МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
Т.в.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях»

Здобувача групи ІТ - 91 Гузя Олександра Юрійовича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр ГУЗЬ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник д.т.н., проф., Лавров Є.А. _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Т. в. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Гузь Олександр Юрійович

1 Тема роботи *Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях*

керівник роботи *Лавров Євгеній Анатолійович, д. т. н., проф.*,

затверджені наказом по університету від «29» травня 2023 р. №0588-VI

2 Строк подання студентом роботи «07» червня 2023 р.

3 Вхідні дані до роботи
Методичні вказівки до виконання та оформлення кваліфікаційних робіт бакалавра
Структура пояснювальної записки до дипломного проекту бакалавра

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області
Моделювання та проектування
Розробка інформаційної системи
Висновки
Список використаних джерел

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Актуальність
Мета дослідження
Вимоги до системи
Планування робіт

Моделювання роботи системи
Діаграма варіантів використання
Математична модель
Логічна модель даних
Технології реалізації
Демонстрація роботи
Висновки

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	17.04.23 – 22.04.23	
2	Постановка мети та задач	22.04.23 – 25.04.23	
3	Вибір технологій реалізації	25.04.23 – 28.04.23	
4	Вибір алгоритму реалізації	28.04.23 – 30.04.23	
5	Створення бази даних	30.04.23 – 06.05.23	
6	Тестування бази даних	06.05.23 – 08.05.23	
7	Розробка графічного інтерфейсу користувача	08.05.23 – 13.05.23	
8	Написання коду програми	13.05.23 – 30.05.23	
9	Тестування роботоспроможності системи	30.05.23 – 04.06.23	

Студент _____ Олександр ГУЗЬ
(підпис)

Керівник роботи _____ Євгеній ЛАВРОВ
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 21 найменням, додатків. Загальний обсяг роботи − 111 сторінок, у тому числі 35 сторінок основного тексту, 3 сторінки списку використаних джерел, 73 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці інформаційної система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях. В роботі проведено аналіз предметної області, актуальності розробки, аналіз продуктів-аналогів. У роботі виконано всі задачі, поставлені у відповідності до вимог до програмного продукту. Результатом проведеної роботи є інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях. Практичне значення роботи полягає у автоматизації процесу пошуку альтернативного шляху живлення у разі аварійної ситуації в електричній мережі.

Ключові слова: java, десктопний додаток, електрична мережа, mysql

ЗМІСТ

ВСТУП

Актуальність. Сучасний світ важко уявити без електрики. Електрику проводять у найвіддаленіші місця нашої планети, вона забезпечує здійснення виробничої, промислової чи побутової діяльності.

Очевидно, що відсутність електроенергії може стати великою проблемою на підприємствах, а непрацездатність яких, в свою чергу, сильно впливає на економіку всієї країни. До того ж навіть короткочасна відсутність світла в людських будинках викликає досить відчутний дискомфорт та невдоволення мешканців міст. Однак, динаміка зносу ліній електропередач (і мереж загалом) останні роки зростає. Продовжується скорочення обсягів робіт з нового будівництва, реконструкції та технічному переозброєнню при деякому збільшенню обсягів з капітального ремонту. Для того, щоб підвищити термін дії обладнання, необхідно проводити капітальний ремонт електроустаткування. Але цей захід може призводити до зменшення технічного рівня мереж. Необхідно застосовувати абсолютно нові системи автоматизації, які можуть дозволяти домагатися якіснішого обслуговування та управління аваріями у розподілених електричних мережах.

Керує розподіленими електричними мережами диспетчер (іноді називають «оператор»). Для підвищення якості прийнятих рішень диспетчерський персонал повинен оперувати лише максимально надійною та повною інформацією. Він опрацьовує величезний потік інформації. Проте в даний час, коли стрімко розвиваються комп'ютерні обчислювальні потужності, з'явилася можливість істотно змінити підхід до виявлення аварій, прискорити темпи їх усунення.

Об’єкт дослідження. Діяльність оператора системи управління електричною мережею при аварійних ситуаціях.

Предмет дослідження. Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях.

Мета дослідження. Розробити інформаційну систему підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях, яка

допомагає диспетчеру зменшити кількість інформації, яку необхідно тримати в голові, а також виступає як повноцінний порадник або експерт.

Для досягнення мети проекту необхідно виконати наступні задачі:

1) Визначити актуальність роботи та дослідити предметну область;
2) Виконати огляд програмного забезпечення для реалізації системи;
3) Виконати огляд і порівняння існуючих алгоритмів;
4) Розробити і створити базу даних для довготривалого збереження інформації;
5) Спроектувати та реалізувати інформаційну систему;
6) Перевірити роботоспроможність системи.

Апробація. Матеріали дослідження доповідались на міжнародній конференції IMA 2023.

Публікації. За матеріалами дослідження опубліковано одні тези доповіді (копія публікації наведена в додатку В).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд застосування інформаційних технологій для підтримки електроенергетики

Електрична енергія має величезне значення у житті та діяльності людей, відіграючи важливу роль у побуті сучасної людини, супроводжуючи її всюди. Сьогодні люди масово користуються ліфтами, побутовою технікою, банкоматами, комп'ютерами, кишеньковими гаджетами. Усі ці та багато інших речей настільки полегшують наше життя, що люди вже просто не можуть уявити своє існування без них. Проте вони не здатні функціонувати без постійного електропостачання. При цьому кількість електроприладів, що оточують нас, не стає меншою, вона постійно збільшується з року в рік, а, відповідно, відбувається зростання споживання енергії [1]. Електричне світло, тепло, гаряча вода, такі необхідні для повноцінного затишку та комфорту в будинку, також надходять до нас завдяки електроенергії, і навіть короткочасна відсутність світла в домі здатна викликати значне невдоволення мешканців будинку [2].

Електрична мережа - сукупність електроустановок, призначених для передачі та розподілення електроенергії від електростанції до споживача. Основні її елементи наступні:

- генерувальні станції;
- високовольтні лінії електропередавання та розподільні лінії які доправляють енергію до розподільних пристроїв підстанцій;
- ввідні пристрої, ввідно-розподільні пристрої;
- головні розподільні щити.

Це все досить складне обладнання, і під впливом різних чинників воно може виходити з ладу.

Аварія в енергосистемі - порушення нормального режиму всієї енергетичної системи або значної її частини. Причини виникнення аварій можуть бути різні,

наприклад, причини, пов'язані з пошкодженням обладнання або тривалою перервою в електропостачанні споживачів. Для стабільної роботи енергосистеми необхідно оперативно усувати наслідки аварій та намагатися проводити заходи для зменшенню кількості аварій в майбутньому.

Очевидно, що диспетчерському персоналу доводиться обробляти величезну кількість інформації. Через це з'являються труднощі на етапі вибору альтернативного маршруту. Необхідно враховувати економічні витрати, людські ресурси, навантаження на підстанції, пріоритет кожної підстанції, відсутність подвійного включення, скласти заявку для групи реагування тощо. Через великий об’єм інформації, в роботі оператора регулярно виникають помилки, які можуть призвести до більш тяжких наслідків ніж відсутність електроенергії у споживачів [3].

Використання інформаційних технологій значно покращує ефективність та надійність електроенергетики. Вони відіграють ключову роль у розвитку сучасних енергетичних мереж та сприяють переходу до більш стійкої та екологічно чистої енергетики. Існує багато областей електроенергетики, у яких інформаційні технології є особливо важливими.

Системи моніторингу [4] та управління дозволяють операторам електромереж отримувати в реальний час інформацію про стан мереж, навантаження, збої та інші важливі параметри. Це дозволяє оперативно реагувати на проблеми та вживати заходів для запобігання вимкненню електроенергії. Інформаційні технології дозволяють впроваджувати системи керування навантаженням, які оптимізують використання енергії та розподіляють навантаження між різними джерелами та споживачами. Це дозволяє більш ефективно використовувати доступну енергію та знижувати пікове навантаження.

Інформаційні технології допомагають ефективно інтегрувати відновлювані джерела енергії, такі як сонячна та вітрова енергія, у мережі. Вони забезпечують моніторинг та управління процесами генерації енергії, а також інтеграцію цих джерел із традиційними мережами електропередачі [5].

Розвиток інформаційних технологій сприяє створенню смарт-мереж, які поєднують споживачів, виробників та дистриб'юторів електроенергії в єдину

мережеву інфраструктуру. Смарт-мережі дозволяють більш ефективно управляти енергопотоками, надавати споживачам інформацію про споживання та стимулювати енергозбереження [6].

Інформаційні технології відіграють важливу роль у забезпеченні безпеки систем електроенергетики. Вони дозволяють виявляти та запобігати кібератакам, моніторити системи на наявність уразливостей та забезпечувати захист від несанкціонованого доступу до критичної інфраструктури [7].

Очевидно, що без допомоги інформаційних технологій весь процес на формування заявки та прийняття рішення щодо резервного включення займає тривалий час. Тому, щоб домогтися налагодженого і своєчасного реагування на аварії необхідно застосовувати комп'ютерні обчислювальні систем. Розроблювана інформаційна система здатна зменшити кількість інформації, яку необхідно тримати в голові диспетчеру, а також виступає як повноцінний порадник або експерт .

1.2 Аналіз програмних продуктів – аналогів для підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях

Розроблювана програма оперує над даними про електромережі. Найзручніше зберігати ці дані в таблицях. Для цього можуть бути використані табличні редактори, такі як Microsoft Excel [8] або OpenOffice Calc [9]. Ці програми мають деякі переваги над розроблюваною системою:

- більше можливостей саме форматування даних;
- потенціально можна отримати функціонал, який не передбачає розроблюваний програмний продукт, але його потрібно програмувати самостійно (наприклад, для вибірки даних з таблиці необхідно прописати спеціальний оператор);
- велика спільнота та багато інформації в інтернеті.

Проте табличні редактори призначені саме для роботи з таблицями, тобто для вирішення обчислювальних завдань (економічних, бухгалтерських, інженерних, статистичних) на великих масивах даних, що подаються у табличній формі. Тому має наступні недоліки:

- файли можуть пошкодитися, а тому потрібно зберігати копію;
- користувач повинен самостійно слідкувати за коректністю даних в таблицях;
- щоб дані були загальнодоступними для інших користувачів файл слід зберігати в хмарному середовищі.

А отже розроблюваний продукт має ряд беззаперечних переваг:

- надається інтерфейс для роботи з даними про мережі;
- цілісність таблиць гарантована;
- всі необхідні операції виконуються дуже швидко, нічого додатково програмувати користувачу не потрібно;
- надаються пропозиції щодо альтернативного шляху живлення підстанцій (у разі використання табличних редакторів необхідно заносити дані про всі можливі конфігурації підстанцій та ліній електропередач, а також самостійно програмувати логіку).

Можна використовувати системи управління реляційними базами даних. Тоді всі операції з даними здійснюються за допомогою запитів. Проте розроблювана програма разом з тим взаємодіє з базою даних, так як дані зберігаються там. Але користувачу надається інтерфейс для роботи з ними, ніяких запитів писати не потрібно.

Так як розподілена електрична мережа за своєю структурою нагадує зважений граф, можна використовувати програмне забезпечення або веб-сервіси для роботи з графами. Зазвичай вони просто надаватимуть більше можливостей візуалізації або редагування, проте для використання таких систем як аналогів розроблюваного програмного продукту користувачу необхідно хоча б поверхнево розуміти теорію графів. До того ж вони для поставлених задач не передбачені, хоча і використовують подібні алгоритми для роботи з графами.

Також існує спеціалізоване програмне забезпечення. Наприклад, NEPLAN Electricity [10] - це програмний інструмент для аналізу, планування, оптимізації та моделювання електричних мереж. Сильною стороною програмного забезпечення є надзвичайно зручний графічний інтерфейс із великими бібліотеками мережевих елементів, пристроїв захисту та схем керування, що дозволяє користувачеві дуже ефективно виконувати дослідження. Програмне забезпечення має модульну концепцію, базується на міжнародних стандартах, таких як IEC, ANSI, IEEE тощо, і налаштовується для ринків Європи та США. Він використовується в мережах передачі, розподілу, генерації/промислових мереж, серед іншого, для мережевого та інвестиційного планування, якості електроенергії, багатоперіодної оптимізації, налаштування та оцінки захисту, динамічного моделювання (RMS/EMT).

Network Manager Electric [11] - це революційне програмне рішення, оптимізоване для операторів електричних мереж і нових завдань, з якими вони стикаються. Цей програмний продукт для електромереж забезпечує найбільш гнучке та відкрите рішення в галузі, прискорюючи планування та проектування, одночасно підтримуючи весь життєвий цикл мережі шляхом будівництва та технічного обслуговування.

Такі системи мають широкий функціонал, але досить важкі та більш вибагливі до програмного забезпечення та характеристик комп’ютера. Розроблювана програма надає простий та інтуїтивно зрозумілий графічний інтерфейс роботи з мережею і пропозиції для побудови альтернативного маршруту живлення електропідстанцій.

1.3 Постановка задач дослідження

Метою даного дослідження є розробка інформаційної системи зі зручним графічним інтерфейсом для підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях.

Основні вимоги до створюваного програмного продукту є наступними:

– додавання, видалення та редагування електричних мереж;

- вказання аварійних ситуацій в доданих електричних мережах;
- надання пропозицій що до альтернативного шляху живлення електропідстанцій шляхом переведення ліній електропередач зі стану «Резерв» у стан «Увімкнути»;
- зручний графічний інтерфейс;
- довготривале зберігання інформації про електричні мережі в базі даних для подальшого аналізу та звітності.

Для досягнення мети проекту необхідно виконати наступні задачі:

- визначити актуальність роботи, цільову аудиторію та дослідити предметну область;
- обрати технології для реалізації системи;
- спроектувати, створити та протестувати базу даних
- виконати моделювання та проектування системи;
- розробити функціонал системи (написання коду програми);
- виконати тестування системи.

Вимоги до проекту в цілому, структури інформаційної системи, видів забезпечення та функціонування системи описані у технічному завданні на розробку проекту (ДОДАТОК А).

1.4 Засоби реалізації інформаційної системи

Для реалізації системи обрана популярна мова програмування Java [12]. Основними її перевагами є:

- просто писати, компілювати, налагоджувати та вивчати;
- Java - це об'єктно-орієнтована мова;
- мова Java не залежить від платформи;
- велике ком’юніті.

Для реалізації довготривалого зберігання інформації використано базу даних MySQL [13]. Основними її перевагами є:

- висока швидкість роботи;
- безкоштовна ліцензія;
- надійна та проста система безпеки;
- підтримка кількох типів таблиць: MyISAM, InnoDB, тощо;
- плагіни, що дозволяють спростити та налаштувати роботу під себе.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Математична модель

Об’єктом дослідження є електрична мережа. Розподілена електрична мережа за своєю структурою нагадує зважений граф (рисунок 2.1), в якому вершини – електропідстанції, а ребра – лінії електропередач.

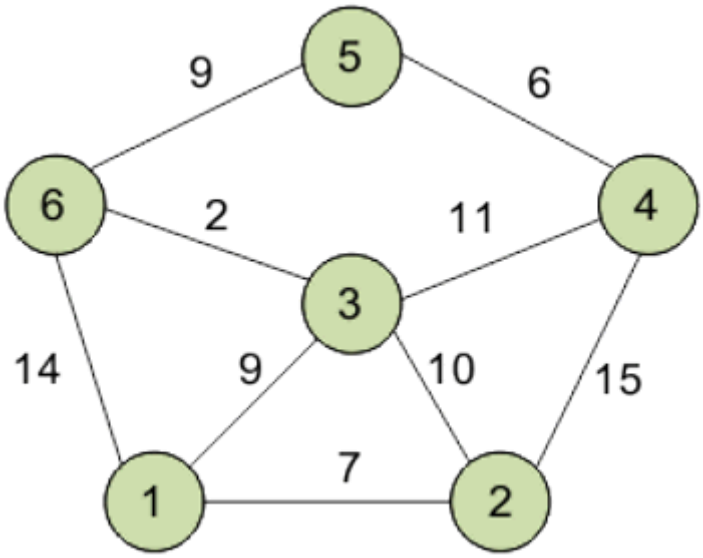


Рисунок 2.1 – Приклад зваженого графа

Існує багато алгоритмів для роботи з графом. В даному випадку кожне ребро має вагу – протяжність ліній електропередач. Так як метою є надання найбільш економічно вигідних шляхів побудови альтернативного маршруту живлення, необхідно мінімізувати відстань від джерела живлення до кожної підстанції для мінімізації втрат енергії.

Для цього використано алгоритм Дейкстри [14] - це алгоритм знаходження найкоротшого шляху у зваженому графі з невід’ємними вагами ребер. Основна ідея алгоритму Дейкстри полягає у поступовому нарощуванні множини відвіданих вершин та оновленні відстаней до інших вершин на основі поточного обраного шляху. Алгоритм використовує пріоритетну чергу для вибору вершини з найменшою

відстанню на кожній ітерації. Результат роботи алгоритму для графа з рис. 2.1 подано на рисунку 2.2:

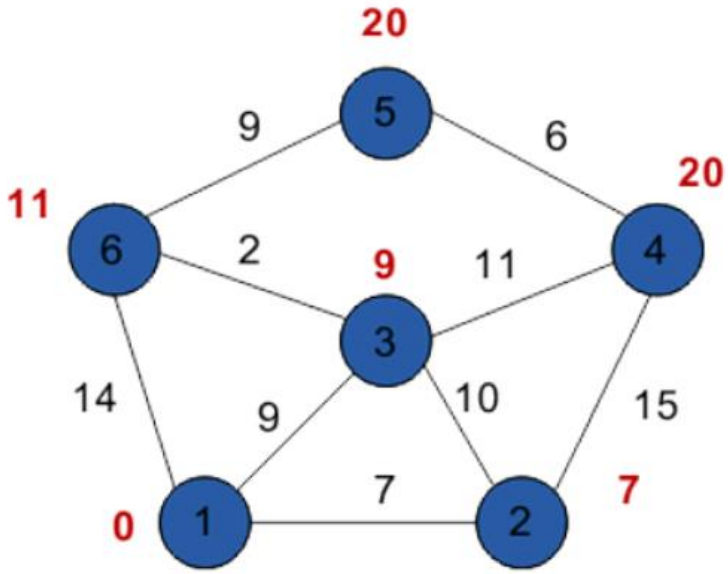


Рисунок 2.2 – Результат роботи алгоритма Дейкстри

Даний алгоритм ефективний при роботі з графами з невеликою кількістю вершин або коли потрібно знайти найкоротший шлях від однієї вершини до решти.

2.2 Функціональне моделювання інформаційної системи в IDEF0

Для демонстрації логічних зв'язків між роботами застосовано нотацію IDEF0 [15] – це одна з основних методологій функціонального моделювання і графічного опису процесів. Вона призначена для формалізації і опису бізнес-процесів. Її особливістю є акцент на ієрархічному представленні об'єктів, що значно полегшує розуміння предметної області. IDEF0 – це дуже проста і водночас наочна мова опису бізнес-процесів.

Контекстна діаграма - найвища діаграма, де об'єкт моделювання представлений єдиним блоком з граничними стрілками. Проаналізувавши постановку задачі та вимоги було виділено такі дані діаграми:

– вхід – дані з таблиць БД, дані про аварійну ситуацію;

– вихід – альтернативний шлях живлення підстанцій;
– управління – регламент роботи персоналу та документація до інформаційної системи;
– механізми – розроблювана інформаційна система, сервер БД MySQL, персонал, Java Virtual Machine.

Контекстна діаграма в нотації IDEF0 зображена на рисунку 2.3:

Рисунок 2.3 – Контекстна діаграма у нотації IDEF0

Будь-який бізнес-процес можна і навіть потрібно деталізувати. Для цього створюється діаграма декомпозиції. У даному випадку робота ділиться на 4 основні етапи:

– вибір електричної мережі;
– додавання аварійної ситуації;
– пошук альтернативного шляху живлення;
– оновлення даних в БД.

