яка дозволяє передавати дані у вигляді об’єктів з серверної частини на клієнтську для зручного відображення подальших даних в інтерфейсі, структура передачі даних представлено на рис.3.3;

Рисунок 3.3 – Структура даних JSON

- JavaScript - динамічна прототипна мова програмування, яка використовується для створення веб-сторінок з клієнтської сторони, JavaScript дозволяє створювати реалізацію для взаємодії системи з користувачем, управлінню браузером, та асинхронно обмінюватися даними з сервером.

Основним елементом клієнтської реалізації буде мова JavaScript, вона дозволяє реалізувати асинхронну передачу даних та прописувати всі необхідні обчислення, які потрібні на стороні клієнта.

Дану мову програмування можна використовувати для:

- написання сценаріїв;
- надання веб-сторінкам інтерактивності;
- програмування зі сторони сервера, використовуючи Node.js;
- для використання на pdf сторінках;
- написання мобільних додатків, за допомогою фреймворку React Native;
- написання прикладних програм.

Мову JavaScript часто плутають з мовою програмування Java, але це зовсім різні мови, які мають різну семантику коду, правила іменування та абсолютно різне застосування в програмуванні.

Відображення карти розроблюваної системи реалізовано за допомогою бібліотеки з відкритим кодом для JavaScript – Leaflet. Вона не велика за обсігом, самодостатня та проста у використанні.

Бібліотека підтримує реалізацію слоїв карт, які будуються за технологією WMS та GeoJSON, для векторного відображення на сторінці. Головна сторінка сайту бібліотеки Leaflet представлена на рис 3.4.

Тут можна знайти документацію про налаштування бібліотеки на сторінці додатку та всю необхідну інформацію для роботи з елементами карти та

налаштування всіх її елементів, таких як мітки, зони, зання точок на мапі, та зміни карти в додатку.

Рисунок 3.4 – Головна сторінка

Система повинна завантажувати дані про мережу, які вводить користувач в систему та зберігати їх і використовувати для обчислень та відображення їх на карті графу. Для цього необхідно використовувати БД в системі, для її реалізації можна скористатися будь-якою СКБД. В даній системі було прийнято рішення використовувати відкритою, об’єктно-реляційною СКБД PostgreSQL. Основою даної СУБД є мова запитів SQL (structured query language).

Можливості і переваги функціональності СКБД PostgreSQL:

- надійність – підтримка принципів ACID (Atomicity, Consistency, Isolation, Durability)
- цілісність даних, реалізується за допомогою зовнішніх ключів;
- продуктивність, реалізується за допомогою системи блокування, масштабованості, керування буферами пам’яті;

- розширюваність – систему можна налаштовувати за допомогою додавання нових функцій;
- двійкова реплікація – дані розподіляються у вигляді вузлів, це дозволяє підвищити швидкість пошуку даних;
- необмежений розмір БД;
- можливість використовувати технологію JSON в БД;
- об’єктно-реляційний тип БД;
- підтримка різних типів даних та можливість створювати свої.

Використовувати СУБД PostgreSQL найкращим рішень буде в парі з Docker – інструмент управління ізольованими контейнерами, що дозволяє керувати ізольованими процесами. Процеси можуть бути запущені довільно і в подальшому ці ізольовані процеси можна просто переносити на інші сервери у вигляді контейнерів даних.

Інструментарій був написаний мовою програмування Go під ліцензією Apache, базується на використанні механізмів ізоляції ядра Linux на основі неймспейсів та груп управління.

Архітектура Docker складається з таких елементів:

- демон (програма, яка запускається на віртуальній машині та працює в фоновому режимі);
- клієнт (необхідний для взаємодії з демоном);
- образ (додаток, який необхідно встановити з DockerHub, складається з ОС та додатку необхідної програми);
- контейнер – образ, який запускається для виконання необхідних функцій скачаного додатку, саме з цим елементом ми працюємо як з повноцінною програмою.

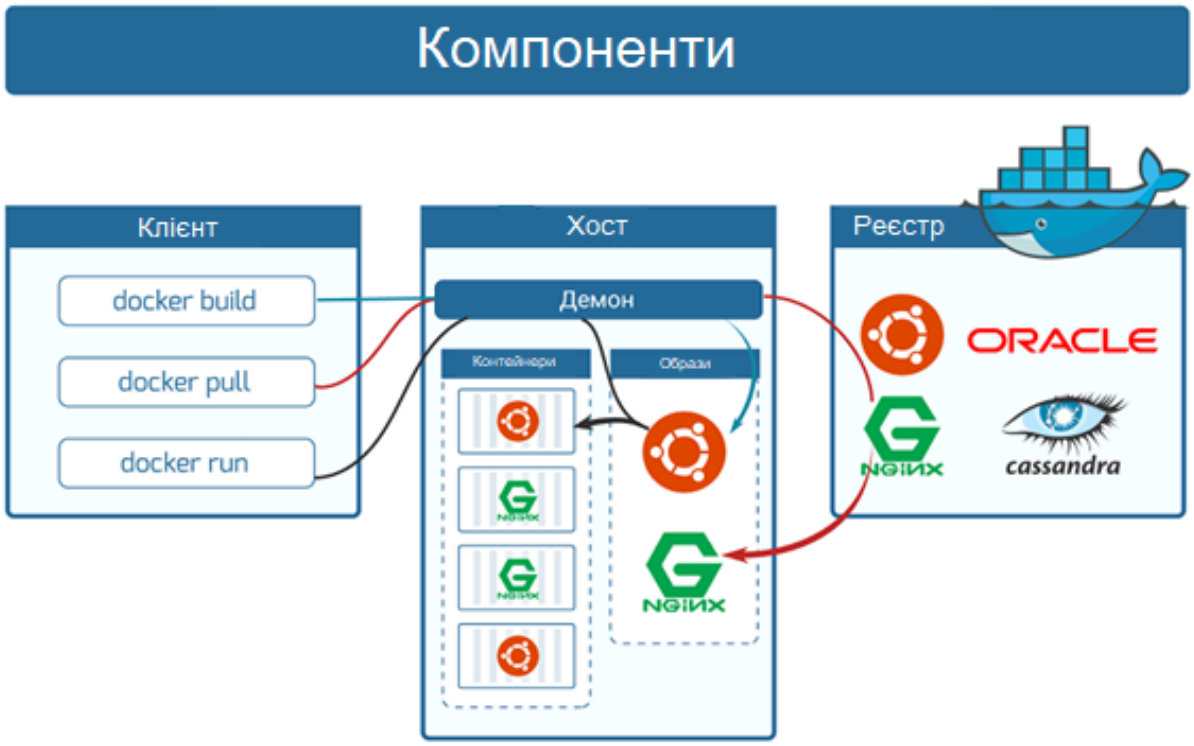


Рисунок 3.5 – Структура Docker

В якості середовища програмування використовуємо PyCharm, яке використовується для роботи з мовою програмування Python. ІС розроблено компанією JetBrains на базі системи IntelliJ IDEA.

ІС має дві версії для персонального використання та професійного (платна), якщо користувач хочу спробувати попрацювати з PyCharm, можна використати безкоштовну версію, але при цьому ряд функцій буде недоступним. Для повноцінної розробки краще використовувати професійну версію. У безкоштовної версії доступний тимчасовий доступ до повноцінної версії.

Середовище дає можливість аналізу коду, графічного відображення, для юніт-тестів, та підтримує розробку на веб-фреймворку Django. PyCharm є кросплатформним додатком, яке може працювати на всіх популярних ОС: Windows, MacOS, Linux.

Можливості ІС:

- великий вибір інструментів для веб-розробки, при використанні фреймфорку Django;
- можливість рефакторингу коду;
- використання утиліти Google App Engine;
- можливість використання систем контролю версій;
- використання інструментів для тестування;
- зручна реалізація навігації між елементами проекту.

Робота системи моніторингу була асинхронною необхідно використовувати асинхронну чергу завдань, для цього ідеально підійде інструмент Celery, який підтримує операції в режимі реального часу та чудово підходить для роботи з фреймворком Django. Написаний мовою програмування Python, але так само підтримує використання скриптів і на інших мовах програмування за допомогою вебхуків.