Декомпозиція системи представлена на рисунку 2.4:

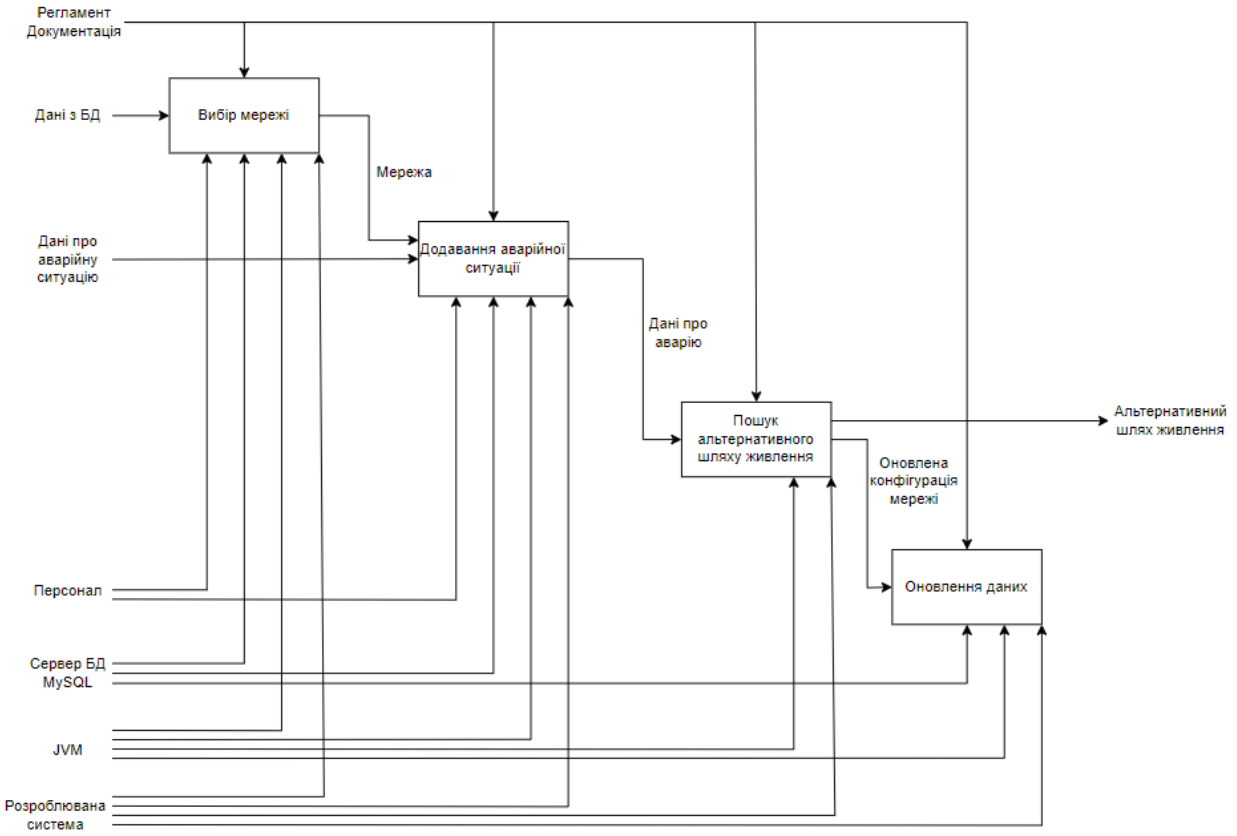


Рисунок 2.4 – Діаграма декомпозиції у нотації IDEF0

На схемі наочно видно, які управляючі елементи і які механізми задіяні та на якому етапі. Так регламент і документація так чи інакше задіяні на кожному етапі. Диспетчер обирає мережу з вже доданих та додає аварійну ситуацію. Дані про існуючі мережі беруться з бази даних, дані про аварійні ситуації додаються в базу даних. Пошук альтернативного шляху живлення виконуються програмою. База даних за потреби оновлюється.

2.3 Моделювання варіантів використання інформаційної системи

Діаграма варіантів використання (use-case diagram [16]) описує функціональне призначення системи. Вона є вихідною концептуальною моделлю системи в процесі її проектування та розробки.

Діаграма варіантів використання в нотації UML представлена на рисунку 2.5

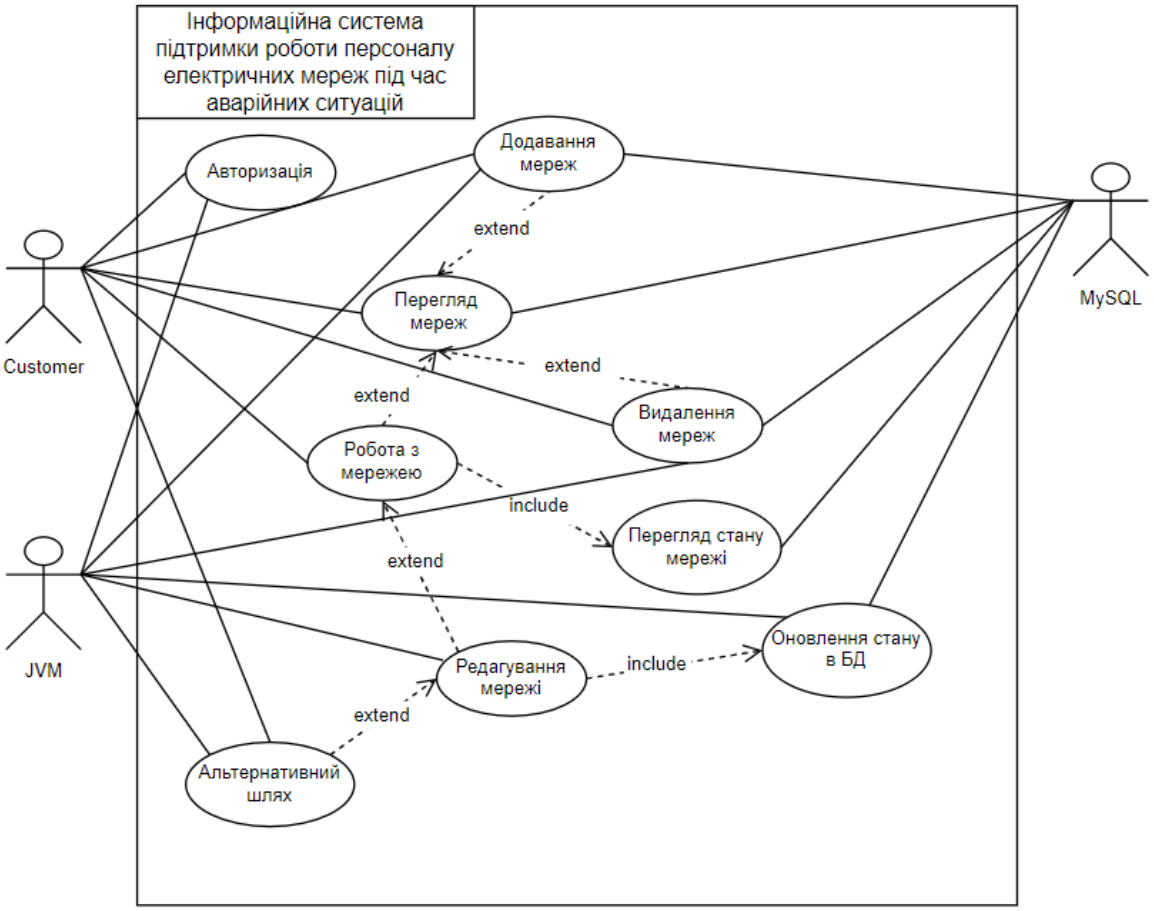


Рисунок 2.5 – Діаграма варіантів використання

Отже, користувач повинен авторизуватися. Також він має можливість переглядати, додавати, видаляти мережі. Можна окремо працювати з деякою мережею. Ця можливість включає зміну параметрів мережі. Після зміни параметрів дані в БД оновлюються. Можна скористатися можливістю програмного пошуку альтернативного шляху живлення мережі у разі аварійної ситуації.

2.4 Діаграма послідовності

Для відображення взаємодії об'єктів впорядкованих за часом, задіяних об'єктів та послідовності надісланих повідомлень використано діаграму послідовності [17].

На рисунку 2.6 представлена діаграма послідовності інформаційної системи:

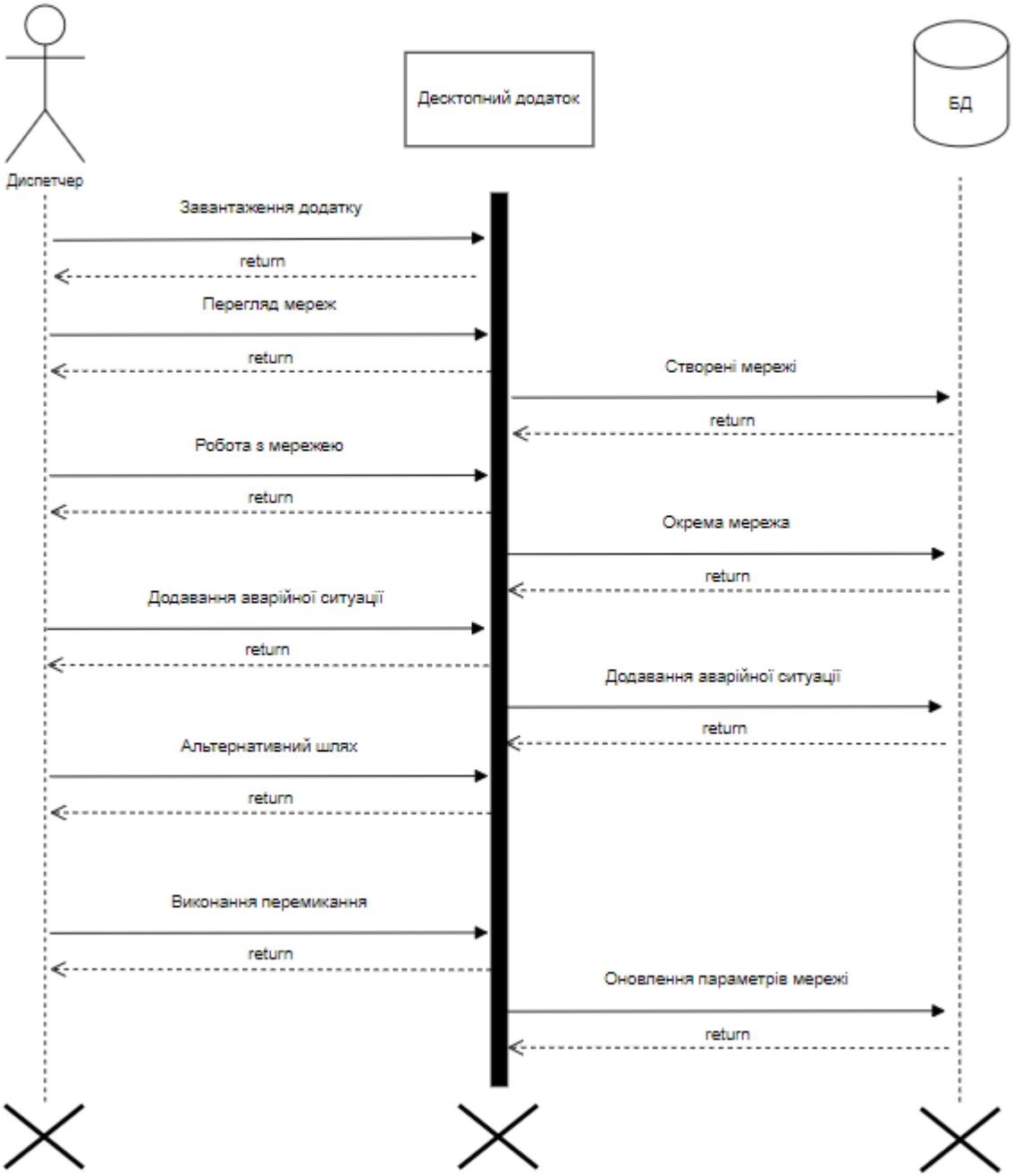


Рисунок 2.5 – Діаграма послідовності

2.5 Діаграми діяльності

Щоб показати як потік управління переходить від одного виду діяльності до іншого, застосовуються діаграми діяльності [18]. При цьому увага фіксується саме на результаті діяльності, Результат може призвести або повернення певного значення, або до зміни стану системи.

На рисунку 2.6 представлена діаграма діяльності для огляду створених мереж або аварійних ситуацій. Після авторизації відкривається відповідне вікно з усіма мережами та аваріями.

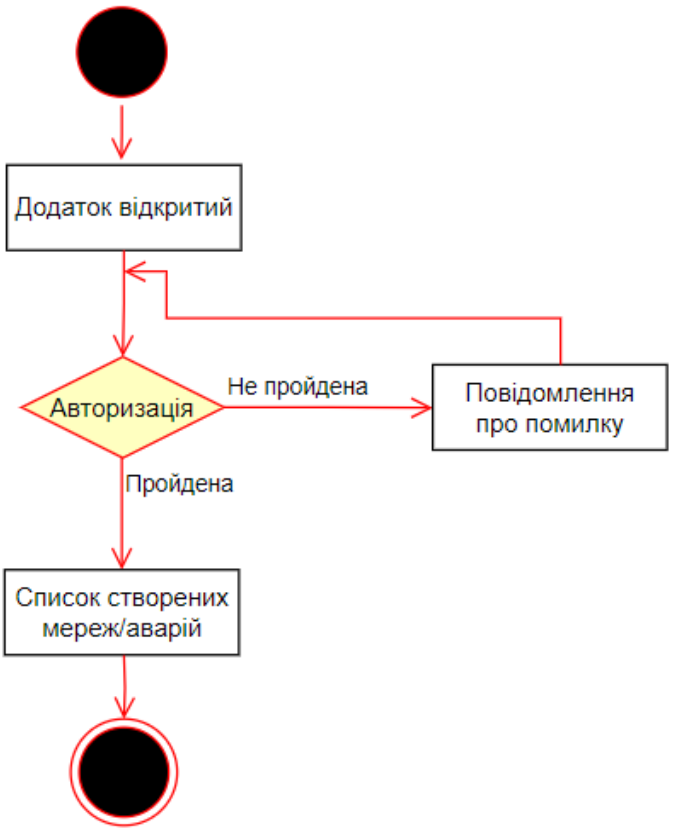


Рисунок 2.6 – Діаграма діяльності для огляду мереж/аварій

Для додавання нової мережі необхідно перейти на відповідне вікно та заповнити поля форми. На рисунку 2.7 представлена діаграма діяльності для додавання мережі.

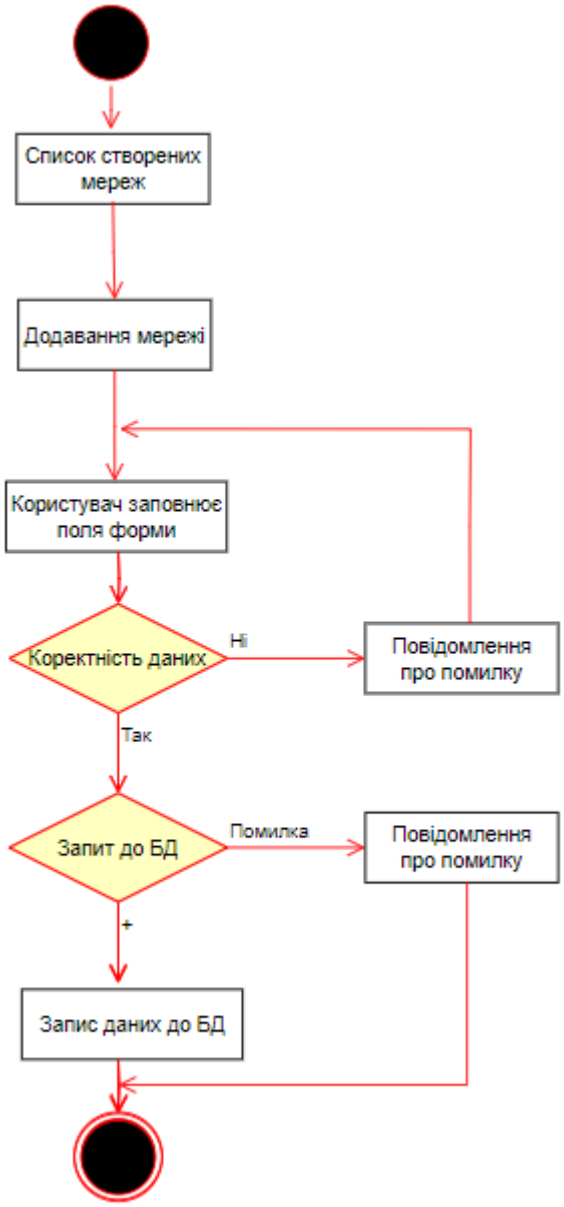


Рисунок 2.7 – Діаграма діяльності для створення мережі

Для додавання аварійної ситуації необхідно перейти на вікно для редагування мережі, перевести обладнання мережі в аварійний стан, внести дані про аварію.

На рисунку 2.8 представлена діаграма діяльності для додавання аварійної ситуації.

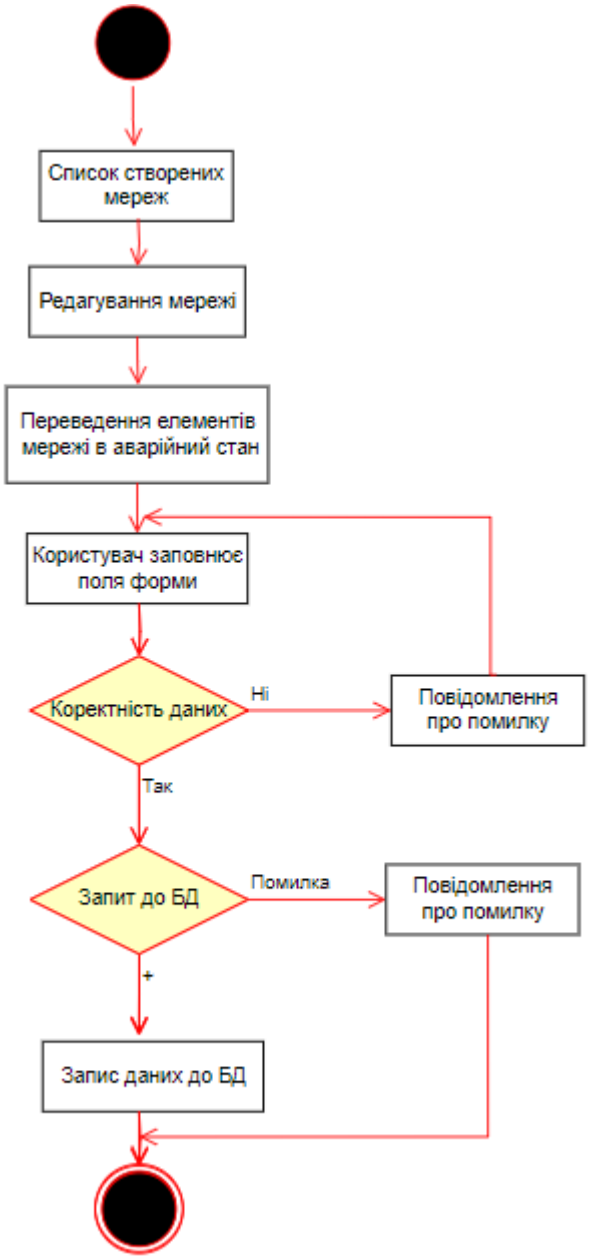


Рисунок 2.8 – Діаграма діяльності для додавання аварії

Для побудови альтернативного шляху у вікні редагування мережі обирається відповідна опція. За потреби дані в базі даних оновлюються. На рисунку 2.9 представлена діаграма діяльності для пошуку альтернативного шляху живлення підстанцій.

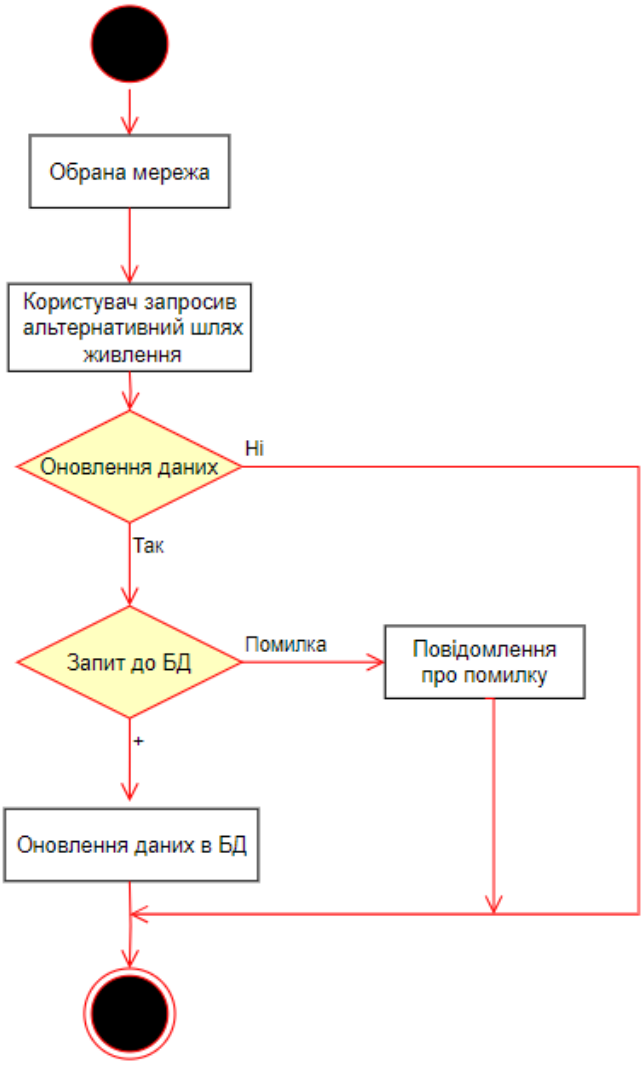


Рисунок 2.9 – Діаграма діяльності для пошуку альтернативного шляху

2.6 Проектування моделі бази даних

Для відображення логічних зв'язків між елементами даних незалежно від середовища зберігання та змісту даних використовується логічна модель даних. Вона описує, як дані організовані та пов'язані один з одним без прив'язки до конкретних технічних деталей реалізації, таких як фізичне сховище або програмне забезпечення. Логічна модель даних визначає сутності (entities), їх атрибути (attributes) та зв'язки (relationships) між сутностями. Вона допомагає зрозуміти структуру даних, основні сутності та їх характеристики, а також взаємозв'язки між ними.

В даному випадку логічна модель даних представлена у вигляді діаграми ER (Entity-Relationship [19]), яка представлена на рисунку 2.10:

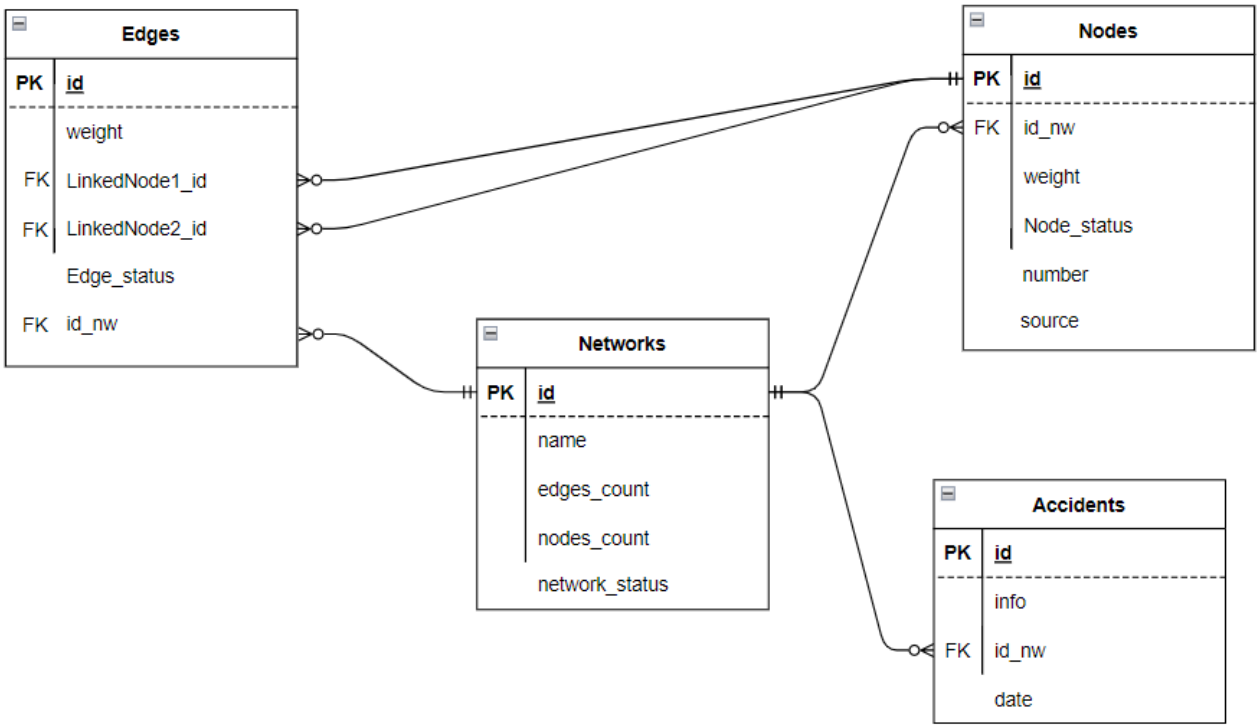


Рисунок 2.10 – Логічна модель даних

База даних включає 4 таблиці:

- Networks: з загальною інформацією про додані мережі;
- Accidents: з інформацією про додані аварійні ситуації;
- Nodes: з інформацією про електропідстанції;
- Edges: з інформацією про лінії електропередач.

Таблиця Networks містить наступні стовбці:

- id: унікальний ідентифікатор;
- name: назва мережі;
- edges_count: кількість ребер графа (з’єднань між підстанціями);
- nodes_count: кількість вузлів графа (підстанцій у мережі);
- network_status: стан мережі («активний» або «неактивний»).

Таблиця Accidents містить наступні стовбці:

- id: унікальний ідентифікатор;
- info: коротка інформація про аварію;
- id_nw: унікальний ідентифікатор мережі, в якій сталася аварія;
- date: дата аварії.

Таблиця Nodes містить наступні стовбці:

- id: унікальний ідентифікатор;
- id_nw: унікальний ідентифікатор мережі, до якої належить підстанція;
- weight: відстань кожного вузла до джерела живлення (вираховується програмно);
- Node_status: стан вузла («активний», «неактивний» або «аварійний»);
- number: номер (або назва) підстанції;
- source: мітка, яка показує чи є дана підстанція джерелом в електромережі.

Таблиця Edges містить наступні стовбці:

- id: унікальний ідентифікатор;
- id_nw: унікальний ідентифікатор мережі, до якої належить підстанція;
- weight: протяжність ліній електропередач;
- LinkedNode1_id та LinkedNode2_id: які підстанції з’єднані даною лінією електропередач;
- Edge_status: стан ребра («активний», «аварійний», «резерв» або «увімкнути»).

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Архітектура програмного додатку

Для візуалізації елементів і компонентів програми, що існують лише на етапі її виконання застосовується діаграма розгортання [20]. На ній зображено тільки компоненти-екземпляри програми, що є виконавчими файлами або динамічними бібліотеками.

На рисунку 3.1 представлена діаграма розгортання розроблюваної системи:

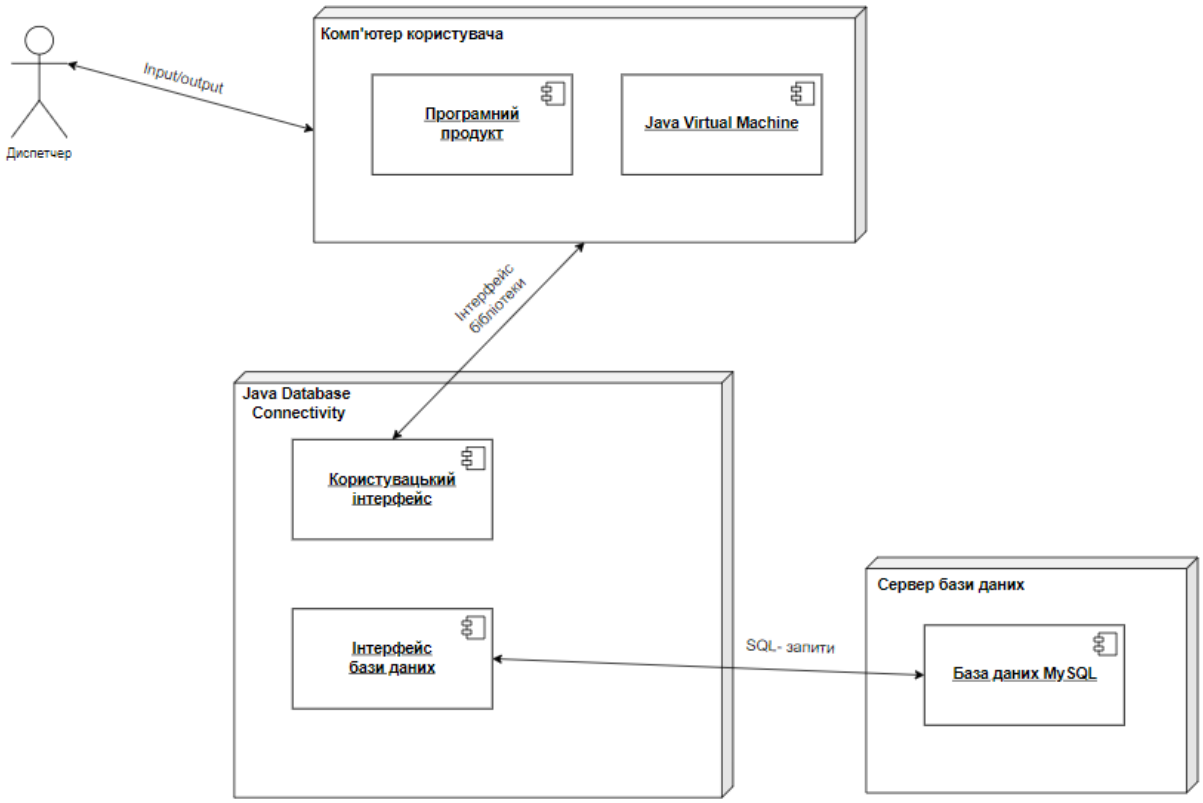


Рисунок 3.1 – Діаграма розгортання

Щоб отримати доступ до бази даних, використано JDBC (Java Database Connectivity [21]) API, який входить до стандартної бібліотеки Java. За наявності відповідної реалізації драйвера, необхідного для підключення, JDBC дозволяє

підключитися до будь-якої бази даних: Postgres, MySQL, SQL Server, Oracle тощо. Для роботи з даними таблиць застосовано SQL (Structured query language).

3.2 Використання програмного додатку

Після запуску додатку відкривається вікно для введення пароля для доступу до системи. Необхідно ввести пароль в поле для введення і натиснути «Вхід», рисунок 3.2:

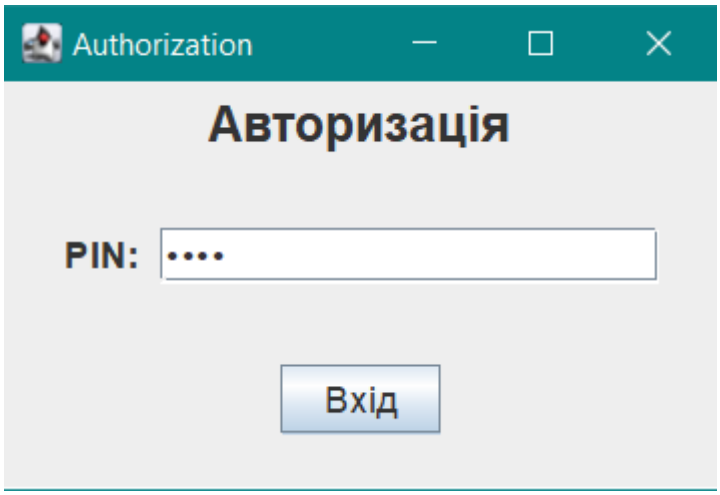


Рисунок 3.2 – Введення пароля

Далі відкривається вікно з доданими мережами (рисунок 3.3) та аварійними ситуаціями (рисунок 3.3 - 3.4).

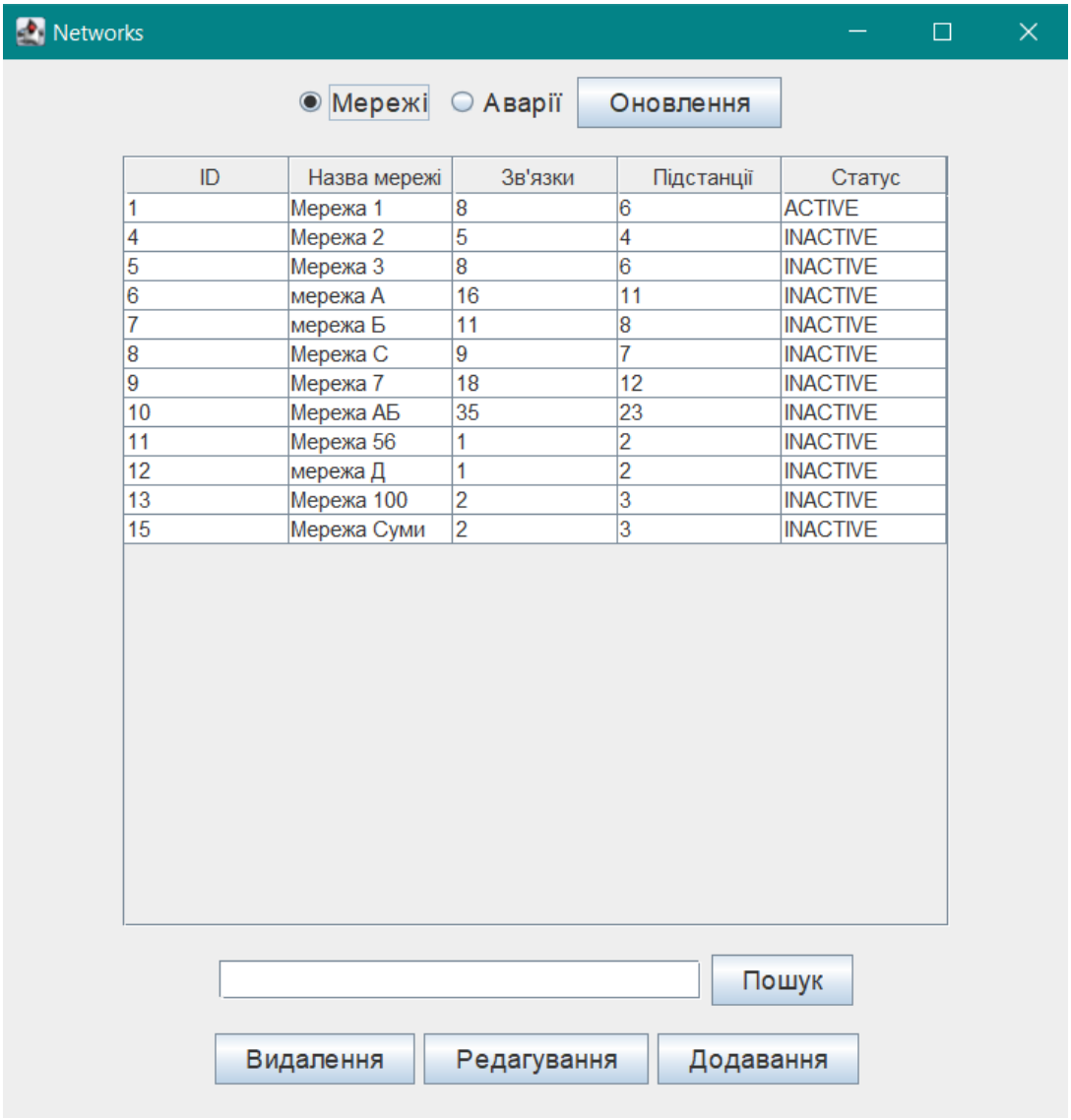


Рисунок 3.3 – Наявні мережі

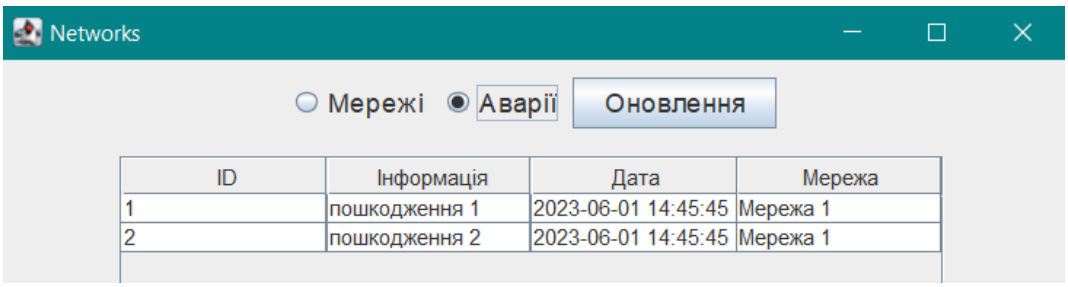


Рисунок 3.4 – Додані аварійні ситуації

Для пошуку необхідно ввести текст у відповідне поле і натиснути «Пошук». Результат зображений на рисунку 3.5:

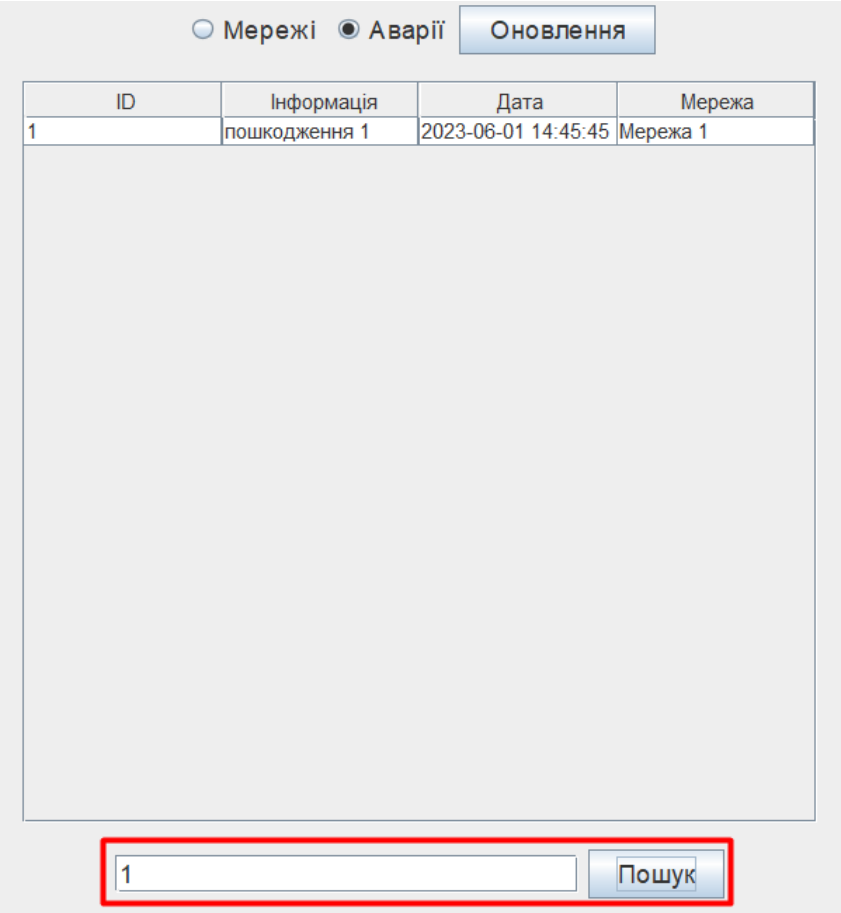


Рисунок 3.5 – Пошук у таблиці

Для додавання нової мережі необхідно натиснути на кнопку «Додавання» у вікні з мережами. Далі необхідно по черзі заповнити всі запропоновані поля. На рисунку 3.6 зображено процес додавання.