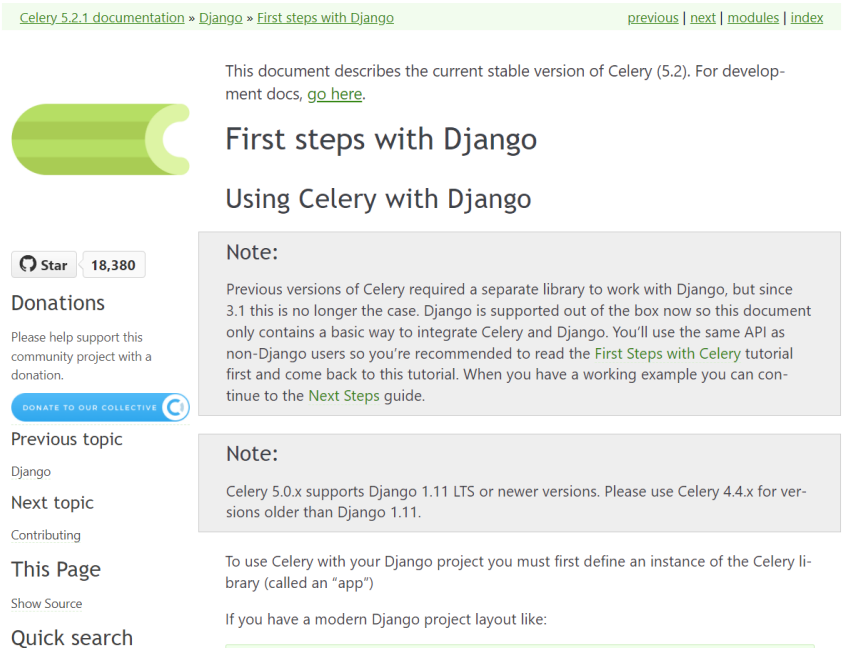


Рисунок 3.6 – Головна сторінка Celery

Так як Celery підтримує роботу з Django, на офіційному сайті інструменту можна знайти окрему частину про роботу з фреймворком, де прописані всі необхідні елементи керування, скриптів та інструкції по встановленню. Приклад головної сторінки сайту Celety представлено на рис.3.6.

Блоки скрипту Celery називаються задачами та виконуються на одному або на синхронно на всіх прописаних раніше вузлах. Ці завдання можуть виконуватися і асинхронно (на фоні) або чекати готовності виконання в черзі.

3.2 Налаштування інструментів розробки

Перш за все необхідно скачати на ПК Python та встановити, та не забути поставити параметр, який показано на рис. 3.7 – Add Python to PATH. Після установки, необхідно перезавантажити ПК.

Рисунок 3.7 – Установка Python

Далі необхідно створити проект для подальшої розробки, для цього треба:

- відкриваємо командну строку або термінал;
- переходимо в потрібну папку, де буде створений проект;
- створюємо віртуальне середовище для проекту за допомогою команди –

```
python -m venv venv;
```

- активуємо віртуальне середовище, за допомогою команди –

```
pip install django;
```

- встановлюємо фреймворк Django всередині віртуального середовища, використовуючи стандартну команду Python –

```
pip install django;
```

- після цього створили проект, наступною командою –

```
python -m django startproject ukrenergo;
```

- також необхідно створити додаток, в якому буде реалізовано основні функції проекта –

```
python manage.py startapp map.
```

Після цього проект створено і далі необхідно установити ІС, як вже було описано вище PyCharm, для цього потрібно перейти на офіційний сайт ІС та завантажити його. Запускаємо додаток та відкриваємо в ньому проект. Необхідно налаштувати інтерпритатор Python, заходимо в налаштування, в вкладку інтерпритатора як показано на рис. 3.8.

В налаштуваннях, у нас вибрано віртуальне середовище, яке ми встановили раніше, обираємо вже існуюче середовище, після чого зберігаємо зміни. Після збереження нових налаштувань, в меню Python interpreter з’являється повна інформація про версії, які використовуються в створюваному проекті, приклад показано на рис.3.9.

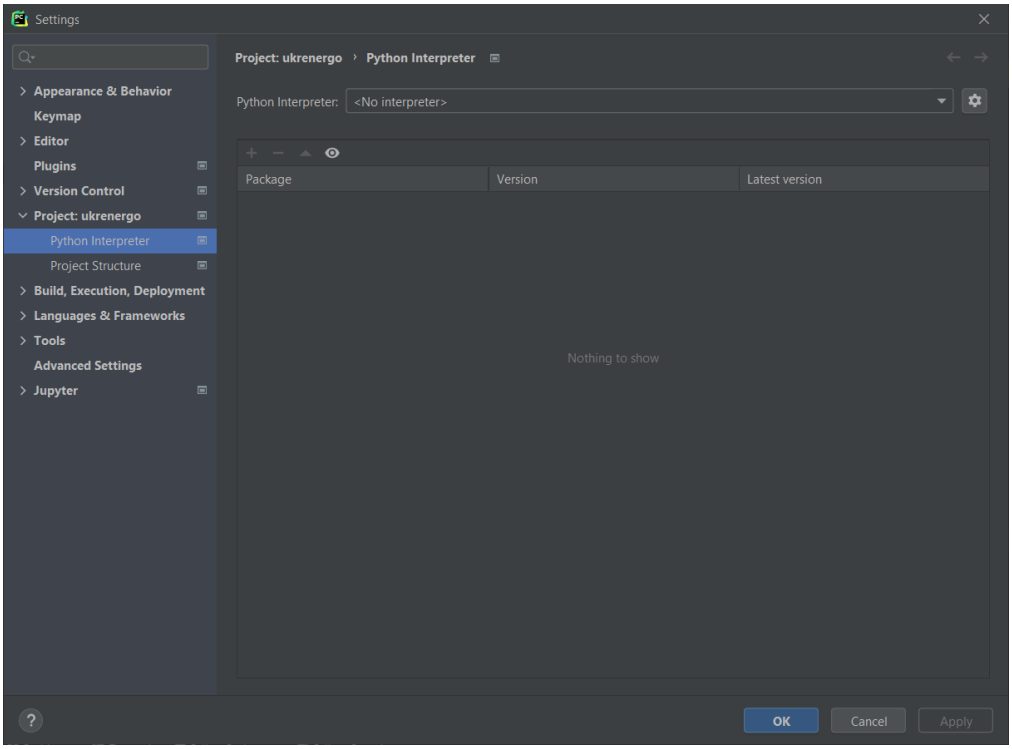


Рисунок 3.8 – Налаштування інтерпритатора

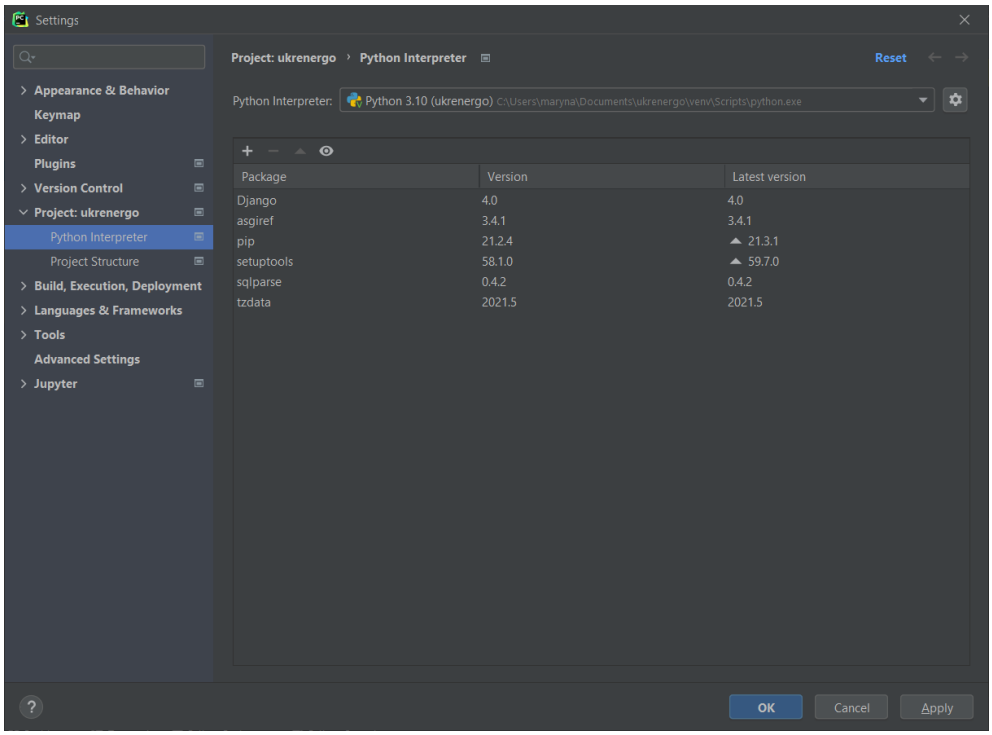


Рисунок 3.9 – Збереження інтерпритатора

Для коректної роботи проекту необхідно додати всі створені додатки до блоку установлених (INSTALLED_APPS) додатків в налаштуваннях проекту (файл – settings.py), приклад на рис.3.10.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'map',
]
```

Рисунок 3.10 – Підключення додатків

Наступним кроком в налаштуванні проект буде створення URLconfig, для цього створили в паці додатку файл з назвою urls.py та в папці всього проекту знаходимо файл з такою ж назвою і редагуємо його, додаючи шлях до URL файлів додатку map у весь проект, як показано на рис.3.11.