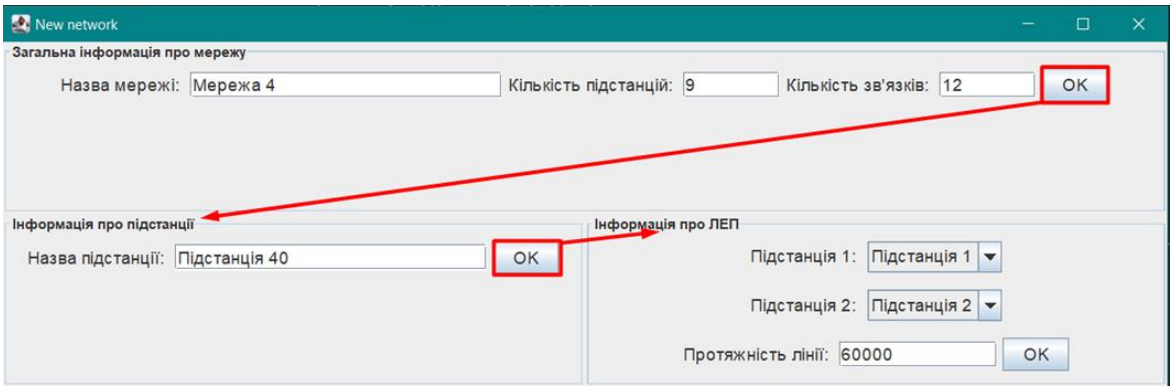


Рисунок 3.6 – Додавання мережі

Для видалення мережі необхідно вибрати мережу в таблиці та натиснути на кнопку «Видалення». Для редагування натискається кнопка «Редагування». Відкривається вікно (рисунок 3.7, рисунок 3.8) з інформацією про мережу.

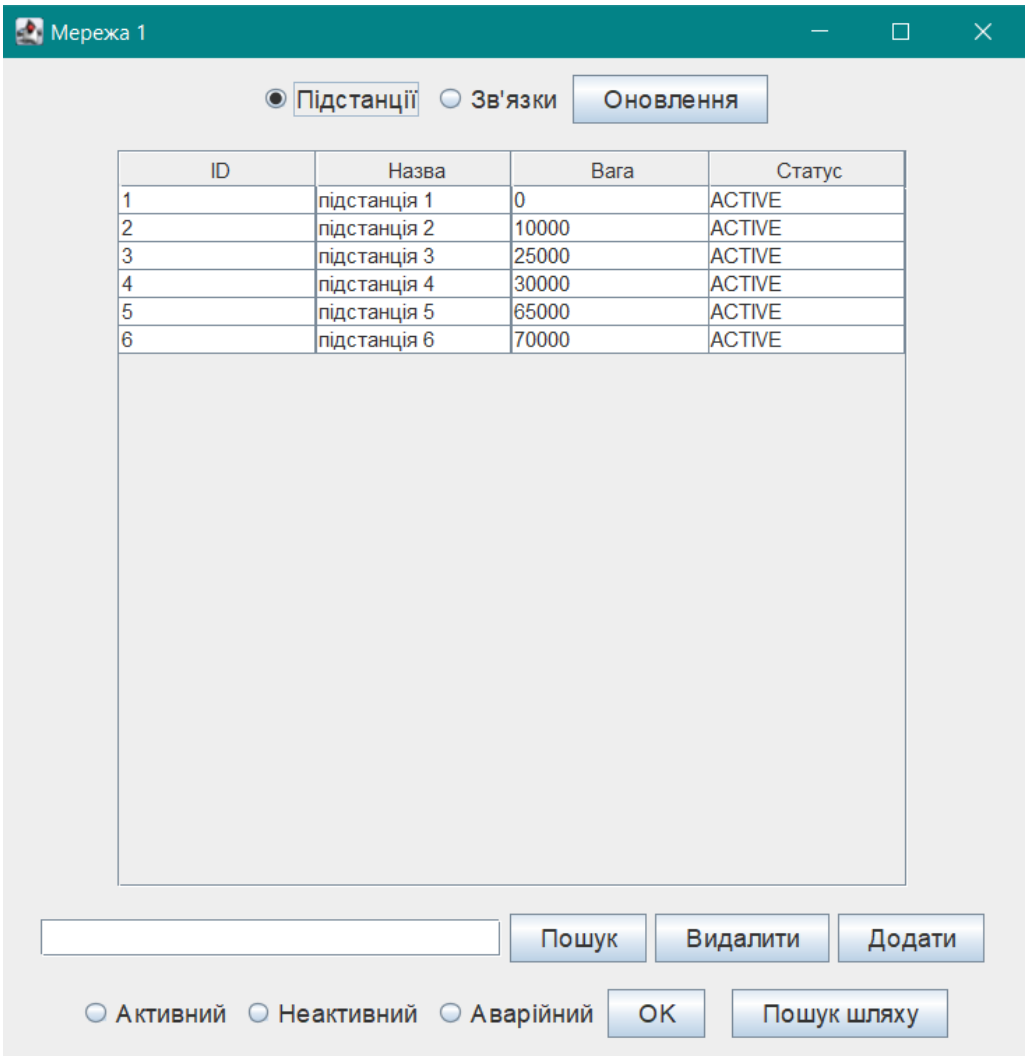


Рисунок 3.7 – Інформація про підстанції

Рисунок 3.8 – Інформація про ЛЕП

Для додавання підстанції необхідно натиснути кнопку «Додати» коли в таблиці відображаються підстанції. Відкриється вікно, в якому необхідно вписати назву у відповідне поле, приклад вікна зображено на рисунку 3.9:

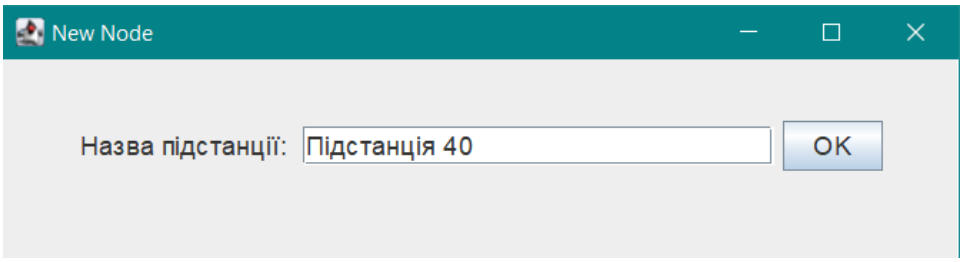


Рисунок 3.9 – Додавання підстанції

Далі слід додати лінії електропередач. Для цього необхідно натиснути кнопку «Додати» коли в таблиці відображаються зв’язки. Необхідно заповнити запропоновані поля у вікні, яке відкриється. Приклад вікна зображено на рисунку 3.10:

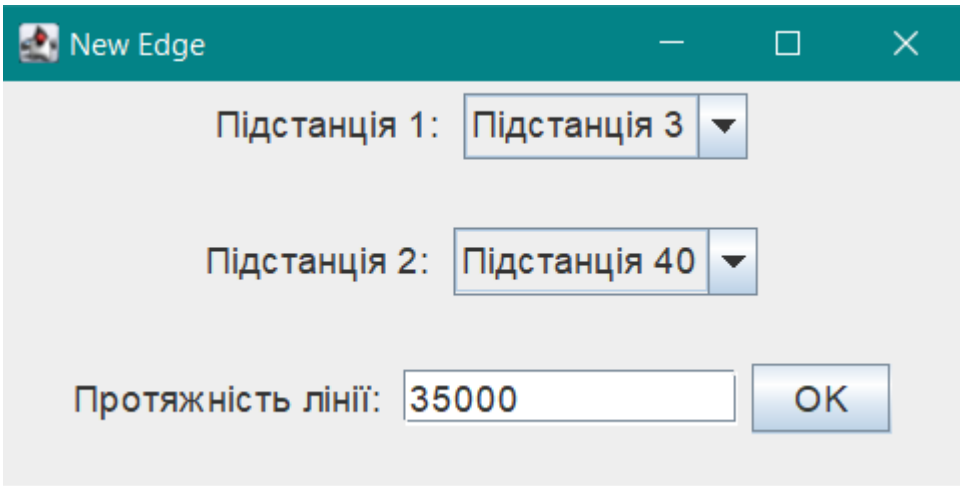


Рисунок 3.10 – Додавання ЛЕП

Для додавання аварійної ситуації необхідно обрати необхідний рядок таблиці та обрати стан «Аварійний» і натиснути кнопку «ОК» , як показано на рисунку 3.11:

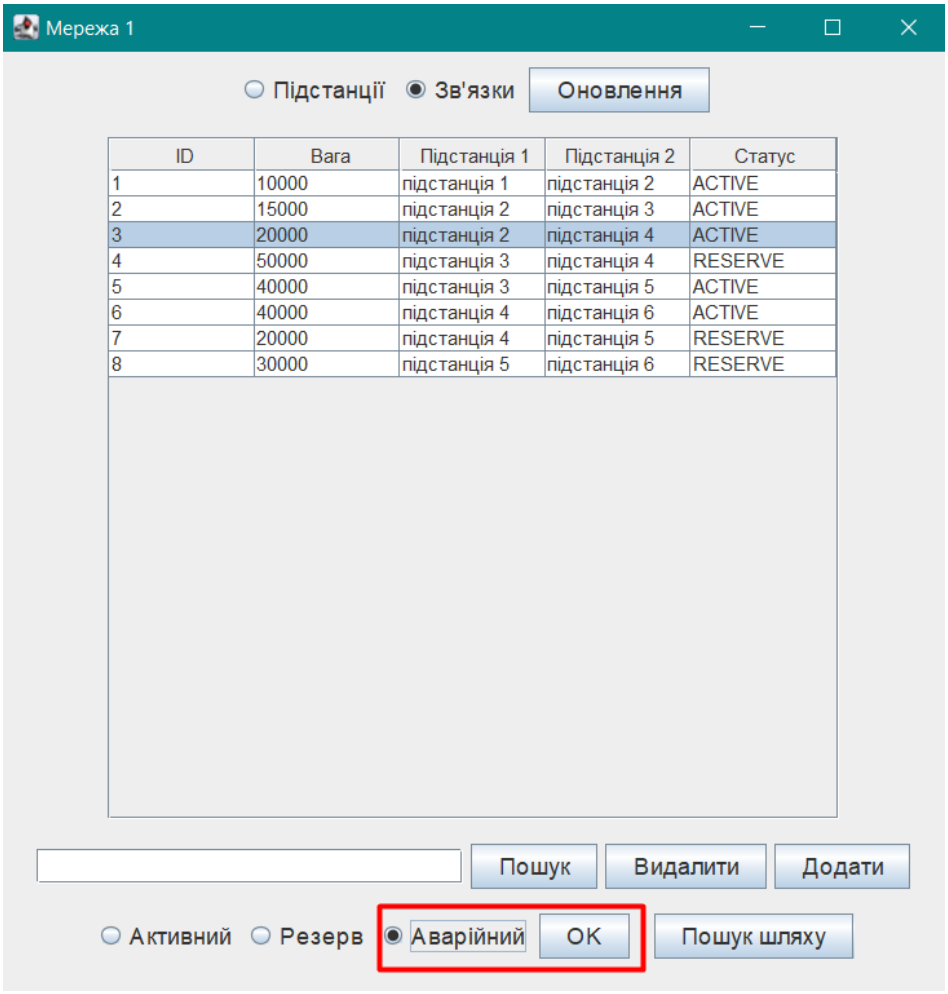


Рисунок 3.11 – Додавання аварійної ситуації

Відкриється вікно, куди необхідно написати короткий опис аварійної ситуації, як показано на рисунку 3.12:

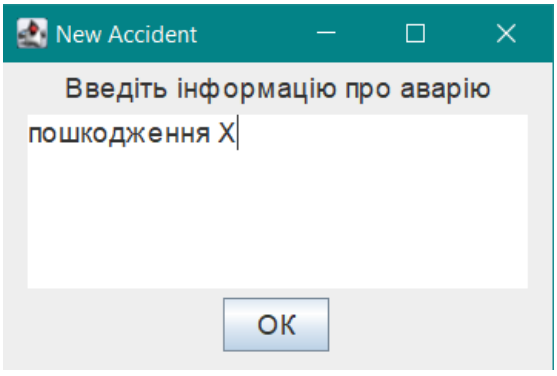


Рисунок 3.12 – Додавання аварійної ситуації

Для пошуку альтернативного шляху живлення слід натиснути на «Пошук шляху» (коли в таблиці відображаються зв’язки) як показано на рисунку 3.13:

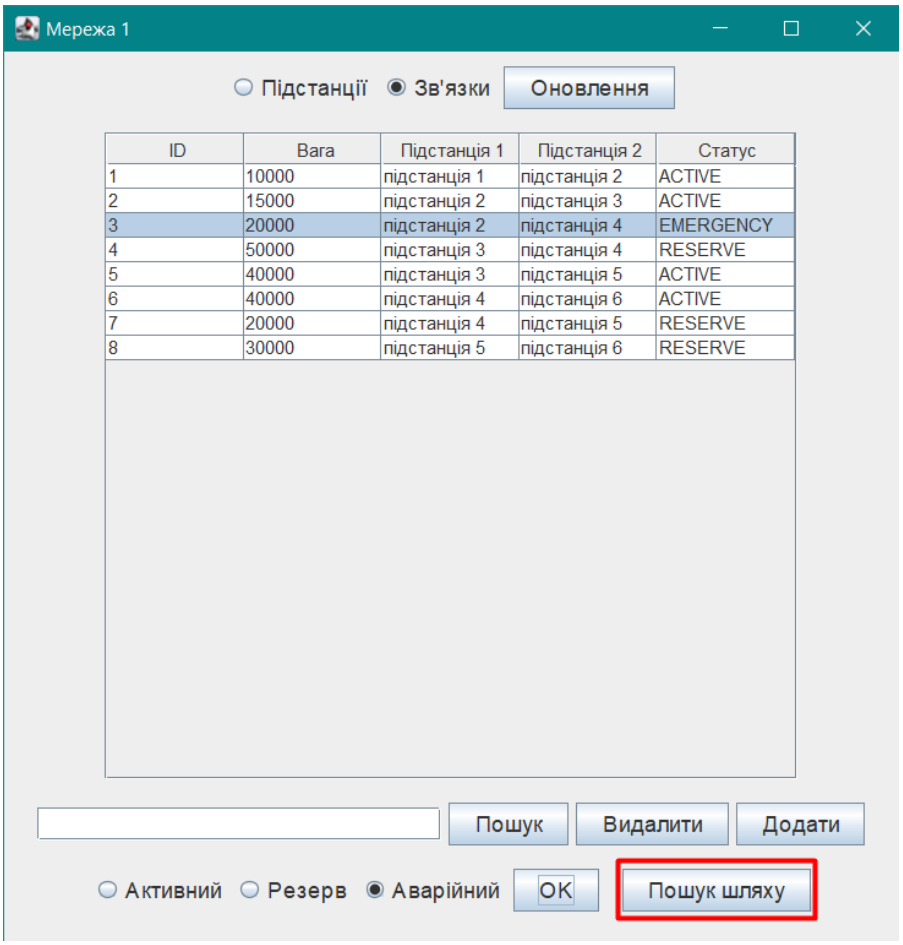


Рисунок 3.13 – Пошук резервного шляху

Далі необхідний рядок у таблиці ЛЕП переходить у режим «Увімкнути», як показано на рисунку 3.14:

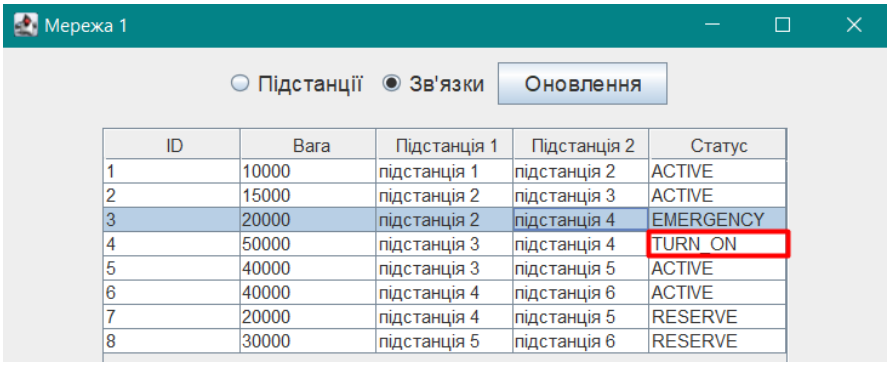


Рисунок 3.14 – Пошук резервного шляху

Для оновлення інформації в базі даних необхідно виділити рядки будь-якої таблиці та натиснути на кнопку «Оновлення» у верхній частині вікна, оновляться виділені рядки. Якщо рядки не виділені таблиця оновиться повністю.

ВИСНОВКИ

У результаті розробки проекту було реалізовано інформаційну систему підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях.

При цьому було вирішено наступні задачі:

- проаналізовано предметну область: виявлено існуючі проблеми, обґрунтовано актуальність розробки, проаналізовано існуючі методи та технології вирішення задачі, сформовано перелік вимог до системи;
- на основі результатів аналізу предметної області сформовано узагальнену постановку задачі а також мету та задачі розробки, на основі цього обрано та обґрунтовано засоби реалізації програмного продукту;
- на основі сформованої мети, задач і вимог до системи виконано моделювання та проектування системи – побудовано основні діаграми, що моделюють функціонування IT- продукту;
- на основі логічної моделі даних створено таблиці бази даних для довготривалого зберігання інформації, виконано тестування роботоспроможності бази даних за допомогою SQL запитів;
- створено зручний та зрозумілий графічний інтерфейс та реалізовано функціонал системи, який чітко відповідає поставленим вимогам.

Розроблена інформаційна система підтримки діяльності персоналу електричних мереж під час аварійних ситуацій допомагає зменшити кількість інформації, яку необхідно тримати в голові диспетчеру, а також виступає як повноцінний порадник або експерт.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global direct primary energy consumption. [Електронний ресурс] – Режим доступу до ресурсу: https://ourworldindata.org/grapher/global-primary-energy?time=earliest..latest
2. Як впоратися з тривогою та дискомфортом під час відключення світла: поради психоаналітика. [Електронний ресурс] – Режим доступу до ресурсу: https://apostrophe.ua/ua/news/society/2022-10-30/kak-spravitsya-s-trevogoy-i-diskomfortom-pri-otklyuchenii-sveta-sovetyi-psihoanalitika/282728
3. Професія диспетчер в енергокомпанії Житомиробленерго. [Електронний ресурс] – Режим доступу до ресурсу: https://www.ztoe.com.ua/news.php?id=1046
4. MONITORING SYSTEMS. [Електронний ресурс] – Режим доступу до ресурсу: https://www.digsilent.de/en/monitoring-systems.html?etcc_cmp=GCC_Allgemein_Suchnetzwerk&etcc_med=SEA&etcc_ori=Google&gclid=Cj0KCQjwj_ajBhCqARIsAA37s0xYAtjBGnbFyfLFU4JzEUFbsZYtgCz4www4IYU3XwQ-pKQIKoTNVJkaAt7vEALw_wcB
5. GRID INTEGRATION. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cleanenergyministerial.org/resources-cesc/grid-integration/
6. Що таке Smart Grid? [Електронний ресурс] – Режим доступу до ресурсу: https://nv.ua/ukr/ukraine/so-skorostyu-sveta/shcho-take-smart-grid-50055452.html
7. Energy and Cybersecurity Integration. [Електронний ресурс] – Режим доступу до ресурсу: https://www.energy.gov/femp/energy-and-cybersecurity-integration
8. Офіційний сайт Microsoft Excel. [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/excel?legRedir=true&CorrelationId=d05e72d6-a6d4-4514-ae32-1273fb93b2b8&rtc=1
9. Офіційний сайт OpenOffice Calc. [Електронний ресурс] – Режим доступу до ресурсу: http://www.openoffice.org/product/calc.html
10. NEPLAN | Electricity. [Електронний ресурс] – Режим доступу до ресурсу: https://www.neplan.ch/neplanproduct/en-electricity/

11. Network Manager Electric. [Електронний ресурс] – Режим доступу до ресурсу: https://www.iqgeo.com/product/electric-network-planning-design-and-workflow-management-software
12. What is Java technology and why do I need it? [Електронний ресурс] – Режим доступу до ресурсу: https://www.java.com/en/download/help/whatis_java.html
13. MySQL. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/
14. Dijkstra's Algorithm. [Електронний ресурс] – Режим доступу до ресурсу: https://www.programiz.com/dsa/dijkstra-algorithm
15. Нотація IDEF0. [Електронний ресурс] – Режим доступу до ресурсу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%202%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B2%D0%B0/page9.html
16. What is Use Case Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
17. ЗАСТОСУВАННЯ UML (ЧАСТИНА 2). ДІАГРАМА ПОСЛІДОВНОСТІ - SEQUENCE DIAGRAM. [Електронний ресурс] – Режим доступу до ресурсу: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy
18. What is Activity Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/
19. What is Entity Relationship Diagram (ERD)? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
20. What is Deployment Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/

21. java.sql (Java SE 11 & JDK 11). [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/package-summary.html

ДОДАТОК А. Технічне завдання

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

«Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях»

1. Призначення й мета інформаційної системи підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях

1.1 Призначення системи

Інформаційна система призначена для оптимізації роботи оператора системи управління електричною, підвищення стійкості та витривалості енергосистеми шляхом надання структурованої інформації для планування процесів на електричних мережах з накопиченої бази даних та найбільш економічно вигідних шляхів побудови альтернативного маршруту для живлення підстанцій у разі аварійної ситуації.

1.2 Мета створення системи

Головна мета проекту – розробка інформаційної системи підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях, яка допомагає зменшити кількість інформації, яку необхідно тримати в голові диспетчеру, а також виступає як повноцінний порадник або експерт.

1.3 Цільова аудиторія

Цільовою аудиторією даного проекту є Замовник, диспетчера електромереж а також всі інші хто так чи інакше залучений в роботу організації енергосистеми**.**

2. Вимоги до проекту

2.1 Вимоги до проекту в цілому

2.1.1 Вимоги до структури й функціонування

Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях повинна бути реалізована за допомогою сучасних технологій створення десктопних додатків та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути представлений десктопним додатком для комп’ютера, який містить якісне інформаційне наповнення та графічні матеріали, зручно структурований та взаємодіє з користувачем.

2.1.2 Вимоги до персоналу

Користувач не повинен мати особливих технічних навичок для роботи з додатком і його підтримкою. Єдиною вимогою є наявність навичок користування персональним комп’ютером та звичайними десктопними додатками.

2.1.3 Вимоги до збереження інформації

Інформація про підстанції, лінії електропередач та аварійні ситуації в мережах має зберігатися у базі даних, реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Розроблювана система призначена виключно для працівників галузі електроенергетики. Звичайні користувачі, власники персональних комп’ютерів, не мають необхідності у використанні системи.

Для диспетчера доступні всі функції, які надає система. Для доступу до інформаційної бази необхідно авторизуватися, ввести пароль у необхідному вікні додатку.

2.2 Структура інформаційної системи

2.2.1 Загальна інформація про структуру інформаційної системи

До структури додатку входять усі його вікна: вікно авторизації, вікно з інформацією про створені електричні мережі, вікно додавання мережі, вікно з інформацією про обрану мережу, вікно з інформацією про аварійні ситуації.

Розроблювана система складається з 7 вікон, їх призначення є наступним:

- вікно авторизації для введення логіна та пароля для доступу до системи;
- вікно з інформацією про створені електричні мережі та аварійні ситуації, яке містить таблицю з основною інформацією та меню для подальшої роботи з певною мережею;
- вікно додавання мережі, яке містить поля з даними про мережу, які необхідно заповнити;
- вікно з інформацією про обрану мережу, яке містить стан мережі та меню для роботи з нею;
- вікно додавання аварійної ситуації;
- вікно додавання нової підстанції в мережу;
- вікно додавання ЛЕП.

2.2.2 Навігаційне меню

Для зручної навігації кожне вікно містить кнопки керування (стандартні кнопки у правому верхньому куті), а також інші зручні механізми для переходу на нові вікна.

2.2.3 Управління контентом

Управління контентом (інформаційна база системи – це вся інформація, яка зберігається у базі даних) має здійснюватися за допомогою графічних вікон та елементів керування (таблиці, кнопки, поля для вводу тощо). Усе інформаційне наповнення десктопного додатку (інформація про електропідстанції та лінії електропередач а також аварійні ситуації) має міститися у базі даних.

2.2.4 Дизайн додатку

Дизайн додатку має бути виконаний у мінімалістичному та сучасному стилі. Під час розробки графічного інтерфейсу (вікон) додатку використовуються переважно відтінки сірого кольору.

Види і розміри тексту, кнопок, полів для введення повинні бути комфортними для перегляду. Інформаційні блоки, графічні матеріали та інші елементи вікон повинні мати зручне і логічне розташування. Шаблон майбутнього програмного продукту зображено на наступних рисунках.

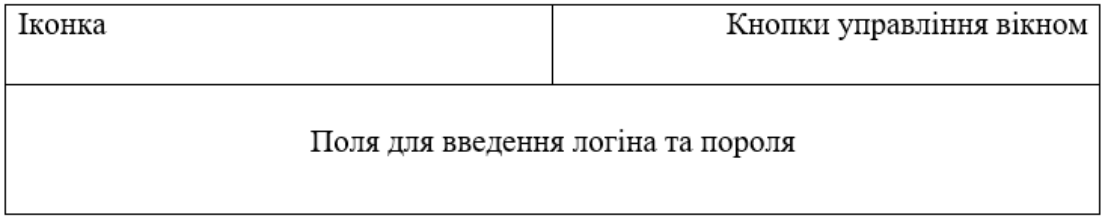


Рисунок А.1 – Схема вікна авторизації

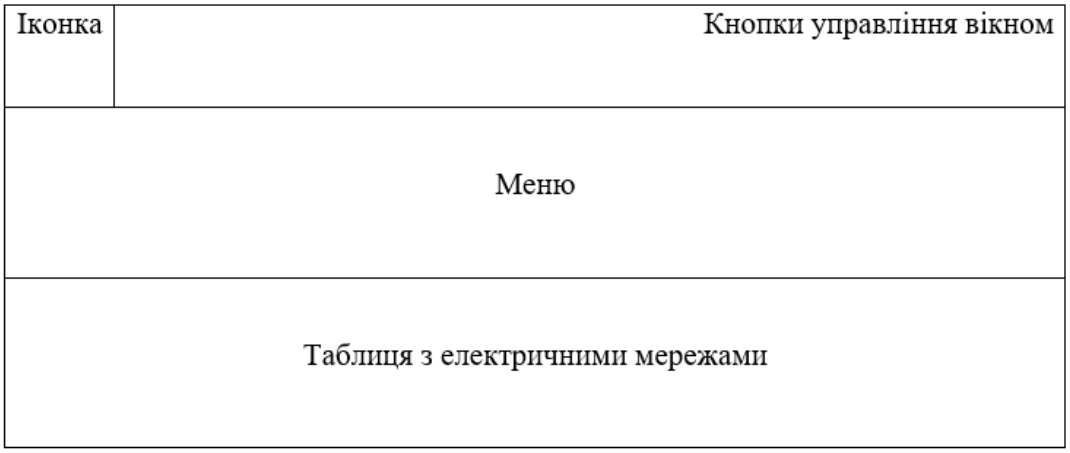


Рисунок А.2 – Схема вікна з інформацією про створені електричні мережі

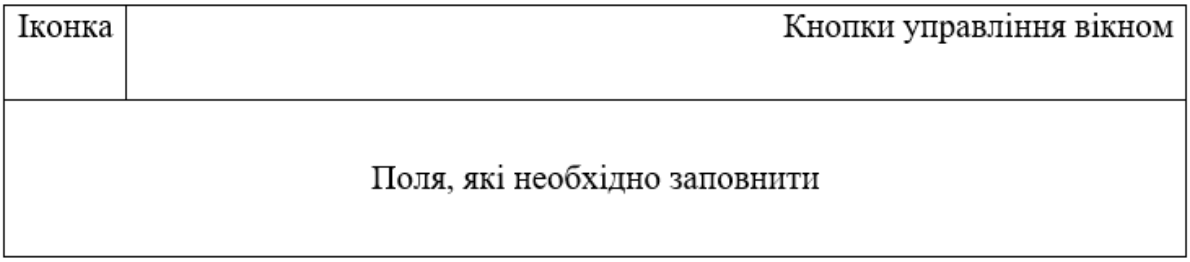


Рисунок А.3 – Схема вікна для додавання мережі, підстанцій або ЛЕП

Рисунок А.4 – Схема вікна з інформацією про мережу

Карта системи зображена на рисунку А.5.

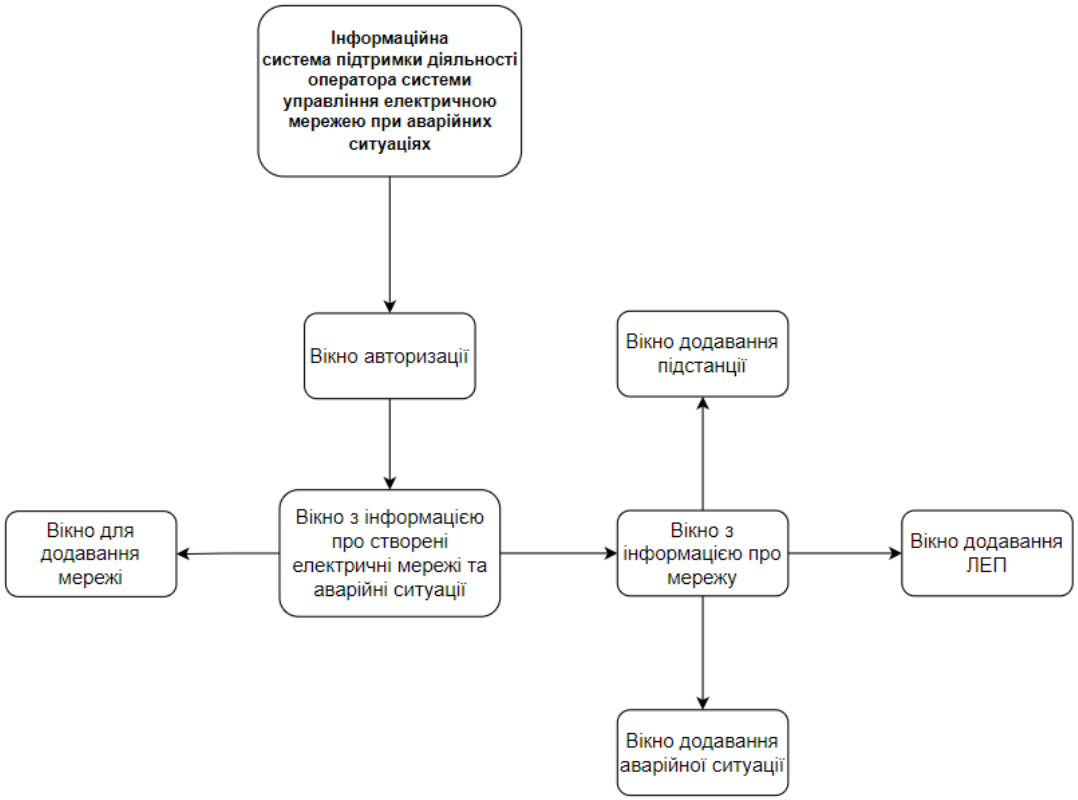


Рисунок А.5 – Схема навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Весь текст у додатку має бути виконаний українською або англійською мовами.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи додатку необхідна будь-яка операційна система, доступ в інтернет.

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Авторизація в системі	Диспетчер
UN-02	Перегляд створених мереж	Диспетчер
UN-03	Додавання нових мереж	Диспетчер
UN-04	Видалення/редагування мереж	Диспетчер
UN-05	Додавання аварійних ситуацій в мережі	Диспетчер
UN-06	Довготривале збереження даних в БД	Диспетчер
UN-07	Отримання пропозицій щодо побудови альтернативного маршруту живлення підстанції у разі аварії	Диспетчер

2.4.2 Системні вимоги

Проаналізувавши потреби замовника було визначено наступні вимоги:

- додавання, редагування, видалення електричних мереж;
- вказання аварійних ситуацій в доданих електричних мережах;
- надання пропозицій що до альтернативного шляху живлення електропідстанцій;
- зберігання інформації про електричні мережі в базі даних для подальшого аналізу та звітності.

3. Склад і зміст робіт зі створення інформаційної системи

Детальний опис етапів створення системи наведено в таблиці А.2.

Таблиця А.2 – Склад і зміст робіт

№	Склад і зміст робіт	Орієнтовний строк розробки
1	Аналіз предметної області	5 дні
2	Постановка мети та задач	3 дні
3	Вибір технологій реалізації	3 дні
4	Вибір алгоритму реалізації	3 днів
5	Створення бази даних	5 днів
6	Тестування бази даних	2 дні
7	Розробка графічного інтерфейсу користувача	3 днів
8	Написання коду програми	15 днів
9	Тестування роботоспроможності системи	5 днів
10	Створення Setup для завантаження додатку	1 день
11	Перевірка працездатності	1 день
12	Написання супровідної документації	5 днів
	Загальна тривалість робіт	51 день

ДОДАТОК Б. Планування робіт

Деталізація мети проекту методом SMART. Деталізацію мети проекту виконують за допомогою SMART-методу. Це дозволяє більш конкретно представити призначення розроблюваного продукту.

Для виконавця даного проекту формат постановки SMART-мети такий:

«Розробити інформаційну систему підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях, яка виступатиме як повноцінний порадник або експерт для диспетчера електромереж».

Результати деталізації мети даного проекту представлено в таблиці Б.1.

Таблиця Б.1 - Деталізація мети методом SMART

Specific	Підвищення стійкості та витривалості енергосистеми шляхом надання структурованої інформації для планування процесів на електричних мережах з накопиченої бази знань.
Measurable	Зменшення часу на ремонтні роботи в майбутньому та більш економічне використання ресурсів.
Achievable	Оператору надаються найбільш економічно вигідні шляхи побудови альтернативного маршруту, вся інформація зручно структурована.
Relevant	Саме використання комп'ютерних обчислювальних потужностей здатне істотно змінити підхід до виявлення аварій, прискорити темпи їх усунення
Time-framed	Ціль має часове обмеження. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Для планування змісту структури робіт побудовано WBS діаграму проекту – це розбиття проекту на конкретні результати, які мають бути досягнуті для реалізації цілей проекту. На верхньому рівні вказано сам

проект, під ним (на першому рівні) – основні задачі, кожна із яких, в свою чергу, деталізується. Діаграма WBS зображена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Після планування змісту структури робіт побудовано OBS діаграму - ієрархічну структуру управління проектом, якав показує відносини між учасниками проекту. Діаграма OBS зображена на рис. Б.2.