```
from xml.etree.ElementInclude import include

import ...

urlpatterns = [
    path('map/', include('map.urls')),
    path('admin/', admin.site.urls),
]
```

Рисунок 3.11 – URL pattern

Для цього було використано функцію include(), яка доззволяє використовувати посинання на інші URLconfig. Коли Django зустрічає функцію include() від переходить по головній строці URL.

Наступним кроком буде установка та налаштування утиліти Docker. Скачуємо та авторизуємся в аккаунті Docker, як показано на рисунку 3.12.

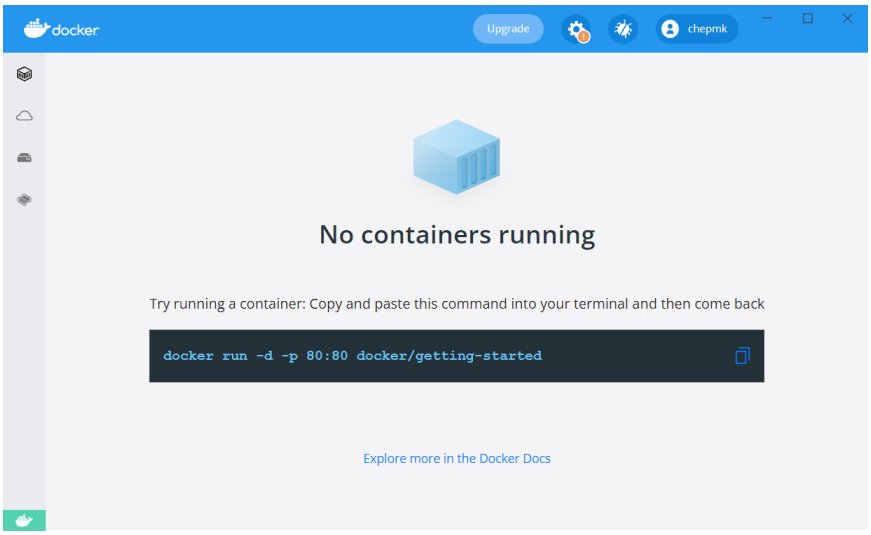


Рисунок 3.12 – Сторінка додатку Docker

Необхідно перевірити його на працездатність, виконуємо команду:

```
docker run -d -p 80:80 docker/getting-started
```

```
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
97518928ae5f: Pull complete
a4e156412037: Pull complete
e0bae2ade5ec: Pull complete
3f3577460f48: Pull complete
e362c27513c3: Pull complete
a2402c2da473: Pull complete
eb65930377cd: Pull complete
69465e074227: Pull complete
Digest: sha256:86093b75a06bf74e3d2125edb77689c8eecf8ed0cb3946573a24a6f71e88cf80
Status: Downloaded newer image for docker/getting-started:latest
a031b03be2303615b1549cef79cdd127188adfb8ff938e1e607ed20a0cf8335b
```

Рисунок 3.13 – Запуск додаток

Якщо все працює правильно, ми бачимо наступний вивід в консолі, приклад показано на рис.3.13.

Для того аби підключити контейнер з конфігурацією PostgreSQL, потрібно використати наступну команду, яка створить контейнер для СКБД та відкриє порт для роботи з ним, щоб можна було отримати доступ з хоста:

```
docker run -p 5432:5432 --name gisEnergo -e
POSTGRES_PASSWORD=1111 -d postgres
```

З цим варіантом можуть виникнути проблеми в майбутньому і при установці нової версії Postgres , дані про БД можуть бути видалені, для того аби цього уникнути необхідно використовувати вторинний контейнер даних, для цього створюємо контейнер даних, використовуючи команду:

```
docker create -v /var/lib/postgresql/data --name
PostgresData alpine
```

Контейнер описаний вище створює Postgres на основі образу Alpine. Важливим нюансом використовувати параметр –v, який відповідає шляху, де знаходиться СКБД, той що очікує Postgres.

Контейнер, який забезпечить безпеку для наших даних, тепер необхідно створити справжній контейнер, який буде працювати з нашими даними, для цього використовуємо наступну команду:

```
docker run -p 5432:5432 --name yourContainerName -e
POSTGRES_PASSWORD=yourPassword
-d --volumes-from PostgresData postgres
```

Ця команда відрізняється від першої параметром –volumes-from PostgresData, що вказує нашій команді, що необхідно використовувати контейнер для безпеки даних PostgresData.

Для перевірки наявних контейнерів, запустити команду, яка покаже вс існуєчі у нас контейнери:

```
docker ps –a
```

```
λ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
f6a07ce33454        postgres            "docker-entrypoint.s…"   9 minutes ago
71fe42ad8d8c        alpine              "/bin/sh"                22 minutes ago

STATUS              PORTS                     NAMES
Up 3 seconds        0.0.0.0:5432->5432/tcp    Moon
Created                                       PostgresData
```

Рисунок 3.14 – Перевірка існуючих контейнерів

На рис. 3.14 бачимо відображення всіх створених контейнерів, на даному прикладі показано двоє створених контейнерів, один з них запущений, тобто працює, контейнер який створено для безпеки даних, він не запускається, його не потрібно видаляти.

Тепер переходимо до роботи безпосередньо з Postgres. Завантажуємо платформу для адміністрування і розробки для PostgreSQL – pgAdmin на офіційному сайті для необхідної ОС. Установлюємо інсталятор для нашої СКБД.

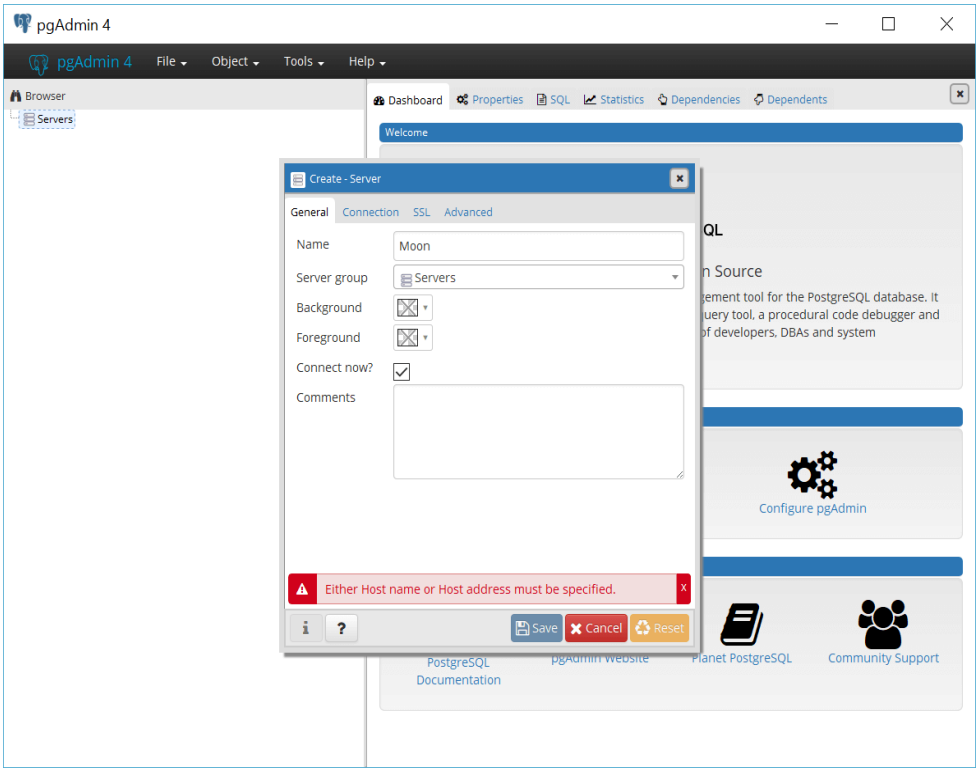


Рисунок 3.15 – Налаштування сервера в pgAdmin,

Правою клавішою миші обирає Сервес та вибираємо Створити –> Сервер. В діалоговому вікні «Створення сервера» вводимо ім’я необхідного сервера, далі переходимо на вкладку «З’єднання», приклад показано в рис.3.15.

Заповнюємо вкладку створення сервера, з даними вказаними в таб. 3.1.

Таблиця 3.1 – Значення для створення сервера

Поле	Значення
host	localhost
username	Имя користувача БД
password	Пароль, який використовували для параметра команди POSTGRES_PASSWORD
maintenance database	Створена БД
port	5432, значення порта задається при установці панелі адміністрування

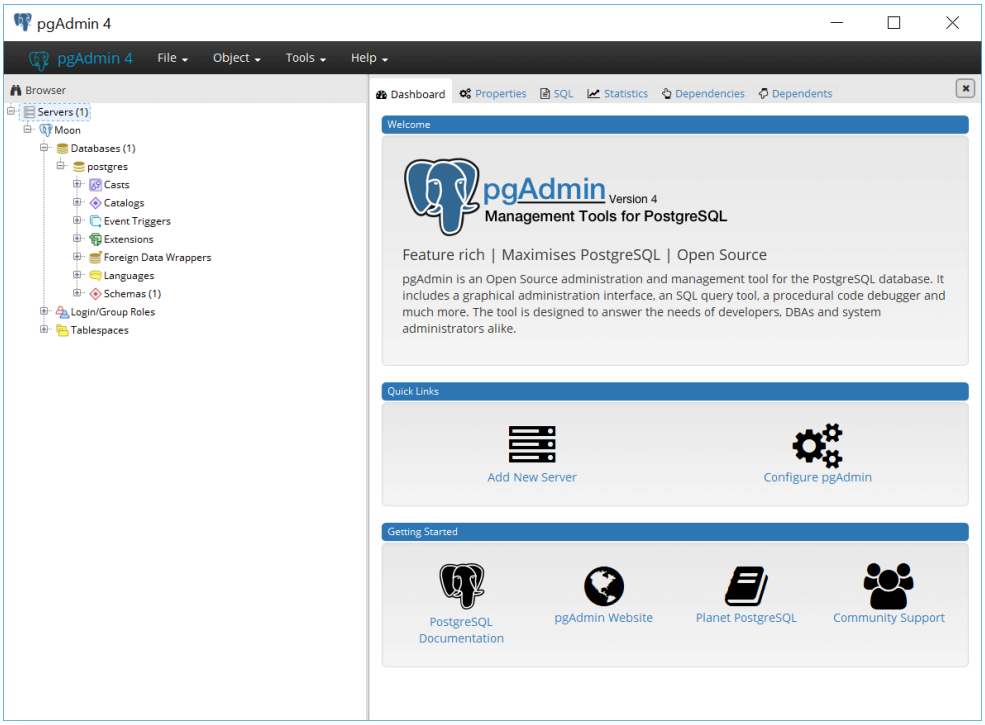


Рисунок 3.16 – Налаштована панель адміністратора

Після збереження налаштувань, зберігаємо та закриваємо діалогове вікно та підключаємося до сервера. На рис.3.16 знизу можемо побачити, які є доступні БД на сервері. БД за замовчуванні є поточна база.

Далі переходимо до налаштування БД в самому проекті Django в IC PyCharm.

PostgreSQL використовує ізоляції як і сам фреймворк. Щоб використовувати вищий рівень ізоляції, можливо встановлення в конфігурації БД наступних налаштувань, які представлені на рис. 3.17.

Використовуючі високі рівні ізоляції, програма повинна оброблювати винятки, щоб перехватувати в разі виникнення помилок.