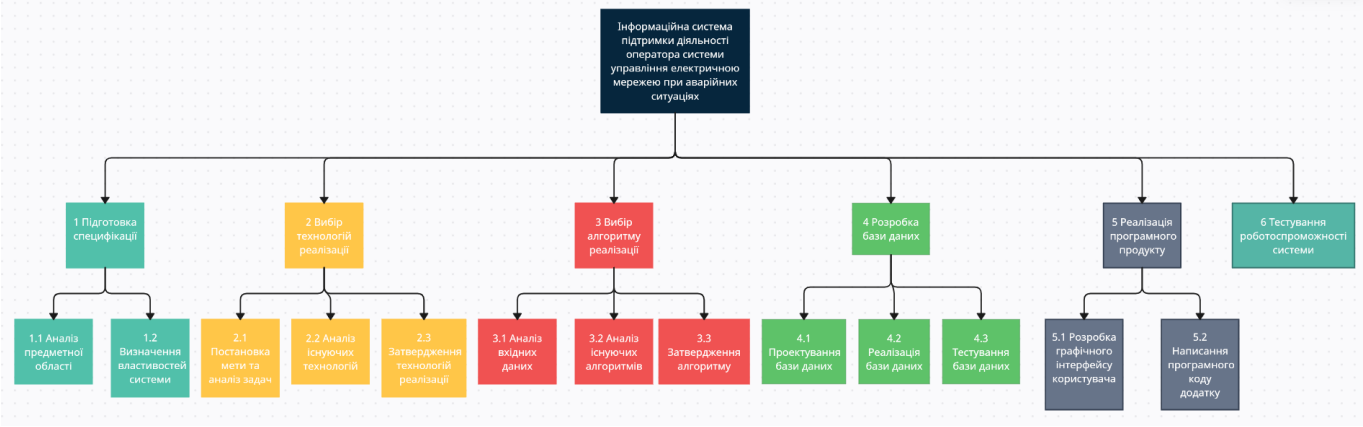


Рисунок Б.1 - WBS. Структура робіт проекту

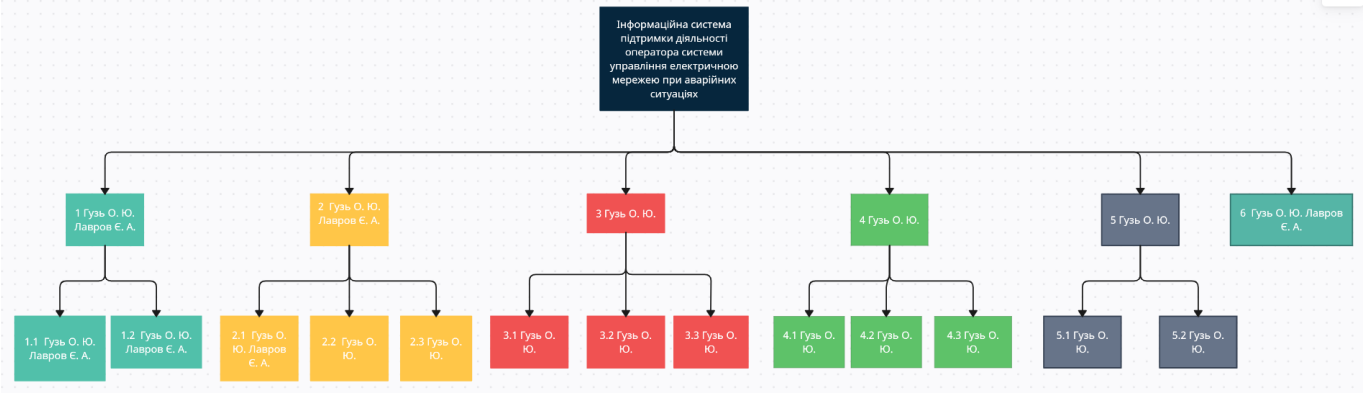
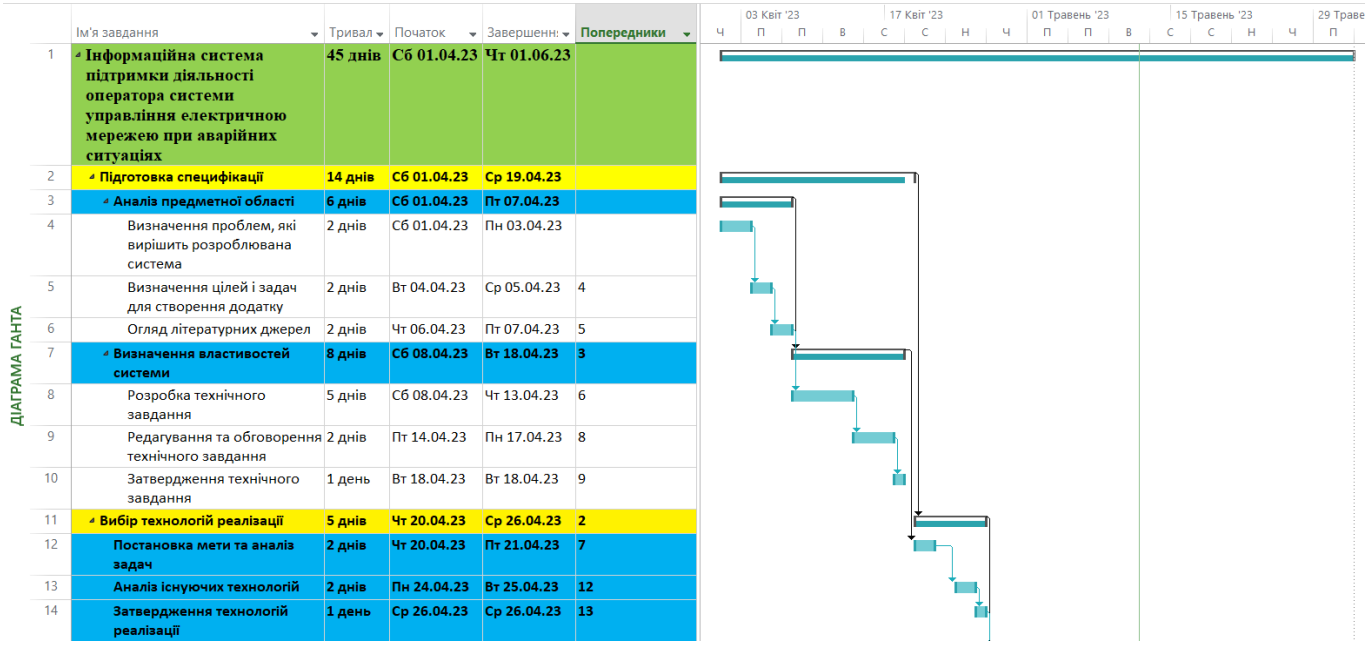


Рисунок Б.2 − Організаційна структура проекту (OBS)

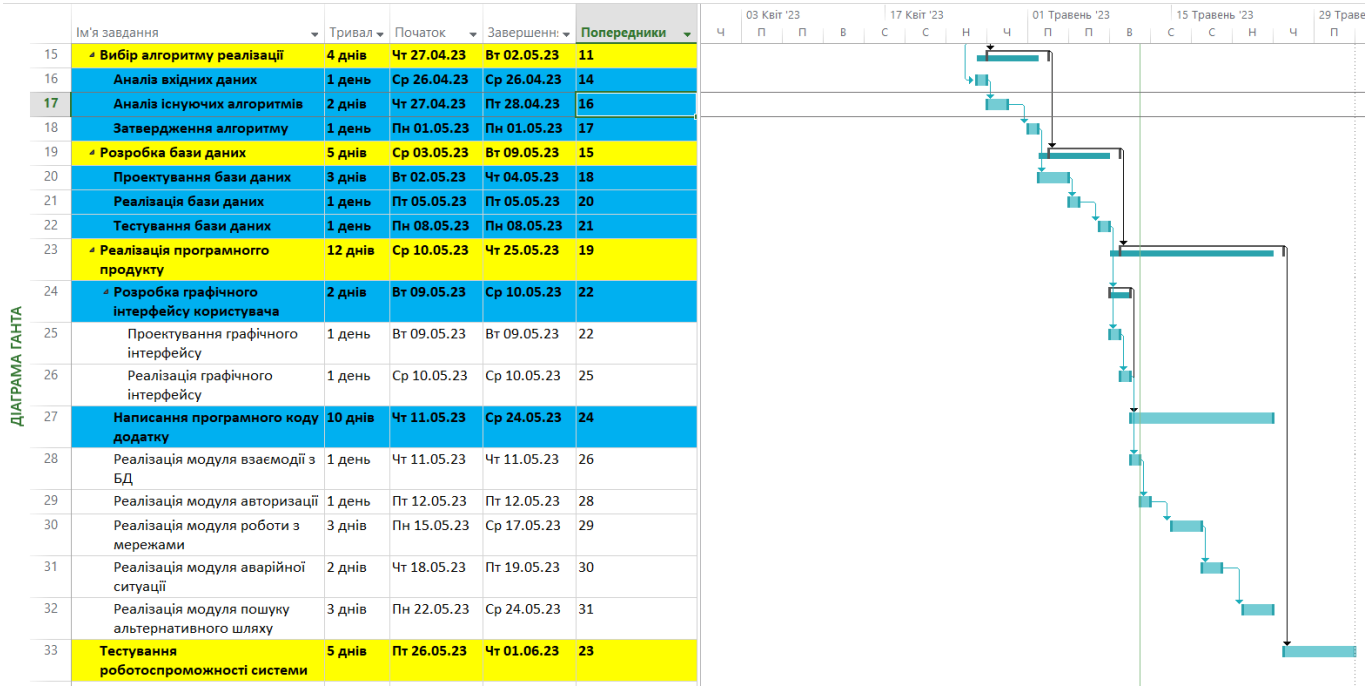
Діаграма Ганта. Для ілюстрації того, як виконується запланована робота з часом, побудовано діаграму Ганта. Вона складається з двох частин: у лівій частині наведено список завдань, а у правій — тимчасова шкала зі смугами, що зображують роботу. Також вказано дати початку та завершення завдань. Діаграма Ганта

зображена на рисунках Б.3-Б.4. Список робіт діаграми Ганта з вказанням залежності між завданнями та виконавцями зображений на рисунку Б.5.



	Ім'я завдання	Тривал	Початок	Завершенн	Попередники
1	**Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях**	**45 днів**	**Сб 01.04.23**	**Чт 01.06.23**	
2	**Підготовка специфікації**	**14 днів**	**Сб 01.04.23**	**Ср 19.04.23**	
3	**Аналіз предметної області**	**6 днів**	**Сб 01.04.23**	**Пт 07.04.23**	
4	Визначення проблем, які вирішить розроблювана система	2 днів	Сб 01.04.23	Пн 03.04.23	
5	Визначення цілей і задач для створення додатку	2 днів	Вт 04.04.23	Ср 05.04.23	4
6	Огляд літературних джерел	2 днів	Чт 06.04.23	Пт 07.04.23	5
7	**Визначення властивостей системи**	**8 днів**	**Сб 08.04.23**	**Вт 18.04.23**	**3**
8	Розробка технічного завдання	5 днів	Сб 08.04.23	Чт 13.04.23	6
9	Редагування та обговорення технічного завдання	2 днів	Пт 14.04.23	Пн 17.04.23	8
10	Затвердження технічного завдання	1 день	Вт 18.04.23	Вт 18.04.23	9
11	**Вибір технологій реалізації**	**5 днів**	**Чт 20.04.23**	**Ср 26.04.23**	**2**
12	**Постановка мети та аналіз задач**	**2 днів**	**Чт 20.04.23**	**Пт 21.04.23**	**7**
13	**Аналіз існуючих технологій**	**2 днів**	**Пн 24.04.23**	**Вт 25.04.23**	**12**
14	**Затвердження технологій реалізації**	**1 день**	**Ср 26.04.23**	**Ср 26.04.23**	**13**

Рисунок Б.3 – Діаграма Ганта



	Ім'я завдання	Тривал	Початок	Завершенн	Попередники
15	**Вибір алгоритму реалізації**	**4 днів**	**Чт 27.04.23**	**Вт 02.05.23**	**11**
16	**Аналіз вхідних даних**	**1 день**	**Ср 26.04.23**	**Ср 26.04.23**	**14**
17	**Аналіз існуючих алгоритмів**	**2 днів**	**Чт 27.04.23**	**Пт 28.04.23**	**16**
18	**Затвердження алгоритму**	**1 день**	**Пн 01.05.23**	**Пн 01.05.23**	**17**
19	**Розробка бази даних**	**5 днів**	**Ср 03.05.23**	**Вт 09.05.23**	**15**
20	**Проектування бази даних**	**3 днів**	**Вт 02.05.23**	**Чт 04.05.23**	**18**
21	**Реалізація бази даних**	**1 день**	**Пт 05.05.23**	**Пт 05.05.23**	**20**
22	**Тестування бази даних**	**1 день**	**Пн 08.05.23**	**Пн 08.05.23**	**21**
23	**Реалізація програмного продукту**	**12 днів**	**Ср 10.05.23**	**Чт 25.05.23**	**19**
24	**Розробка графічного інтерфейсу користувача**	**2 днів**	**Вт 09.05.23**	**Ср 10.05.23**	**22**
25	Проектування графічного інтерфейсу	1 день	Вт 09.05.23	Вт 09.05.23	22
26	Реалізація графічного інтерфейсу	1 день	Ср 10.05.23	Ср 10.05.23	25
27	**Написання програмного коду додатку**	**10 днів**	**Чт 11.05.23**	**Ср 24.05.23**	**24**
28	Реалізація модуля взаємодії з БД	1 день	Чт 11.05.23	Чт 11.05.23	26
29	Реалізація модуля авторизації	1 день	Пт 12.05.23	Пт 12.05.23	28
30	Реалізація модуля роботи з мережами	3 днів	Пн 15.05.23	Ср 17.05.23	29
31	Реалізація модуля аварійної ситуації	2 днів	Чт 18.05.23	Пт 19.05.23	30
32	Реалізація модуля пошуку альтернативного шляху	3 днів	Пн 22.05.23	Ср 24.05.23	31
33	**Тестування роботоспроможності системи**	**5 днів**	**Пт 26.05.23**	**Чт 01.06.23**	**23**

Рисунок Б.4 – Продовження діаграми Ганта

	A	B	C
1	Назва	Виконавець	Керівник
2	**Інформаційна система підтримки діяльності оператора системи управління електричною мережею при аварійних ситуаціях**	Гузь О. Ю.	Лавров Є. А.
3	**Підготовка специфікації**	+	+
4	**Аналіз предметної області**	+	+
5	Визначення проблем, які вирішить розроблювана система	+	+
6	Визначення цілей і задач для створення додатку	+	+
7	Аналіз існуючих методів	+	
8	**Визначення властивостей системи**	+	+
9	Розробка технічного завдання	+	
10	Редагування та обговорення технічного завдання	+	+
11	Затвердження технічного завдання	+	+
12	**Вибір технологій реалізації**	+	+
13	**Постановка мети та аналіз задач**	+	+
14	**Аналіз існуючих технологій**	+	
15	**Затвердження технологій реалізації**	+	
16	**Вибір алгоритму реалізації**	+	
17	**Аналіз вхідних даних**	+	
18	**Аналіз існуючих алгоритмів**	+	
19	**Затвердження алгоритму**	+	
20	**Розробка бази даних**	+	
21	**Проектування бази даних**	+	
22	**Реалізація бази даних**	+	
23	**Тестування бази даних**	+	
24	**Реалізація програмного продукту**	+	
25	**Розробка графічного інтерфейсу користувача**	+	
26	Проектування графічного інтерфейсу	+	
27	Реалізація графічного інтерфейсу	+	
28	**Написання програмного коду додатку**	+	
29	Реалізація модуля взаємодії з БД	+	
30	Реалізація модуля авторизації	+	
31	Реалізація модуля роботи з мережами	+	
32	Реалізація модуля аварійної ситуації	+	
33	Реалізація модуля пошуку альтернативного шляху	+	
34	**Тестування роботоспроможності системи**	+	+

Рисунок Б.5 – Список робіт для побудови діаграми Ганта

Аналіз ризиків. Під час виконання якісної оцінки ризиків визнаються ризики, які мають бути усунені якнайшвидше. Реагування буде залежати від ступеня важливості ризику. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконуватися одночасно або окремо. Це залежить від ступеня забезпечення проекту.

У таблиці Б.2 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця Б.2 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу.

Оцінка	**Ймовірність виниклення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

На основі мети проекту, вимог до розроблюваного продукту, створеного календарного графіку проекту зі вказаними задачами та часом виконання визначено можливі ризики, які можуть виникнути в процесі реалізації проекту. Перелік ризиків наведено в таблиці Б.3.

Таблиця Б.3 – Ідентифікація ризиків проекту

№ ризику	**Назва ризику**
1	Аварійні відключення електроенергії
2	Неоптимальний розподіл часу
3	Неправильне визначення термінів
4	Непорозуміння між розробником та замовником
5	Низька кваліфікація розробників
6	Недотримання сучасних технологій

Продовження таблиці Б.3

№ ризику	Назва ризику
7	Нечітке завдання на розробку
8	Часте внесення змін у ТЗ

На основі таблиці Б.3 дано оцінку кожному ризику щодо його ймовірності виникнення, впливу на хід власного проекту та його ранг. Результат представлений в таблиці Б.4.

Таблиця Б.4 – Визначення ймовірності, впливу та рангу ризиків проекту

№ ризику	Назва ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Аварійні відключення електроенергії	**0.5**	**0.8**	**0.4**
2	Неоптимальний розподіл часу	**0.5**	**0.2**	**0.1**
3	Неправильне визначення термінів	**0.5**	**0.4**	**0.2**
4	Непорозуміння між розробником та замовником	**0.1**	**0.2**	**0.02**
5	Низька кваліфікація розробників	**0.5**	**0.4**	**0.2**
6	Недотримання сучасних технологій	**0.3**	**0.4**	**0.12**
7	Нечітке завдання на розробку	**0.1**	**0.4**	**0.04**
8	Часте внесення змін у ТЗ	**0.1**	**0.4**	**0.04**

Для зменшення негативного впливу ризиків на проект виконується планування реагування на них. Воно включає визначення ефективності розробки та оцінка наслідків впливу ризику на проект. Оцінювання виконується за показниками, що описані в таблиці Б.2. У результаті планування реагування було отримано матрицю (таблиця Б.5) ймовірності виникнення ризиків та впливу ризику на проект. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі. Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.6.

Таблиця Б.5 – Матриця ймовірності та впливу згідно проекту

Ймовірність ризику (Й)	**Вплив загрози (ризику)**				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0.05	0.1	0.2	0.4	0.8
0.9					
0.7					
0.5			R2 (0.1)	R3 (0.2) R5 (0.2)	R1 (0.4)
0.3				R6 (0.12)	
0.1			R4 (0.02)	R7 (0.04) R8 (0.04)	

Таблиця Б.6 – Шкала оцінювання за рівнем ризику

№	**Назва**	**Межі**	**Ризикі, які входять**
1	Прийнятні	0,005≤R≤0,05	4, 7, 8
2	Виправдані	0,05<R≤0,14	2, 6
3	Недопустимі	0,14<R≤0,72	1, 3, 5

На основі попередньо представлених даних створено заходи реагування на виявлені ризики. У таблиці Б.7 описано ризики та стратегії реагування на кожен з них.

Таблиця Б.7 – Ризики проекту та стратегії реагування

ID	Статус	Опис	Ймовірність	Вплив	Ранг	План А	Тип стратегії	План Б
R1	Відкритий	Аварійні відключення електроенергії	Висока	Високий	0.4	1) Регулярно зберігати проект 2) Тримати копію проекту наприклад на Гугл Диску	Прийняття	
R2	Відкритий	Неоптимальний розподіл часу	Середня	Середній	0.1	1) Ретельно проаналізувати всі види робіт, використати діаграму Ганта, мережевий графік 2) Скласти таблицю з роботами та витратами часу, слідувати їй	Зменшення	При виявленні неоптимального розподілу часу перевірити відповідність виконаних робіт плану, в разі чого скористатись резервним часом
R3	Відкритий	Неправильне визначення термінів	Середня	Високий	0.2	1) Чітко визначити з замовником кінцеві сроки виконання	Ухилення	

Продовження таблиці Б.7

ID	Статус	Опис	Ймовірність	Вплив	Ранг	План А	Тип стратегії	План Б
R4	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	0.02	1) Налагодити гарні відносини між розробником та керівником. 2) Дотримуватися ділового етикету спілкування. 3) Створити комфортні умови для співпраці	Зменшення	При виявленні непорозуміння потрібно з'ясувати, що саме стало причиною непорозуміння обсудити її та створити здорову атмосферу в колективі.
R5	Відкритий	Низька кваліфікація розробників	Середня	Високий	0.2	1) При виявлення проблем під час розробки звертатись до висококваліфікованих спеціалістів	Прийняття	
R6	Відкритий	Недотримання сучасних технологій	Низька	Високий	0.12	1) Використання лише найновіший версій механізмів розробки 2) Слідування лише найбільш актуальним рекомендаціям до розробки веб-сайтів	Ухилення	

Продовження таблиці Б.7

ID	Статус	Опис	Ймовірність	Вплив	Ранг	План А	Тип стратегії	План Б
R7	Відкритий	Нечітке завдання на розробку	Низька	Високий	0.04	1) Ясно і однозначно обговорити із замовником усі види вимог. 2) Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3) Періодичний контроль замовником етапів роботи.	Зменшення	При виявленні невідповідностей деяких характеристик продукту заявленим вимогам потрібно уважно та чітко окреслити те, що було виконано невірно та внести правки.
R8	Відкритий	Часте внесення змін у ТЗ	Низька	Високий	0.04	1) Ретельно проаналізувати з замовником розроблене ТЗ, внести правки 2) Виключити варіанти внесення змін до ТЗ	Ухилення	

ДОДАТОК В. Копії публікацій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
МІЖНАРОДНИЙ УНІВЕРСИТЕТ «АСТАНА»

ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА

IMA :: 2023

МАТЕРІАЛИ
та програма

МІЖНАРОДНОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ
молодих учених

(Суми – Астана, 24–28 квітня 2023 року)

Суми
Сумський державний університет
2023

Інформаційна система діагностування аварійних ситуацій електричних мереж

Гузь О.Ю., *студент ІТ-91*; Лавров Є.А., *професор*
Сумський державний університет, м. Суми, Україна

Вступ. Сучасні підприємства вкрай залежні від електрики настільки, що тривале відключення електроенергії може викликати надзвичайні ситуації. Через те, що диспетчеру доводиться обробляти величезну кількість інформації, на етапі вибору альтернативного маршруту з'являються труднощі з розташуванням вузлів, постає питання про економічні витрати, необхідно враховувати людські ресурси, навантаження на підстанції, пріоритетність підстанцій, переконатися у відсутності подвійного включення, скласти заявку для групи реагування і так далі..

Постановка задачі. Розробити інформаційну систему підтримки роботи диспетчера електропідстанцій, підвищення стійкості та витривалості енергосистеми шляхом надання структурованої інформації для планування процесів на електричних мережах з накопиченої бази даних та найбільш економічно вигідних шляхів побудови альтернативного маршруту для живлення підстанцій у разі аварійної ситуації.