```
import psycopg2.extensions

DATABASES = {
    # ...
    'OPTIONS': {
        'isolation_level': psycopg2.extensions.ISOLATION_LEVEL_SERIALIZABLE,
    },
}
```

Рисунок 3.17 – Ізоляційні рівні

Підключення БД до Django встановленої раніше бази можно зробити через файл налаштування проекту. По дефолту в Django налаштована конфігурація SQlite, що є зручним у використанні для тестування або навчання.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'ukrenergo_proj',
        'USER': 'ukrenergo_proj',
        'PASSWORD': '1111',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Рисунок 3.18 – Налаштування БД

Для того аби додати іншу СКБД до проекту, а нашому разі PostgreSQL, необхідно налаштувати параметри прив'язки БД в файлі settings.py, налаштування показані на рисунку 3.18, для кожного з елементів об'єкта задаємо необхідні параметри, які представлені в таблиці 3.2.

Таблиця 3.2 – Налаштування бази даних

Ім'я об'єкта	Значення
ENGINE	задаємо параметри БД, яку підключаємо;
NAME	задаємо ім'я БД;
USER	вказуємо ім'я користувача;
PASSWORD	вказуємо пароль до бази даних, яку будемо використовувати;
HOST	хост БД
PORT	порт БД

Після налаштувань необхідно виконати команду для створення міграцій, які показано на рис.3.19, для того аби запустити проект, виконуємо її даною командою:

```
python manage.py migrate.
```

```
(venv) C:\Users\maryna\Documents\ukrenergo>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Рисунок 3.19. – Виконання міграцій

Для роботи БД необхідно визначити моделі, які будуть використовуватися для подальшої роботи системи. Для цього створюємо файл Models, та прописуємо необхідні моделі та задаємо їм поля з даними, які нам необхідні для зберігання даних про мережу. Задання моделей показано на рис. 3.20.

```
from django.db import models


class Data_of_top(models.Model):
    name = models.CharField(max_length=200)
    top1 = models.CharField(max_length=200)
    top2 = models.CharField(max_length=200)


class Top(models.Model):
    name = models.ForeignKey(Data_of_top, on_delete=models.CASCADE)
    value = models.IntegerField(default=0)
```

Рисунок 3.20 – Моделі

Щоб застосувати зміни, які ми зробили в БД, необхідно включити додаток ,ap в програму, для цього виконуємо команду –

```
python manage.py makemigrations polls
```

після цього бачимо зміни в командній строці, приклад представлено на рис. 3.21.

```
Migrations for 'map':
  map\migrations\0001_initial.py
    - Create model Data_of_top
    - Create model Top
```

Рисунок 3.21 – Виконання міграцій

Виконуючи дану команду, ми повідомляємо Django, що були внесені зміни в модель БД і що ми хочемо зберегти нові міграції.

Наступним кроком є етап створення суперкористувача. Який може працювати з БД через сайт адміністратора, який є влаштований в Django і зручний у використанні. Для цього використовується команда –

```
python manage.py createsuperuser
```

яку ми вводимо в командній строці і після цього за допомогою інструкцій налаштовуємо всі параметри користувача, для входу в систему адміністрування. Як показано на рис. 3.22 вводимо ім’я користувача, електрону пошту та пароль, після чого супер користувача створено і ми можемо зайти на сервер розробки.

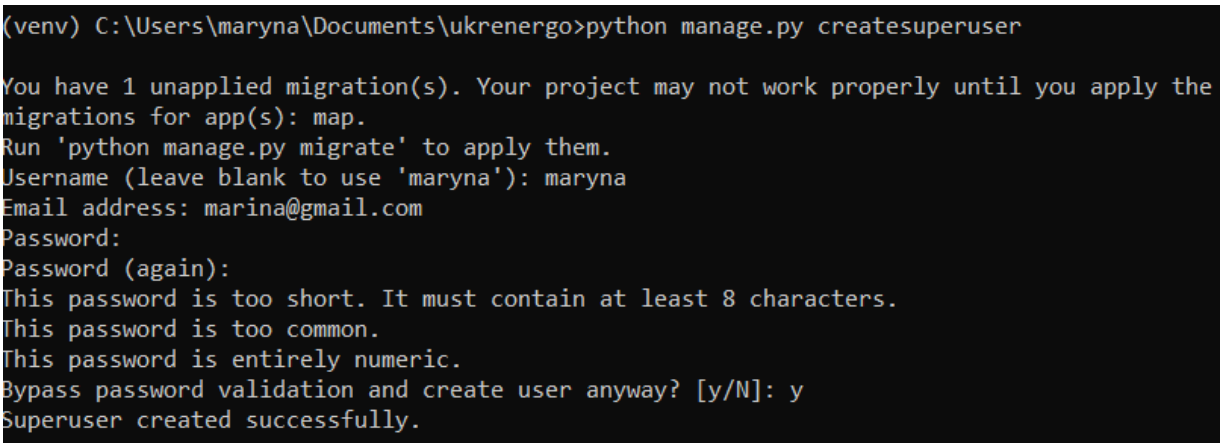


Рисунок 3.22 – Створення супер користувача

Запускаємо сервер через команду:

```
python manage.py runserver
```

Заходимо в браузер через локальний домен http://127.0.0.1:8000/admin/, бачимо екран для входу адміністратору, ввводимо задані дані, по прикладу на рис. 3.23.

На вході в панель адмінітсратора видно, що моделі які було ствоерно раніше не відображаються. Для того аби це виправити, необхідно вказати в додатку в файлі map/admin.py наступні строки коду, представлені на рис. 3.24, які добавляють на сторінку адміністратора компоненти нашої моделі.

Рисунок 3.23 – Вхід на Django administration

```
from django.contrib import admin

from .models import Data_of_top, Top

admin.site.register(Data_of_top)
admin.site.register(Top)
```

Рисунок 3.24 – Додання моделей на панель адміністратора

Після того як ми зареєстрвували моделі, аби вони відображалися на сторінці адмінітратора, ми бачимо, що на панели відбулися зміни у нас є доступ до елементів нашої БД, а також до груп та користувачів – моделі, які вбудовані в конфігурацію Django, візуальне відображення ми бачимо на рис. 3.25 .