Результати. Для реалізації системи використовуємо методологію, розроблену Москвіним Н. В. та Суходоєєвим М. С.[1]. Функціонально-структурний аналіз полягає у формуванні ланцюжка міркувань, що призводить до однозначних висновків про стан мережі[1]. Ці міркування подаються у вигляді дерева, слідуючи по гілках якого можна дійти тієї чи іншої відповіді на питання про стан мережі[1].Можна виділити [1] чотири смислових і три функціональні варіанти комутаційної мережі, для представлення його як об'єкта з рівневою структурою. Сенси: 1. Передача потоку за "нормально включеною схемою"; 2. Передача потоку за вимушеною схемою (відхилення від нормальної); 3. Не надсилання потоку до навантажень; 4. Погашення частини мережі (зняття напруги).

Функції: 1. Здійснення передачі потоку енергії всім споживачів з необхідним рівнем надійності 2. Передача потоку всім споживачів здійснюється за використання резервних маршрутів 3. Відсутність потоку енергії (виведення з роботи елементів без резервування).

Результатом роботи системи є процес порівняння (нормальної та передбачуваної сполуки), агрегування (схеми мережі), класифікація (аварія або резервний шлях), складання рапорту, який відбиває повний стан мережі та схему її комутації. В системі передбачені (рис 1): вікно авторизації для введення логіна та пароля для доступу до системи; вікно з інформацією про створені електричні мережі, яке містить таблицю електричних мереж та меню для подальшої роботи з певною мережею; вікно додавання мережі, яке містить поля з даними про мережу, які необхідно заповнити; вікно з інформацією про обрану мережу, яке містить стан мережі та меню для роботи з нею; вікно з інформацією про аварійні ситуації, яке містить дані про всі додані аварійні ситуації: мережа, характер аварії, дата.

Рисунок 1 – Схема навігації

Висновки. Використання системи звільнить диспетчера від рутинної роботи та зменшить кількість допущених помилок.

1 Moskvin N.V. Development of an expert system for diagnosing emergency situations in an electrical network: master's thesis 2020

ДОДАТОК Г. Лістинг програми

Файл Main.java

```
package com.company;

import com.company.GUI.*;

public class Main {

    public static void main(String[] args) {

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            AuthorizationGUI authorizationGUI = new AuthorizationGUI();
        }
        catch (Exception ex){
            System.out.println(ex);
        }
    }
}
```

Файл AuthorizationGUI.java

```
package com.company.GUI;

import com.company.functional.Authorization;

import javax.swing.*;
import java.awt.*;

public class AuthorizationGUI {
    private final JFrame jf;

    public AuthorizationGUI(){
        jf = new JFrame("Authorization");
        jf.setSize(300, 200);
        jf.setLayout(new GridLayout(3, 1));
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel name = new JPanel();
        JLabel jLabel = new JLabel("Авторизація");
        jLabel.setFont(new Font("Arial", Font.BOLD, 20));
        name.add(jLabel);

        JPanel pass = new JPanel();
        JLabel pin = new JLabel("PIN: ");
```

```
        pin.setFont(new Font("Arial", Font.BOLD, 15));
        JPasswordField password = new JPasswordField(15);
        password.setFont(new Font("Arial", Font.PLAIN, 15));
        pass.add(pin);
        pass.add(password);

        JPanel enter = new JPanel();
        JButton enter_button = new JButton("Вхід");
        enter_button.addActionListener(new Authorization(jf, password));
        enter_button.setFont(new Font("Arial", Font.PLAIN, 15));
        enter.add(enter_button);

        jf.add(name);
        jf.add(pass);
        jf.add(enter);
        jf.setVisible(true);
    }
}
```

Файл NetworksGUI.java

```
package com.company.GUI;

import com.company.functional.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

public class NetworksGUI {
    private final JFrame jf;

    public NetworksGUI() {
        jf = new JFrame("Networks");
        jf.setSize(600, 620);
        jf.setLayout(new FlowLayout());
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        DefaultTableModel network_model = new DefaultTableModel();
        DefaultTableModel accident_model = new DefaultTableModel();
        JTable network_table = new JTable(network_model);
        JTable accident_table = new JTable(accident_model);
        JPanel network_table_panel = new JPanel();
        JPanel accident_table_panel = new JPanel();

        JPanel top = new JPanel();
        JRadioButton networks = new JRadioButton("Мережі");
        networks.addItemListener(new NetworksAccidentsDisplay("networks", network_model,
network_table_panel, accident_table_panel));
        networks.setFont(new Font("Arial", Font.PLAIN, 15));
```

```
        JRadioButton emergency = new JRadioButton("Аварії");
        emergency.addItemListener(new NetworksAccidentsDisplay("accidents", accident_model,
network_table_panel, accident_table_panel));
        emergency.setFont(new Font("Arial", Font.PLAIN, 15));
        ButtonGroup bg = new ButtonGroup();
        bg.add(networks);
        bg.add(emergency);
        JButton update_button = new JButton("Оновлення");
        update_button.addActionListener(new UpdateNetworksAndAccidents(networks,
network_table, accident_table));
        update_button.setFont(new Font("Arial", Font.PLAIN, 15));
        top.add(networks);
        top.add(emergency);
        top.add(update_button);

        network_model.addColumn("ID");
        network_model.addColumn("Назва мережі");
        network_model.addColumn("Зв'язки");
        network_model.addColumn("Підстанції");
        network_model.addColumn("Статус");
        JScrollPane scrollPane = new JScrollPane(network_table);
        network_table_panel.add(scrollPane);

        accident_model.addColumn("ID");
        accident_model.addColumn("Інформація");
        accident_model.addColumn("Дата");
        accident_model.addColumn("Мережа");
        JScrollPane scrollPane1 = new JScrollPane(accident_table);
        accident_table_panel.add(scrollPane1);
        accident_table_panel.setVisible(false);

        JPanel search_panel = new JPanel();
        JTextField search_field = new JTextField(20);
        search_field.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton search_button = new JButton("Пошук");
        search_button.addActionListener(new NetworksAccidentsSearch(networks, network_model,
accident_model, search_field));
        search_button.setFont(new Font("Arial", Font.PLAIN, 15));
        search_panel.add(search_field);
        search_panel.add(search_button);

        JPanel edit_panel = new JPanel();
        JButton delete_button = new JButton("Видалення");
        delete_button.addActionListener(new NetworkAccidentDelete(networks, network_model,
accident_model, network_table, accident_table));
        delete_button.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton edit_button = new JButton("Редагування");
        edit_button.addActionListener(new EditingNetwork(networks, network_table));
        edit_button.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton add_button = new JButton("Додавання");
        add_button.addActionListener(new AddingNetwork(networks));
        add_button.setFont(new Font("Arial", Font.PLAIN, 15));
```

```
        edit_panel.add(delete_button);
        edit_panel.add(edit_button);
        edit_panel.add(add_button);

        jf.add(top);
        jf.add(network_table_panel);
        jf.add(accident_table_panel);
        jf.add(search_panel);
        jf.add(edit_panel);
        jf.setVisible(true);
    }
}
```

Файл AddNetworkGUI.java

```
package com.company.GUI;

import com.company.functional.CreatingNetwork;
import com.company.functional.NewEdge;
import com.company.functional.NewNode;

import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.awt.*;

public class AddNetworkGUI {

    private final JFrame jf;

    public AddNetworkGUI() {
        jf = new JFrame("New network");
        jf.setSize(1000, 330);
        jf.setLayout(new GridLayout(2, 1));

        JPanel nodes_panel = new JPanel();
        JPanel edges_panel = new JPanel(new GridLayout(3, 1));
        JComboBox<String> linked_node1 = new JComboBox<>(new String[]{});
        JComboBox<String> linked_node2 = new JComboBox<>(new String[]{});

        JPanel info = new JPanel();
        info.setBorder(new TitledBorder("Загальна інформація про мережу"));
        JLabel lname = new JLabel("Назва мережі: ");
        lname.setFont(new Font("Arial", Font.PLAIN, 15));
        JTextField name = new JTextField(20);
        name.setFont(new Font("Arial", Font.PLAIN, 15));
        JLabel lname1 = new JLabel("Кількість підстанцій: ");
        lname1.setFont(new Font("Arial", Font.PLAIN, 15));
        JTextField nodes = new JTextField(6);
        nodes.setFont(new Font("Arial", Font.PLAIN, 15));
        JLabel lname2 = new JLabel("Кількість зв'язків: ");
```

```
            lname2.setFont(new Font("Arial", Font.PLAIN, 15));
            JTextField edges = new JTextField(6);
            edges.setFont(new Font("Arial", Font.PLAIN, 15));
            JButton new_network_button = new JButton("OK");
            new_network_button.addActionListener(new CreatingNetwork(name, nodes, edges,
nodes_panel));
            new_network_button.setFont(new Font("Arial", Font.PLAIN, 15));
            info.add(lname);
            info.add(name);
            info.add(lname1);
            info.add(nodes);
            info.add(lname2);
            info.add(edges);
            info.add(new_network_button);

            JPanel edges_nodes_panel = new JPanel(new GridLayout(1, 2));

            nodes_panel.setBorder(new TitledBorder("Інформація про підстанції"));
            JLabel lname3 = new JLabel("Назва підстанції: ");
            lname3.setFont(new Font("Arial", Font.PLAIN, 15));
            JTextField node_name = new JTextField(20);
            node_name.setFont(new Font("Arial", Font.PLAIN, 15));
            JButton new_node_button = new JButton("OK");
            new_node_button.addActionListener(new NewNode(name, nodes, node_name, edges_panel,
linked_node1, linked_node2));
            new_node_button.setFont(new Font("Arial", Font.PLAIN, 15));
            nodes_panel.add(lname3);
            nodes_panel.add(node_name);
            nodes_panel.add(new_node_button);
            edges_nodes_panel.add(nodes_panel);
            nodes_panel.setVisible(false);

            edges_panel.setBorder(new TitledBorder("Інформація про ЛЕП"));
            JPanel edges_panel1 = new JPanel();
            JLabel lname4 = new JLabel("Підстанція 1: ");
            lname4.setFont(new Font("Arial", Font.PLAIN, 15));
            linked_node1.setFont(new Font("Arial", Font.PLAIN, 15));
            edges_panel1.add(lname4);
            edges_panel1.add(linked_node1);
            JPanel edges_panel2 = new JPanel();
            JLabel lname5 = new JLabel("Підстанція 2: ");
            lname5.setFont(new Font("Arial", Font.PLAIN, 15));
            linked_node2.setFont(new Font("Arial", Font.PLAIN, 15));
            edges_panel2.add(lname5);
            edges_panel2.add(linked_node2);
            JPanel edges_panel3 = new JPanel();
            JLabel lname6 = new JLabel("Протяжність лінії: ");
            lname6.setFont(new Font("Arial", Font.PLAIN, 15));
            JTextField weight = new JTextField(10);
            weight.setFont(new Font("Arial", Font.PLAIN, 15));
            JButton new_edge_button = new JButton("OK");
```

```
            new_edge_button.addActionListener(new NewEdge(name, edges, weight, linked_node1, linked_node2));
            new_edge_button.setFont(new Font("Arial", Font.PLAIN, 15));
            edges_panel3.add(lname6);
            edges_panel3.add(weight);
            edges_panel3.add(new_edge_button);
            edges_panel.add(edges_panel1);
            edges_panel.add(edges_panel2);
            edges_panel.add(edges_panel3);
            edges_nodes_panel.add(edges_panel);
            edges_panel.setVisible(false);

            jf.add(info);
            jf.add(edges_nodes_panel);
            jf.setVisible(true);
        }
    }
```

Файл NetworkGUI.java

```
package com.company.GUI;

import com.company.functional.*;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

public class NetworkGUI {
    private final JFrame jf;
    Network network;

    public NetworkGUI(Network network) {
        this.network = network;
        jf = new JFrame(network.getName());
        jf.setSize(600, 620);
        jf.setLayout(new FlowLayout());

        JPanel edges_table_panel = new JPanel();
        JPanel nodes_table_panel = new JPanel();
        DefaultTableModel edges_table_model = new DefaultTableModel();
        DefaultTableModel nodes_table_model = new DefaultTableModel();
        JTable edges_table = new JTable(edges_table_model);
        JTable nodes_table = new JTable(nodes_table_model);
        JPanel edge_status_panel = new JPanel();
        JPanel node_status_panel = new JPanel();


        JPanel top = new JPanel();
```

```
        JRadioButton substation = new JRadioButton("Підстанції");
        substation.addItemListener(new NodesEdgesDisplay("nodes", nodes_table_model,
nodes_table_panel, edges_table_panel, node_status_panel, edge_status_panel, network));
        substation.setFont(new Font("Arial", Font.PLAIN, 15));
        JRadioButton connections = new JRadioButton("Зв'язки");
        connections.addItemListener(new NodesEdgesDisplay("edges", edges_table_model,
nodes_table_panel, edges_table_panel, node_status_panel, edge_status_panel, network));
        connections.setFont(new Font("Arial", Font.PLAIN, 15));
        ButtonGroup bg = new ButtonGroup();
        bg.add(substation);
        bg.add(connections);
        JButton update_button = new JButton("Оновлення");
        update_button.addActionListener(new UpdateNetwork(substation, nodes_table, edges_table,
network));
        update_button.setFont(new Font("Arial", Font.PLAIN, 15));
        top.add(substation);
        top.add(connections);
        top.add(update_button);

        edges_table_model.addColumn("ID");
        edges_table_model.addColumn("Вага");
        edges_table_model.addColumn("Підстанція 1");
        edges_table_model.addColumn("Підстанція 2");
        edges_table_model.addColumn("Статус");
        JScrollPane scrollPane = new JScrollPane(edges_table);
        edges_table_panel.add(scrollPane);

        nodes_table_model.addColumn("ID");
        nodes_table_model.addColumn("Назва");
        nodes_table_model.addColumn("Вага");
        nodes_table_model.addColumn("Статус");
        JScrollPane scrollPane1 = new JScrollPane(nodes_table);
        nodes_table_panel.add(scrollPane1);
        nodes_table_panel.setVisible(false);

        JPanel edit_panel = new JPanel();
        JTextField search_field = new JTextField(20);
        search_field.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton search_button = new JButton("Пошук");
        search_button.addActionListener(new NodesEdgesSearch(substation, search_field,
nodes_table_model, edges_table_model, network));
        search_button.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton delete_button = new JButton("Видалити");
        delete_button.addActionListener(new NodeEdgeDelete(substation, nodes_table_model,
edges_table_model, nodes_table, edges_table, network));
        delete_button.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton add_button = new JButton("Додати");
        add_button.addActionListener(new NodeEdgeAdd(substation, nodes_table_model,
edges_table_model, network));
        add_button.setFont(new Font("Arial", Font.PLAIN, 15));
        edit_panel.add(search_field);
        edit_panel.add(search_button);
```

```
        edit_panel.add(delete_button);
        edit_panel.add(add_button);

        JRadioButton active_edge = new JRadioButton("Активний");
        active_edge.setFont(new Font("Arial", Font.PLAIN, 15));
        JRadioButton reserve_edge = new JRadioButton("Резерв");
        reserve_edge.setFont(new Font("Arial", Font.PLAIN, 15));
        JRadioButton emergency_edge = new JRadioButton("Аварійний");
        emergency_edge.setFont(new Font("Arial", Font.PLAIN, 15));
        ButtonGroup bg1 = new ButtonGroup();
        bg1.add(active_edge);
        bg1.add(reserve_edge);
        bg1.add(emergency_edge);
        JButton change_edge_status_button = new JButton("OK");
        change_edge_status_button.addActionListener(new ChangeEdgeStatus(active_edge,
reserve_edge, emergency_edge, edges_table, network));
        change_edge_status_button.setFont(new Font("Arial", Font.PLAIN, 15));
        edge_status_panel.add(active_edge);
        edge_status_panel.add(reserve_edge);
        edge_status_panel.add(emergency_edge);
        edge_status_panel.add(change_edge_status_button);

        JRadioButton active_node = new JRadioButton("Активний");
        active_node.setFont(new Font("Arial", Font.PLAIN, 15));
        JRadioButton inactive_node = new JRadioButton("Неактивний");
        inactive_node.setFont(new Font("Arial", Font.PLAIN, 15));
        JRadioButton emergency = new JRadioButton("Аварійний");
        emergency.setFont(new Font("Arial", Font.PLAIN, 15));
        ButtonGroup bg2 = new ButtonGroup();
        bg2.add(active_node);
        bg2.add(inactive_node);
        bg2.add(emergency);
        JButton change_node_status_button = new JButton("OK");
        change_node_status_button.addActionListener(new ChangeNodeStatus(active_node,
inactive_node, emergency, nodes_table, network));
        change_node_status_button.setFont(new Font("Arial", Font.PLAIN, 15));
        node_status_panel.add(active_node);
        node_status_panel.add(inactive_node);
        node_status_panel.add(emergency);
        node_status_panel.add(change_node_status_button);
        node_status_panel.setVisible(false);

        JPanel way_panel = new JPanel();
        JButton search_way_button = new JButton("Пошук шляху");
        search_way_button.addActionListener(new FindRoute(edges_table_model,
nodes_table_model, connections, substation, network));
        search_way_button.setFont(new Font("Arial", Font.PLAIN, 15));
        way_panel.add(search_way_button);

        jf.add(top);
        jf.add(edges_table_panel);
        jf.add(nodes_table_panel);
```

```
        jf.add(edit_panel);
        jf.add(edge_status_panel);
        jf.add(node_status_panel);
        jf.add(way_panel);
        jf.setVisible(true);
    }
}
```

Файл AddNodeGUI.java

```
package com.company.GUI;

import com.company.functional.AddNode;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.*;

import static javax.swing.SwingUtilities.getRootPane;

public class AddNodeGUI {
    private final JFrame jf;

    public AddNodeGUI(Network network) {
        jf = new JFrame("New Node");
        jf.setSize(550, 150);
        jf.setLayout(new GridLayout(1, 2));
        getRootPane(jf).setBorder(BorderFactory.createEmptyBorder(30, 0, 0, 0));

        JPanel nodes_panel = new JPanel();
        JLabel lname3 = new JLabel("Назва підстанції: ");
        lname3.setFont(new Font("Arial", Font.PLAIN, 15));
        JTextField node_name = new JTextField(20);
        node_name.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton new_node_button = new JButton("OK");
        new_node_button.addActionListener(new AddNode(node_name, network));
        new_node_button.setFont(new Font("Arial", Font.PLAIN, 15));
        nodes_panel.add(lname3);
        nodes_panel.add(node_name);
        nodes_panel.add(new_node_button);

        jf.add(nodes_panel);
        jf.setVisible(true);
    }
}
```