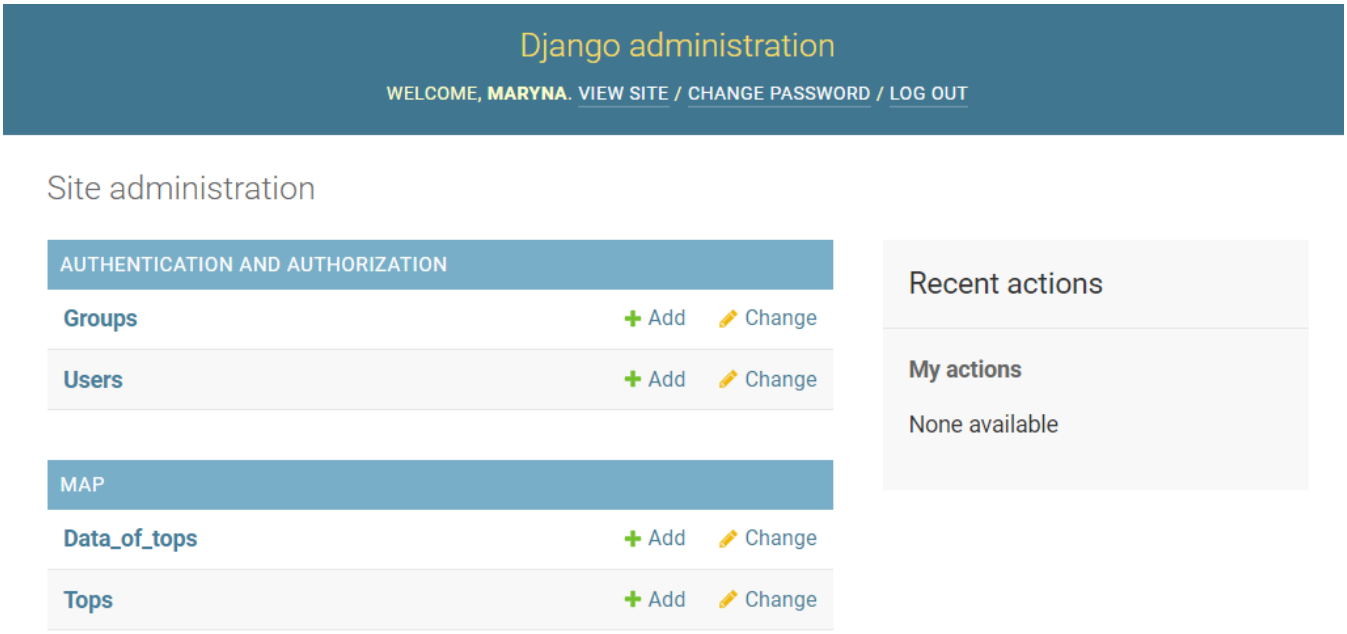


Рисунок 3.25 – Панель адміністратора

Налаштування проект закінчено, далі можна приступати до розробки самої системи. Спочатку необхідно визначити, які саме блоки функцій (головні) будуть виконуватися в системі та як ці блоки будуть взаємодіяти між собою для злагодженої роботи додатку.

Візуальне відображення додатку створюєтья за допомогою мови розмітки гіпертексту HTML. Для того аби правильно використовувати файли для роботи з візуальною частиною необхідно створити папки з шаблонами та статичні файли на кожний вид файлів, ієрархія папок представлено на рис. 3.26, з додаванням необхідних файлів для візуалізації системи.

В папку map (папка головного додатку) додаємо нову папку static, в ній створюємо папку для скриптів, які ми будемо додавати JavaScript скрипти. Також в папці map додаємо папку template, де будуть зберігатися всі html-файли.

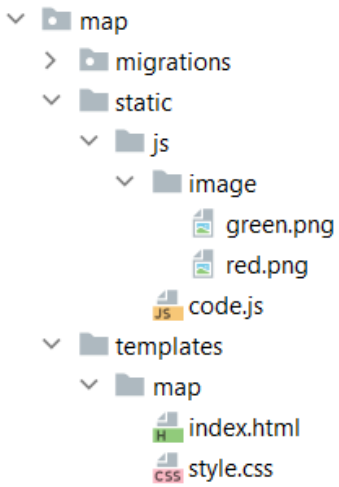


Рисунок 3.26 – Ієрархія папок для шаблонів і статичних файлів

Потрібно підготування сторінку для відображення в веб-додатку карти. Для цього в файлі index.html додаємо CSS та JavaScript, в заголовок документа включаємо посилання на налаштування стилей та функціоналу бібліотеки Leaflet. Представлення цих тегів описано на рис.3.27 нижче.

```
<head>
    <title></title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
          integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A=="
          crossorigin=""/>

    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
            integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA=="
            crossorigin="">
    </script>
</head>
```

Рисунок 3.27 – Leaflet підключення

Реалізація асинхронної роботи в проекті реалізується за допомогою інструменту черги завдань Celery. Яка чудово працює в взаємодії з фреймворком Django, для її використання необхідно лише підключити скрипти доступу в налаштування проекту та підключення до Docker, яке було встановлено раніше.

Переходимо на офіційний сайт Celery та бачимо необхідні скрипти налаштування. Для підключення необхідно створити новий файл в корінній папці проекту з наступним кодом (рис.3.28).

```python
import os

from celery import Celery

# Set the default Django settings module for the 'celery' program.
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'proj.settings')

app = Celery('proj')

# Using a string here means the worker doesn't have to serialize
# the configuration object to child processes.
# - namespace='CELERY' means all celery-related configuration keys
#   should have a `CELERY_` prefix.
app.config_from_object('django.conf:settings', namespace='CELERY')

# Load task modules from all registered Django apps.
app.autodiscover_tasks()


@app.task(bind=True)
def debug_task(self):
    print(f'Request: {self.request!r}')
```

Рисунок 3.28 – Налаштування Celery

В модулі __init.py__, що являється гарантом того, що програма Celery запуститься під час запуску Django, приклад коду на рис.3.29.

```python
# This will make sure the app is always imported when
# Django starts so that shared_task will use this app.
from .celery import app as celery_app

__all__ = ('celery_app',)
```

Рисунок 3.29 – Запуск Celery поруч з Django

Для того щоб підключення карти працювало коректно, необхідно використовувати API для карти, яке можно безкоштовно отримати після реєстрації сайті mapbox.com, далі створити API код, та вставити його в атрибуті integrity тега link для підключення як і стилей так і скриптів, приклад створення відображення сторінки де створено код показано на рисунку 3.30. нижче.

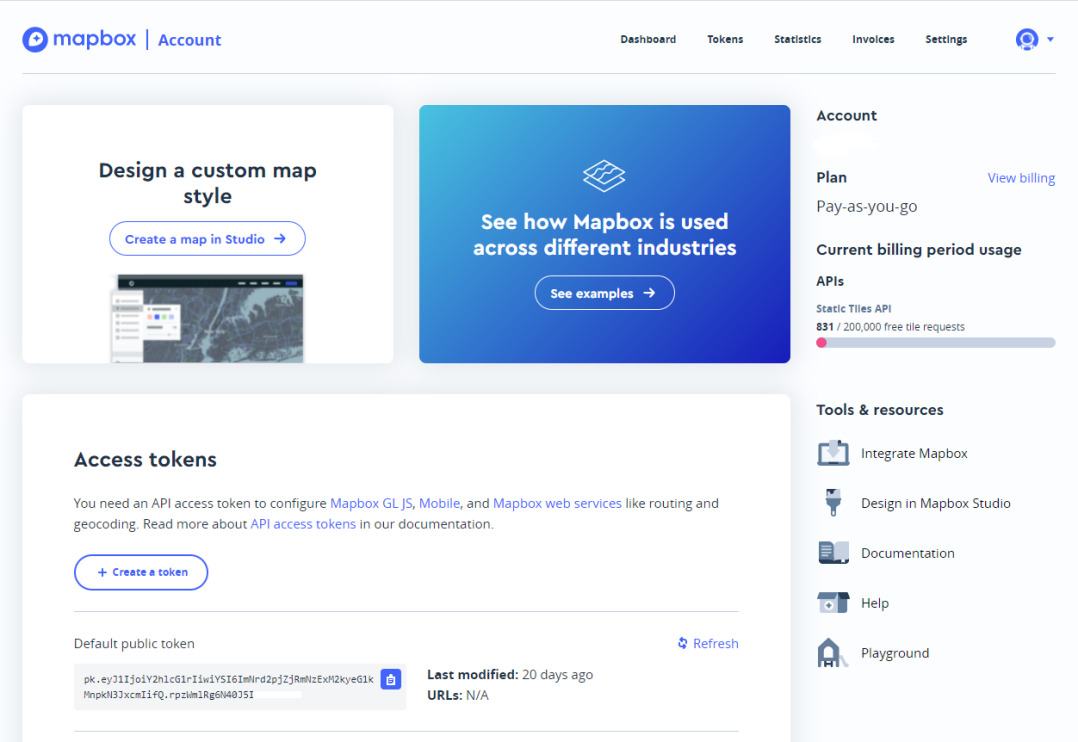


Рисунок 3.30 – API для карти

3.3 Реалізація основного програмного модуля

Додання карти в скрипті відбувається за допомогою об’єкта, який створюється функцією map() за заданням видимих координатів карти при відкритті сторінки. Далі цей об’єкт використовується для додання всіх елементів на сторінку. До об’єкту викликається функція titleLayer() та задає прошарок мапи. Всі маркери задаються функцією marker() та координат широти та довготи, приклад приведений на рис.3.31.

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
    .openPopup();
```

Рисунок 3.31 – Додання маркерів на карту

В файлі code.js прописуємо створення всіх точок, за допомогою координат цих місць на карті, які будуть відображатися інтерфейсі програми та додаємо їх на карту за допомогою функції .addTo(map), описано на рис.1.

```
var top1 = [50.441628, 30.494025]; var marker1 = L.marker(top1).addTo(map);
var top2 = [50.442557, 30.494974]; var marker2 = L.marker(top2).addTo(map);
var top3 = [50.396357, 30.559610]; var marker3 = L.marker(top3).addTo(map);
var top4 = [50.617739, 30.447734]; var marker4 = L.marker(top4).addTo(map);
var top5 = [50.391669, 30.363574]; var marker5 = L.marker(top5).addTo(map);
var top6 = [50.519467, 30.909148]; var marker6 = L.marker(top6).addTo(map);
var top7 = [50.331550, 30.435221]; var marker7 = L.marker(top7).addTo(map);
var top8 = [50.478396, 30.408997]; var marker8 = L.marker(top8).addTo(map);
var top9 = [50.469784, 30.447596]; var marker9 = L.marker(top9).addTo(map);
var top10 = [50.494739, 29.662038]; var marker10 = L.marker(top10).addTo(map);
var top11 = [50.390421, 30.478696]; var marker11 = L.marker(top11).addTo(map);
var top12 = [50.532786, 30.657715]; var marker12 = L.marker(top12).addTo(map);
var top13 = [50.396357, 30.559610]; var marker13 = L.marker(top13).addTo(map);
var top14 = [50.441628, 30.494025]; var marker14 = L.marker(top14).addTo(map);
var top15 = [51.036704, 31.873678]; var marker15 = L.marker(top15).addTo(map);
var top16 = [50.586671, 30.488328]; var marker16 = L.marker(top16).addTo(map);
```

Рисунок 3.32 – Точки мапи

За допомогою даних змінних, ми зробили візуалзацію додатку, додавши всю інщоврмацію про станціїї мережі, аби додати лінії мережі необхідно використати функцію polydgon(), де задаються дві точки, які необхідно з’єднанти.

На рис. 3.33 видно як відображається карта, з усіма внесеними в систему даними. Видно де знаходяться вузли мрежі та які з них з’єднуються між собою.

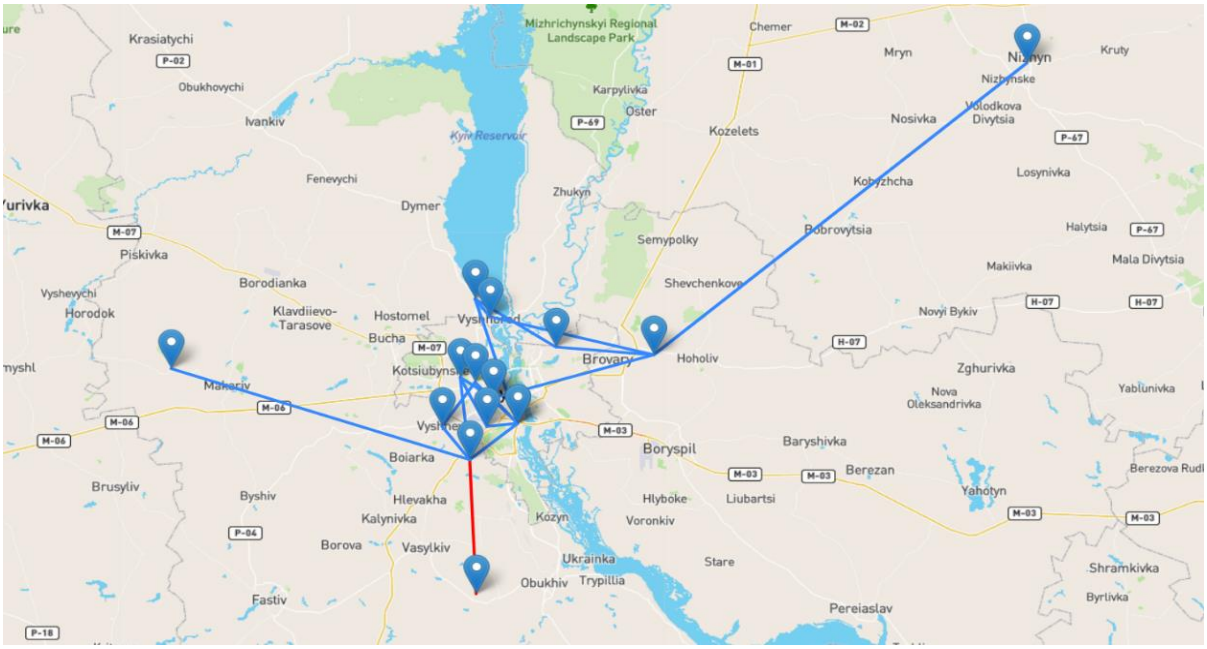


Рисунок 3.33 – Карта програми

Дана візуалізація дозволяє нам нагллядно показати, шо дана мереже прадставляється у вигляді графу, який ми можемо використовувати у розрахунках, як структуру даних.

Ці дані дозволяють нам визначити мережу в програмі за допомогою словника двумірного масиву, де ключом є вершина від якої йде зв’язка, перший елемент масива – вершина з якою зв’язаний цей шлях. Другим масивом в двумірному масиві – всі вершини через які проходить даний шлях. Останнім (третім) елементом масиву є коефіцієнти готовності, тобто показнити того чи справно працюють дані лінії в заданому шляху. Приклад представлення даного словника показано на рис.3.34.

```
paths = {
    1: [[13], [1, 2, 13], [0.99, 0.99]],
    2: [[6], [2, 14, 6], [0.99, 0.99]],
    3: [[1], [3, 11, 8, 1], [0.99, 0.99, 0.99]]
```

Рисунок 3.34 – Словник шлях мережі

Заданий мережою граф можна представити у вигляді матриці сумжності та вже від цього відштовхуватися для розрахунів для пошуку зв’язності мреежі. Матриця суміжності представлена на рис. 3.35.

0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
0	0	0	**1**	1	0	0	0	1	0	1	0	0	0	0	0
0	0	**1**	0	0	0	0	0	0	0	0	1	0	0	0	**1**
0	0	**1**	0	0	0	**1**	0	**1**	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**1**	**1**
0	0	0	0	**1**	0	0	1	0	1	0	0	1	0	0	0
1	0	0	0	0	0	**1**	0	0	0	**1**	0	0	0	0	0
0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0
0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	1	0	0	0	0	0	0	**1**	0	0
0	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	0	0
0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0

Рисунок 3.35 – Матриця суміжності

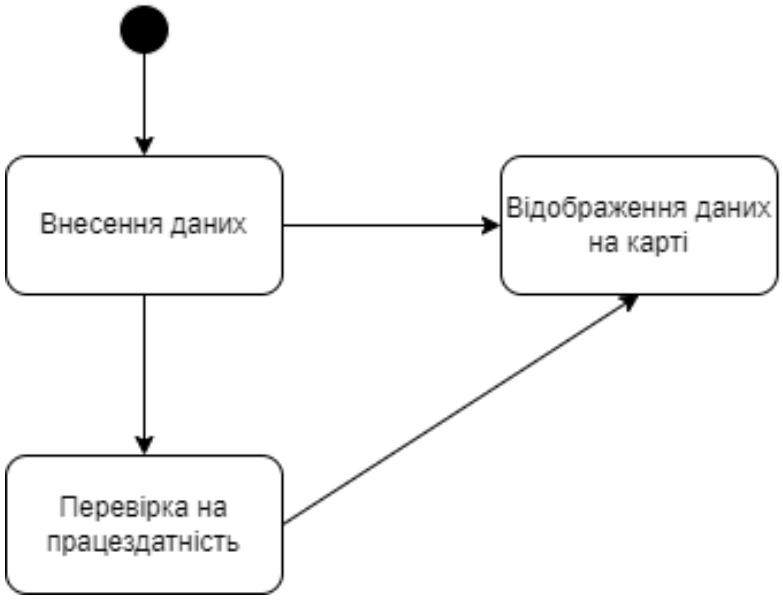


Рисунок 3.36 – Блок-схема діаграми стану

Додатку необхідні ввідні дані, по яким далі проводяться всі обчислення, після чого вони відображаються в додатк, далі отримані дані розраховуються заданими в алгоритмі формулами, система перевіряються на працездатність та відображаються на мапі, як описано на рис. 3.36 на блок-схемі діаграми станів.

Для автоматизації розрахунку зв'язності мережі розроблена методика у вигляді наступного алгоритму дій:

Отримуємо необхідні дані з мережі (час, який система знаходиться в стані роботи t_r; час інтервалу, поки мережа в стані очікування технічного обслуговування $t_{m12} = [t_{m1}, t_{m2})$; час інтервалу, поки мережа в стані очікування від відмови до виявлення $t_{m01} = [t_{m0}, t_{m1})$; час інтервалу поки система ремонтується $t_{m23} = [t_{m2}, t_{m3}))$

Вираховуємо тривалість повного відновлення:

$$t_m = t_{m01} + t_{m12} + t_{m23}$$

Розраховуємо коефіцієнт готовності каналу:

$$k = \frac{t_r}{t_r + t_m}$$

Перевіряємо зв'язність l-ого шляху, для попередження порушення роботи мережі, що може призвести до відмови каналів зв'язку:

$$k_{ij}^1 = \prod_{\forall a \in \mu_{ij}^l} k_a$$

Далі перевіряємо зв'язність шляхів k_{ij}:

$$k_{ij} = k_{ij}^{max} = 1 - \prod_{\forall \mu_{ij}^1 \in \mu_{ij}} (1 - k_{ij}^1)$$

Повна блок-схема алгоритму представлена на рис.3.37 нижче.

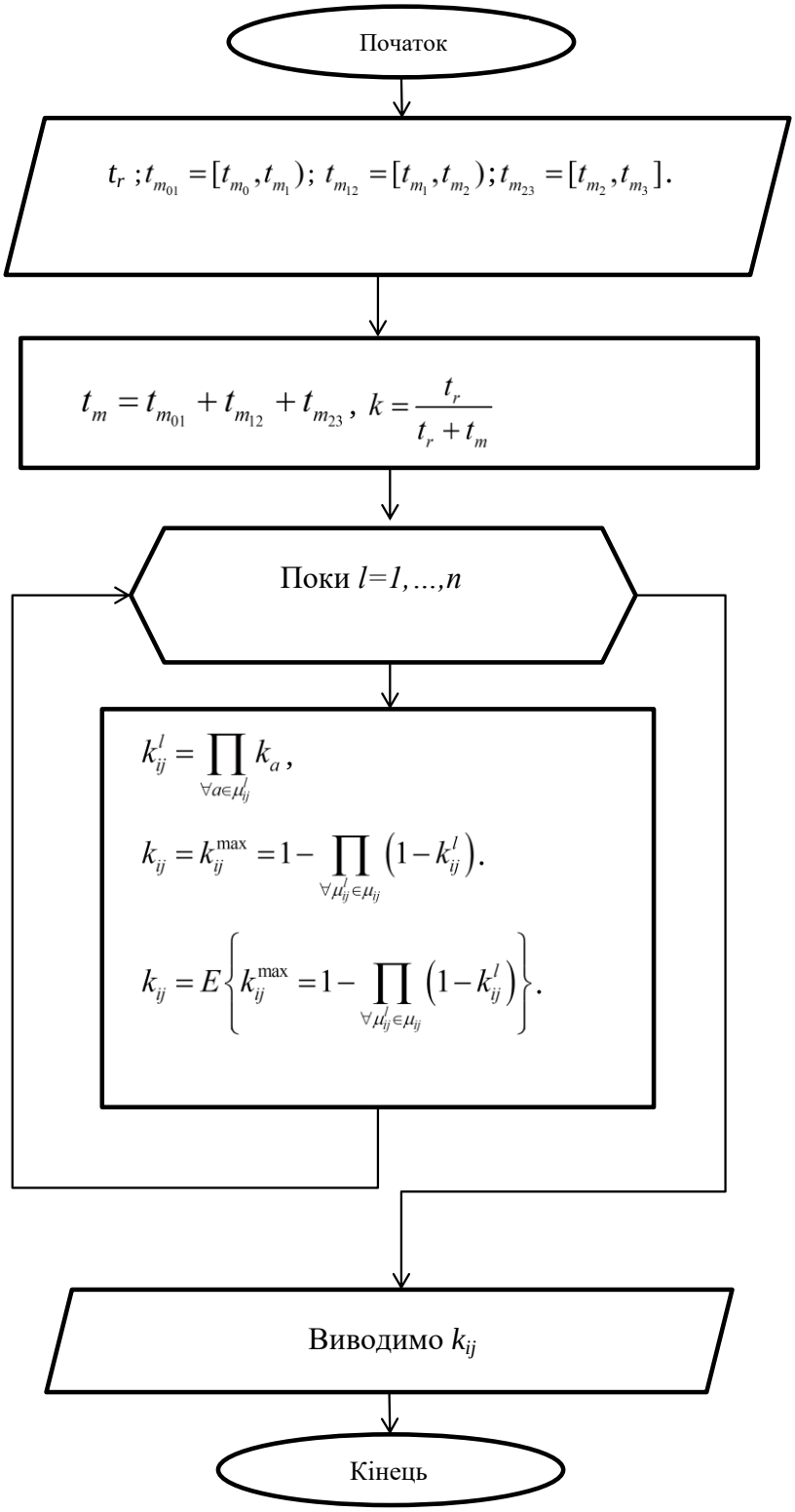


Рисунок 3.37 – Блок-схема алгоритму

Система моніторигу була зроблена на замовлення ПрАТ «НЕК «Укренерго». Головним технічним завданням було використання мови програмування Python, яка дозволить зробити систему швидкою та зручною для використання і модернізації в майбутньому.

Основим обчислюваним модулем додатку є скрипт, який реалізує роботу алгоритму та визначає чи є мережа зв’язною, представлено на рис.3.38.

```
k1ij = []
total = 1
for key in paths:
    count = 0
    while count < len(paths.get(key)[2]):
        total *= paths.get(key)[2][count]
        count += 1
    k1ij.append(total)

total = 1
for column in range(len(k1ij)):
    total *= (1 - k1ij[column])
total = 1 - total
```

Рисунок 3.38 – Код для розрахунку зв'язності мережі

ВИСНОВКИ

В ході магістерської роботи було проаналізовано математичні моделі, які допоможуть у вирішення поставленої задачі. Основною темою данних досліджень є визначення функціональної стікості інформаційних мереж. Було досліджено наукові роботи, які можуть допомогти забезпечити відмовостійкість мережі та автоматизувати віирішення проблем до того як це призведе до критичної поломки системи. Дослідження аналогічних існуючих систем показало, що на теперішній час не існує повного теоретичного вирішення даної задачі. Тому завданням магістрерської роботи було удосконалити математичну модель для визначення працездатної мережі в режимі реального часу та розробити програмну систему моніторингу мережі, яка б відповідала всім поставленим у завданні умовам.

Результати магістерської роботи:

- проведенно дослідження існуючих аналогів систем моніторингу інформаційних мереж;
- досліджено існуючу наукову складову, яка показала, що більшість обчислень мають проблеми обчислень функціональної стійкості мережі та є досить складними для реалізації, що робить їх не актуальними для інформаційних мереж;
- у взятій за основу математичній моделй удосконалено розрахунки зв'язності мережі, за допомогою який можна визначити пошкодження на лінії мережі;
- розроблено геоінформаційну систему моніторингу, яка працює в режимі онлайн. Обчислення відбуваються за алгоритмом, в основі якого лежать удосконалені математематичні розрахунки, за допомогою яких можна визначити коли мережа несправна та в якому саме місці відбулася поломка.

Програмний продукт моніторингу мережі було розроблено для ПрАТ «НЕК «Укренерго», за заданим стандартом, за допомогою якого були поставленні вимоги до розроблюваної системи.

Інструментами розробки даної системи було обрано мову програмування Python та фрейморк для веб-розробки Django. За допомогою цих технологій було розроблено серверну частину системи та реалізовано функціональну частину проекту. Інтерфейс користувача додатку було створено за допомогою веб-засобів, графічне представлення карти створено за допомогою відкритої бібліотеки для мови JavaScript – Leaflet, де додано всі вузли системи у вигляді точок на карті.

Розроблена система має ряд переваг між схожими на рингу програмами для моніторингу інформаційних систем. Відображення карти з усіма вузлами мережі та ліній з'єднань. В разі виникнення поломки система сповістить адміністратора мережі про поломку та перекине передачу даних на інший вузол.

Дану систему можна використовувати в сфері інформаційних та телекомунікаційних мереж для повідомлення про помилки та їх усунення. В подальшому систему можна модернізувати додавши використання штучного інтелекту, для того аби можливі помилки можна було передбачити до моменту їх виникнення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Жебка В. В. Моніторинг сталості інформаційно-телекомунікаційної системи і опрацювання заходів її захисту від небезпек / В. В. Жебка, П. В. Анахов // Метрологія та прилади. – 2021. – №1(87). – С. 23-29;
2. Козак М. М. Лінійні споруди зв'язку / М. М. Козак. Під ред. С. Б. Доброчинського, Г. М. Петрунчака. – Вінниця: 2009. – 317 с;
3. Коваленко Ірина Структурна живучість та надійність телекомунікаційних мереж / Ірина Коваленко // Новітні інформаційні системи та технології. – Полтава: ПНТУ, 2016. – Т. (5)
4. Марк Лутц. Программирование на Python / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2011. — Т. II;
5. Додонов А.Г., Ландэ Д.В. Живучесть информационных систем. – К.: Наук. думка, 2011. – 256 с.
6. Djangoproject.com [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2005-2021, Django Software Foundation] – Режим доступа: https://docs.djangoproject.com/en/3.0/;
7. Введение в теорию живучести вычислительных систем / Додонов А .Г., Кузнєцов М. Г., Горбачик Е .С. Отв. ред. Гуляев В. А. — К.: Наук. думка, 1990. — 184 с.
8. У. Чан, П. Биссекс, Д. Форсье. Django. Разработка веб-приложений на Python = Python Web Development with Django / пер. с англ. А. Киселёв. — СПб.: Символ-Плюс, 2009;
9. Половко О. М. Основи теорії надійності. 2-ге вид., перераб. та дод. / О. М. Половко, С. В. Гуро;
10. Барабаш О. В. Оцінювання показника функціональної стійкості графа структури розгалуженої інформаційної мережі;
11. Барабаш О. В., Кравченко Ю. В. Функціональна стійкість — властивість складних технічних систем. Зб. наук. пр. НАОУ. Бюл. № 40. — К.: НАОУ, 2002;

12. Королев А. В., Кучук Г. А., Пашнею А. А. Адаптивная маршрутизация в корпоративных сетях. — Харьков: ХВУ, 2003;
13. Python.org [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2001-2021, Python Software Foundation] – Режим доступа: https://docs.python.org/3/.
14. Лаврут О.О. Дослідження якості управління потоками інформації у моделі військової телекомунікаційної мережі представленій в тензорному вигляді / О.О. Лаврут // Військово-технічний збірник.- Львів: АСВ, 2015.- Вип. 12/2015. – С. 27-33.
15. Docker.com [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2013-2021 Docker Inc] – Режим доступа: https://docs.docker.com/desktop/;
16. Павленко Л. А., Геоінформаційні системи, Харків. Вид. ХНЕУ, 2013;
17. Зайченко Ю.П., Гонта Ю.В. Структурная оптимизация сетей ЭВМ. – К.: Техніка, 1986. — 168 с.
18. Уилсон Р. Введение в теорию графов: Пер. с англ. — М.: Мир, 1977. — 208 с.
19. Королев А. В., Кучук Г. А., Пашнею А. А. Адаптивная маршрутизация в корпоративных сетях. — Харьков: ХВУ, 2003. — 224 с.
20. Надежность и живучесть систем связи / Дудник Б. Я., Овчаренко В. Ф. и др. Подред. Дудника Б. Я. — М.: Радио и связь, 1984. — 216 с.
21. Стекольников Ю.И.Живучесть систем –СПб.:Политехника, 2002. –155 с.
22. Celeryproject.org [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2009-2021, Ask Solem & contributors] – Режим доступа: https://docs.celeryproject.org/en/stable/django/first-steps-with-django.html;
23. Князева Н. А. Алгоритмы оценки структурной живучести инфокоммуникационной сети // Сучасні інформаційно-комунікаційні технології. VIII наук.-техн. конф.: збірник тез. –К.: 2012. –С. 192–193.
24. Стеклов В.К., Беркман Л.Н. Проектування телекомунікаційних мереж: Підруч. для студ. вищ. навч. зал. за напрямком «Телекомунікації» / за ред. В.К. Стеклова. – К.: Техніка, 2002. - 792 с.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Магістерська робота

«РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ СТАНУ МЕРЕЖІ НА ОСНОВІ ТЕОРІЇ СТІЙКОСТІ»

Виконала: Чепур Марина Костянтинівна
Керівник: Жебка Вікторія Вікторівна

Київ - 2021

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: автоматизація процесу визначення стійкості мережі за допомогою геоінформаційної системи для моніторингу стану мережі на основі теорії стійкості.

Об'єкт дослідження: процес управління мережею.

Предмет дослідження: геоінформаційна система моніторингу стану мережі.

ІСНУЮЧІ СИСТЕМИ МОНІТОРИНГУ СТАНУ МЕРЕЖІ

В даній роботі мова йде про ГІС моніторингу, за допомогою яких можна відобразити стан мережі та відстежувати наявні в ній пристрої та з'єднання.
Аналогічних інформаційних систем на основі теорії стійкості в вільному доступі немає.
Із програм представлених на сьогодняшній час, які дозволяють здійсювати моніторинг мережі представлені такі Cacti, Nagios, Icinga, LanState, Nedi, Ntop, Zabbix, Observium, TheDude.

	Оновлення даних в реальному часі	Автоматична побудова графу мережевої карти	Зручний інтерфейс	Кросплатформеність	Широкі можливості відображення даних
gisEnergo	+	+	+	+	+
Cacti	-	+	+/-	+	+
Nagios	-	-	-	+	+
Icinga	+	-	+	+	+
LanState	+	+	+	-	-
Nedi	-	+	+	-	+
Ntop	-	+	+	+	+
Zabbix	-	-	+	+/-	+
Observium	+	-	+	+	+
TheDude	+	+	-	-	+

Вони не відповідають заданим умовам, тому було прийнято рішення розробити програмну систему моніторингу мережі, яка б відповідала всім поставленим у завданні критеріям: автоматичне заповнення даних для відображення карти та оновлення даних мережі.

Відмовостійкість мережі

Основна властивість відмовостійкості це те, що кінцевий користувач до останнього не знає про те, що система або деякі її компоненти вийшли з ладу. Це свідчить про те, що відмовостійка мережа автоматично змінює налаштування системи, коли відбувається якийсь збій. Для цього програмне забезпечення системи шукає резервні шляхи, які допоможуть мережі все ще працювати, не роблячи повне загасання мережі. Негативні наслідки які можуть впливати на відмовостійкість мережі:

ФОРМУЛИ ДЛЯ РОЗРАХУНКУ ЗВ'ЯЗНОСТІ

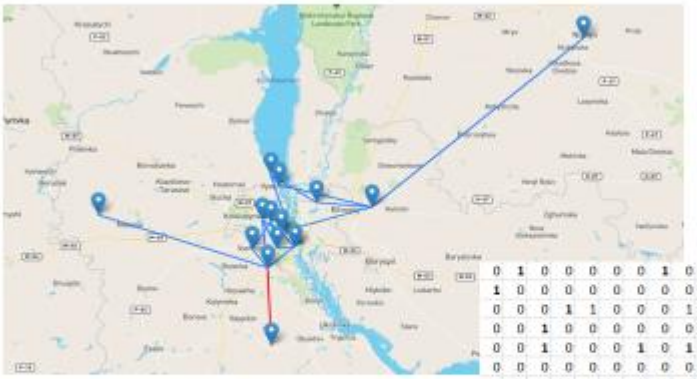
Модель інформаційної системи, представлена у вигляді неорієнтованого зв'язного графу:

$$G(V,E), \quad v_i \in V \quad , e_{ij} \in E, \qquad i,j = \overline{1,n} \qquad (1)$$

Для визначення надійності мережі, між умовними вузлами a_s і a_t, при умові що задано купа шляхів, які можуть бути використанні для заданого зв'язку. Значення вірогідності безвідмовної роботи всіх ребер - p_{ij}. Знаходження значення надійності (ρ_{st}^k), заданого шляху (μ_{st}^k) , вираховується за формулою:

$$\rho_{st}^k = \prod_{\forall b_{ij} \in \mu_{st}^k} \rho_{ij}$$

ГРАФ МЕРЕЖІ



0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ

Науковий результат:
розробка геоінформаційної системи для моніторингу стану мережі на основі теорії стійкості.

Практичний результат:

КОД ПРОГРАМНОГО МОДУЛЯ

```
paths = {
    1: [[13], [1, 2, 13], [0.99, 0.99]],
    2: [[6], [2, 14, 6], [0.99, 0.99]],
    3: [[1], [3, 11, 8, 1], [0.99, 0.99, 0.99]]
```

```
k1ij = []
total = 1
for key in paths:
    count = 0
    while count < len(paths.get(key)[2]):
        total *= paths.get(key)[2][count]
        count += 1
    k1ij.append(total)

total = 1
for column in range(len(k1ij)):
    total *= (1 - k1ij[column])
total = 1 - total
```

Висновки

Результати магістерської роботи:

- проведенно дослідження існуючих аналогів систем моніторингу інформаційних мереж;
- проаналізовано математичні моделі, які допоможуть у вирішення поставленої задачі.
- удосконалено технологію пошуку зв'язності вузлів мережі на основі стійкості мережі.
- розроблено геоінформаційну систему моніторингу, яка працює в режимі онлайн.

Статті:
Стаття написана у співавторстві зі студенткою групи ПДМ-61, Балашовою Єлизаветою
Стаття №3 журналу Телекомунікаційні та інформаційні технології. ДУТ.

Тези доповідей:
Тези на тему «Розробка геоінформаційної системи моніторингу стану мережі на основі теорії стійкості», XIII науково-технічна конференція «ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ», ДУТ

ДЯКУЮ ЗА УВАГУ!