Файл AddEdgeGUI.java

```
package com.company.GUI;

import com.company.functional.AddEdge;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.awt.*;

public class AddEdgeGUI {
   private final JFrame jf;

   public AddEdgeGUI(Network network) {
      jf = new JFrame("New Edge");
      jf.setSize(400, 200);

      JPanel edges_panel = new JPanel(new GridLayout(3, 1));
      JPanel edges_panel1 = new JPanel();
      JLabel lname4 = new JLabel("Підстанція 1: ");
      lname4.setFont(new Font("Arial", Font.PLAIN, 15));
      JComboBox<String> linked_node1 = new JComboBox<>(network.getNodesNames(x ->
true));
      linked_node1.setFont(new Font("Arial", Font.PLAIN, 15));
      edges_panel1.add(lname4);
      edges_panel1.add(linked_node1);
      JPanel edges_panel2 = new JPanel();
      JLabel lname5 = new JLabel("Підстанція 2: ");
      lname5.setFont(new Font("Arial", Font.PLAIN, 15));
      JComboBox<String> linked_node2 = new JComboBox<>(network.getNodesNames(x ->
true));
      linked_node2.setFont(new Font("Arial", Font.PLAIN, 15));
      edges_panel2.add(lname5);
      edges_panel2.add(linked_node2);
      JPanel edges_panel3 = new JPanel();
      JLabel lname6 = new JLabel("Протяжність лінії: ");
      lname6.setFont(new Font("Arial", Font.PLAIN, 15));
      JTextField weight = new JTextField(10);
      weight.setFont(new Font("Arial", Font.PLAIN, 15));
      JButton new_edge_button = new JButton("OK");
      new_edge_button.addActionListener(new AddEdge(linked_node1, linked_node2, weight,
network));
      new_edge_button.setFont(new Font("Arial", Font.PLAIN, 15));
      edges_panel3.add(lname6);
      edges_panel3.add(weight);
      edges_panel3.add(new_edge_button);
      edges_panel.add(edges_panel1);
      edges_panel.add(edges_panel2);
      edges_panel.add(edges_panel3);

      jf.add(edges_panel);
      jf.setVisible(true);
   }
```

```
}
```

Файл AddAccidentGUI.java

```
package com.company.GUI;

import com.company.functional.AddAccident;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.*;

public class AddAccidentGUI {
    private final JFrame jf;

    public AddAccidentGUI(Network network) {
        jf = new JFrame("New Accident");
        jf.setSize(300, 200);

        JPanel jPanel = new JPanel();
        JLabel jLabel = new JLabel("Введіть інформацію про аварію");
        jLabel.setFont(new Font("Arial", Font.PLAIN, 15));
        JTextArea accident_info = new JTextArea(5, 20);
        accident_info.setFont(new Font("Arial", Font.PLAIN, 15));
        JButton add_accident = new JButton("OK");
        add_accident.addActionListener(new AddAccident(accident_info, network));
        add_accident.setFont(new Font("Arial", Font.PLAIN, 15));
        jPanel.add(jLabel);
        jPanel.add(accident_info);
        jPanel.add(add_accident);

        jf.add(jPanel);
        jf.setVisible(true);
    }
}
```

Файл AddAccident.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AddAccident implements ActionListener {

    private final JTextArea info;
```

```
    private final Network network;

    public AddAccident(JTextArea info, Network network) {
        this.info = info;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(!info.getText().equals("")){
            String text = info.getText();
            int network_id = network.getId();
            dbManager manager = new dbManager();
            manager.addAccident(text, network_id);
            info.setText("");
            JOptionPane.showMessageDialog(null, "Додавання виконано", "",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

Файл AddEdge.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Edge;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AddEdge implements ActionListener {

    private final JComboBox<String> linked_node1;
    private final JComboBox<String> linked_node2;
    private final JTextField edges_field;
    private final Network network;

    public AddEdge(JComboBox<String> linked_node1, JComboBox<String> linked_node2,
JTextField edges_field, Network network) {
        this.linked_node1 = linked_node1;
        this.linked_node2 = linked_node2;
        this.edges_field = edges_field;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
```

```
            dbManager manager = new dbManager();
            int weight = Integer.parseInt(edges_field.getText());
            String node_name1 = linked_node1.getItemAt(linked_node1.getSelectedIndex());
            String node_name2 = linked_node2.getItemAt(linked_node2.getSelectedIndex());
            int linked_node1_id = manager.getNodeID(node_name1, network.getId());
            int linked_node2_id = manager.getNodeID(node_name2, network.getId());
            manager.addEdge(weight, linked_node1_id, linked_node2_id, network.getId());
            edges_field.setText("");
            int id = manager.getEdgeID(linked_node1_id, linked_node2_id, "and");
            network.edges.add(new Edge(id, weight, node_name1, node_name2, "RESERVE"));
            JOptionPane.showMessageDialog(null, "Додавання виконано!", "OK",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

Файл AddNetwork.java

```
package com.company.functional;

import com.company.GUI.AddNetworkGUI;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AddingNetwork implements ActionListener {

    private final JRadioButton rbutton;

    public AddingNetwork(JRadioButton rbutton) {
        this.rbutton = rbutton;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(rbutton.isSelected()) {
            AddNetworkGUI addNetworkGUI = new AddNetworkGUI();
        }
    }
}
```

Файл AddNode.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Network;
import com.company.objects.Node;
```

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AddNode implements ActionListener {

    private final JTextField field;
    private final Network network;

    public AddNode(JTextField field, Network network) {
        this.field = field;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        String name = field.getText();
        dbManager manager = new dbManager();
        manager.addNode(network.getId(), name, 0);
        int id = manager.getNodeID(name, network.getId());
        network.nodes.add(new Node(id, name, 0, "INACTIVE"));
        JOptionPane.showMessageDialog(null, "Додавання виконано!", "OK",
JOptionPane.INFORMATION_MESSAGE);
        field.setText("");
    }
}
```

Файл Authorization.java

```
package com.company.functional;

import com.company.GUI.NetworksGUI;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Authorization implements ActionListener {

    private final JFrame frame;
    private final String password = "3754";
    private final JPasswordField password_to_check_field;

    public Authorization(JFrame jf, JPasswordField pass) {
        frame = jf;
        password_to_check_field = pass;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
```

```
            String password_to_check = String.valueOf(password_to_check_field.getPassword());
            if(password.equals(password_to_check)){
                frame.setVisible(false);
                NetworksGUI netgui = new NetworksGUI();
            } else {
                JOptionPane.showMessageDialog(null, "Пароль невірний!", "Message",
JOptionPane.WARNING_MESSAGE);
                password_to_check_field.setText("");
            }
        }
    }
```

Файл ChangeEdgeStatus.java

```
package com.company.functional;

import com.company.GUI.AddAccidentGUI;
import com.company.objects.Edge;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeEdgeStatus implements ActionListener {

    private final JRadioButton active_edge;
    private final JRadioButton reserve_edge;
    private final JRadioButton emergency_edge;
    private final JTable edges_table;
    private final Network network;

    public ChangeEdgeStatus(JRadioButton active_edge, JRadioButton reserve_edge, JRadioButton
emergency_edge, JTable edges_table, Network network) {
        this.active_edge = active_edge;
        this.reserve_edge = reserve_edge;
        this.emergency_edge = emergency_edge;
        this.edges_table = edges_table;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(edges_table.getSelectedRowCount() > 0){
            int[] selected = edges_table.getSelectedRows();
            for(int i : selected){
                String previous_status = edges_table.getValueAt(i, 4).toString();
                if(active_edge.isSelected() && !previous_status.equals("ACTIVE")){
                    String linked_node_2 = edges_table.getValueAt(i, 3).toString();
                    for(Edge edge : network.edges){
```

```
                    if(edge.getLinked_node_2().equals(linked_node_2) &&
edge.getStatus().equals("ACTIVE")){
                        JOptionPane.showMessageDialog(null, linked_node_2 + " вже заживлена!", "",
JOptionPane.INFORMATION_MESSAGE);
                        return;
                    }
                }
                edges_table.setValueAt("ACTIVE", i, 4);
                network.getEdge((int)edges_table.getValueAt(i, 0)).setStatus("ACTIVE");
            }
            else if(reserve_edge.isSelected() && !previous_status.equals("RESERVE")){
                edges_table.setValueAt("RESERVE", i, 4);
                network.getEdge((int)edges_table.getValueAt(i, 0)).setStatus("RESERVE");
            }
            else if(emergency_edge.isSelected() && !previous_status.equals("EMERGENCY")){
                edges_table.setValueAt("EMERGENCY", i, 4);
                network.getEdge((int)edges_table.getValueAt(i, 0)).setStatus("EMERGENCY");
                AddAccidentGUI addAccidentGUI = new AddAccidentGUI(network);
            }
        }
    }
}
```

Файл ChangeNodeStatus.java

```
package com.company.functional;

import com.company.GUI.AddAccidentGUI;
import com.company.objects.Network;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeNodeStatus implements ActionListener {

    private final JRadioButton active_node;
    private final JRadioButton inactive_node;
    private final JRadioButton emergency_node;
    private final JTable nodes_table;
    private final Network network;

    public ChangeNodeStatus(JRadioButton active_node, JRadioButton inactive_node,
JRadioButton emergency_node, JTable nodes_table, Network network) {
        this.active_node = active_node;
        this.inactive_node = inactive_node;
        this.emergency_node = emergency_node;
        this.nodes_table = nodes_table;
        this.network = network;
```

```
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(nodes_table.getSelectedRowCount() > 0){
            int[] selected = nodes_table.getSelectedRows();
            for(int i : selected){
                String previous_status = nodes_table.getValueAt(i, 3).toString();
                String node_name = nodes_table.getValueAt(i, 1).toString();
                if(active_node.isSelected() && !previous_status.equals("ACTIVE")){
                    nodes_table.setValueAt("ACTIVE", i, 3);
                    network.getNode((int)nodes_table.getValueAt(i, 0)).setStatus("ACTIVE");
                    for(int j = 0; j < network.edges.size(); ++j){
                        if(network.edges.get(j).getLinked_node_1().equals(node_name) ||
                            network.edges.get(j).getLinked_node_2().equals(node_name))
                        {
                            network.edges.get(j).setStatus("RESERVE");
                        }
                    }
                }
                else if((inactive_node.isSelected() && !previous_status.equals("INACTIVE")) ||
                        (emergency_node.isSelected() && !previous_status.equals("EMERGENCY")))
                {
                    String new_status = "";
                    if(inactive_node.isSelected())
                        new_status = "INACTIVE";
                    else
                        new_status = "EMERGENCY";
                    nodes_table.setValueAt(new_status, i, 3);
                    network.getNode((int)nodes_table.getValueAt(i, 0)).setStatus(new_status);
                    for(int j = 0; j < network.edges.size(); ++j){
                        if(network.edges.get(j).getLinked_node_1().equals(node_name) ||
                                network.edges.get(j).getLinked_node_2().equals(node_name))
                        {
                            network.edges.get(j).setStatus("EMERGENCY");
                        }
                    }
                    if(new_status.equals("EMERGENCY"))
                    {
                        AddAccidentGUI addAccidentGUI = new AddAccidentGUI(network);
                    }
                }
            }
        }
    }
}
```

Файл CreatingNetwork.java

```
package com.company.functional;
```

```
import com.company.database.dbManager;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CreatingNetwork implements ActionListener {

    private final JTextField name;
    private final JTextField nodes_count;
    private final JTextField edges_count;
    private final JPanel node_panel;

    public CreatingNetwork(JTextField name, JTextField nodes_count, JTextField edges_count,
JPanel node_panel) {
        this.name = name;
        this.nodes_count = nodes_count;
        this.edges_count = edges_count;
        this.node_panel = node_panel;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(!name.getText().equals("") && !nodes_count.getText().equals("") &&
!edges_count.getText().equals("")) {
            String sname = name.getText();
            int n_count = Integer.parseInt(nodes_count.getText());
            int e_count = Integer.parseInt(edges_count.getText());
            dbManager manager = new dbManager();
            manager.createNetwork(sname, e_count, n_count);
            node_panel.setVisible(true);
        }
    }
}
```

Файл EditingNetwork.java

```
package com.company.functional;

import com.company.GUI.NetworkGUI;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class EditingNetwork implements ActionListener {
```

```
    private final JRadioButton rbutton;
    private final JTable table;

    public EditingNetwork(JRadioButton rbutton, JTable table) {
        this.rbutton = rbutton;
        this.table = table;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(rbutton.isSelected()) {
            if (table.getSelectedRowCount() == 1) {
                int network_id = (int) table.getValueAt(table.getSelectedRow(), 0);
                String network_name = table.getValueAt(table.getSelectedRow(), 1).toString();
                int edges_count = (int) table.getValueAt(table.getSelectedRow(), 2);
                int nodes_count = (int) table.getValueAt(table.getSelectedRow(), 3);
                String network_status = table.getValueAt(table.getSelectedRow(), 4).toString();
                NetworkGUI networkGUI = new NetworkGUI(new Network(network_id,
network_name, edges_count, nodes_count, network_status));
            }
        }
    }
}
```

Файл FindRoute.java

```
package com.company.functional;

import com.company.algorithm.Route;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class FindRoute implements ActionListener {

    private final DefaultTableModel edges_model;
    private final DefaultTableModel nodes_model;
    private final JRadioButton edges_button;
    private final JRadioButton nodes_button;
    private final Network network;

    public FindRoute(DefaultTableModel edges_model, DefaultTableModel nodes_model,
JRadioButton edges_button, JRadioButton nodes_button, Network network) {
        this.edges_model = edges_model;
        this.nodes_model = nodes_model;
```

```
        this.edges_button = edges_button;
        this.nodes_button = nodes_button;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(edges_button.isSelected()) {
            Route route = new Route(network);
            if(!route.checkCurrentRoute()) {
                if(route.checkPossibleRoute()) {
                    route.buildRoute();
                    while (edges_model.getRowCount() > 0) edges_model.removeRow(0);
                    int counter = 0;
                    for (Edge edge : network.edges) {
                        edges_model.insertRow(counter, edge.getArray());
                        counter++;
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Неможливо заживити всі підстанції", "",
JOptionPane.INFORMATION_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(null, "Всі активні підстанції заживлені", "",
JOptionPane.INFORMATION_MESSAGE);
            }
        } else if(nodes_button.isSelected()) {
            Route route = new Route(network);
            route.rebuildRoute();
            while(nodes_model.getRowCount() > 0) nodes_model.removeRow(0);
            int counter = 0;
            for(Node node : network.nodes){
                nodes_model.insertRow(counter, node.getArray());
                counter++;
            }
        }
    }
```

Файл NetworkAccidentDeletet.java

```
package com.company.functional;

import com.company.GUI.NetworkGUI;
import com.company.database.dbManager;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;

public class NetworkAccidentDelete implements ActionListener {

    private final JRadioButton rbutton;
    private final DefaultTableModel netw_model;
    private final DefaultTableModel accident_model;
    private final JTable netw_table;
    private final JTable accident_table;

    public NetworkAccidentDelete(JRadioButton rbutton, DefaultTableModel netw_model,
DefaultTableModel accident_model, JTable netw_table, JTable accident_table) {
        this.rbutton = rbutton;
        this.netw_model = netw_model;
        this.accident_model = accident_model;
        this.netw_table = netw_table;
        this.accident_table = accident_table;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(rbutton.isSelected()){
            dbManager dbManager = new dbManager();
            if(netw_table.getSelectedRowCount() > 0) {
                int[] selected = netw_table.getSelectedRows();
                int counter = 0;
                for(int i : selected){
                    int network_id = (int) netw_table.getValueAt(i, 0);
                    dbManager.deleteNetwork(network_id);
                    netw_model.removeRow(i - counter);
                    counter++;
                }
            }
        } else {
            dbManager dbManager = new dbManager();
            if(accident_table.getSelectedRowCount() > 0) {
                int[] selected = accident_table.getSelectedRows();
                int counter = 0;
                for(int i : selected){
                    int accident_id = (int) accident_table.getValueAt(i, 0);
                    dbManager.deleteAccident(accident_id);
                    accident_model.removeRow(i - counter);
                    counter++;
                }
            }
        }
    }
}
```

Файл NetworksAccidentsDisplay.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Accident;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.ArrayList;

public class NetworksAccidentsDisplay implements ItemListener {

    private final String option;
    private final DefaultTableModel model;
    private final JPanel network_table_panel;
    private final JPanel accident_table_panel;

    public NetworksAccidentsDisplay(String option, DefaultTableModel model, JPanel
network_table_panel, JPanel accident_table_panel) {
        this.option = option;
        this.model = model;
        this.network_table_panel = network_table_panel;
        this.accident_table_panel = accident_table_panel;
    }

    @Override
    public void itemStateChanged(ItemEvent itemEvent) {
        if(option.equals("networks")){
            if(itemEvent.getStateChange() == ItemEvent.SELECTED) {
                accident_table_panel.setVisible(false);
                network_table_panel.setVisible(true);
                dbManager manager = new dbManager();
                ArrayList<Network> networks = manager.getNetworks();
                int counter = 0;
                while(model.getRowCount() > 0) model.removeRow(0);
                for(Network network : networks){
                    model.insertRow(counter, network.getArray());
                    counter++;
                }
            }
        } else {
            if(itemEvent.getStateChange() == ItemEvent.SELECTED) {
                network_table_panel.setVisible(false);
                accident_table_panel.setVisible(true);
                dbManager manager = new dbManager();
                ArrayList<Accident> accidents = manager.getAccidents();
                int counter = 0;
                while(model.getRowCount() > 0) model.removeRow(0);
                for(Accident accident : accidents){
                    model.insertRow(counter, accident.getArray());
```

```
                counter++;
            }
        }
    }
}
```

Файл NetworksAccidentsSearch.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Accident;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class NetworksAccidentsSearch implements ActionListener {

    private final JRadioButton rbutton;
    private final DefaultTableModel netw_model;
    private final DefaultTableModel accident_model;
    private final JTextField field;

    public NetworksAccidentsSearch(JRadioButton rbutton, DefaultTableModel netw_model,
DefaultTableModel accident_model, JTextField field) {
        this.rbutton = rbutton;
        this.netw_model = netw_model;
        this.accident_model = accident_model;
        this.field = field;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        String text = field.getText();
        if(rbutton.isSelected()){
            dbManager manager = new dbManager();
            ArrayList<Network> networks;
            if(text.equals("")){
                networks = manager.getNetworks();
                int counter = 0;
                while(netw_model.getRowCount() > 0) netw_model.removeRow(0);
                for(Network network : networks){
                    netw_model.insertRow(counter, network.getArray());
                    counter++;
                }
```

```
            } else {
                networks = manager.searchNetworks(text);
                int counter = 0;
                while(netw_model.getRowCount() > 0) netw_model.removeRow(0);
                for(Network network : networks){
                    netw_model.insertRow(counter, network.getArray());
                    counter++;
                }
            }
        } else {
            dbManager manager = new dbManager();
            ArrayList<Accident> accidents;
            if(text.equals("")) {
                accidents = manager.getAccidents();
                int counter = 0;
                while(accident_model.getRowCount() > 0) accident_model.removeRow(0);
                for(Accident accident : accidents){
                    accident_model.insertRow(counter, accident.getArray());
                    counter++;
                }
            } else {
                accidents = manager.searchAccidents(text);
                int counter = 0;
                while(accident_model.getRowCount() > 0) accident_model.removeRow(0);
                for(Accident accident : accidents){
                    accident_model.insertRow(counter, accident.getArray());
                    counter++;
                }
            }
        }
    }
}
```

Файл NewEdge.java

```
package com.company.functional;

import com.company.database.dbManager;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class NewEdge implements ActionListener {

    private static int counter = 0;
    private static int network_id = 0;
    private final JTextField network_name;
    private final JTextField edge_weight;
    private final JComboBox<String> linked_node1;
```

```
    private final JComboBox<String> linked_node2;
    private final JTextField edges_field;

    public NewEdge(JTextField network_name, JTextField edges_field, JTextField edge_weight,
JComboBox<String> linked_node1, JComboBox<String> linked_node2) {
        this.network_name = network_name;
        this.edge_weight = edge_weight;
        this.linked_node1 = linked_node1;
        this.linked_node2 = linked_node2;
        this.edges_field = edges_field;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(!edge_weight.getText().equals("")) {
            counter++;
            dbManager manager = new dbManager();
            if (counter == 1)
                network_id = manager.getNetworkID(network_name.getText());
            int weight = Integer.parseInt(edge_weight.getText());
            String node_name1 = linked_node1.getItemAt(linked_node1.getSelectedIndex());
            String node_name2 = linked_node2.getItemAt(linked_node2.getSelectedIndex());
            int linked_node1_id = manager.getNodeID(node_name1, network_id);
            int linked_node2_id = manager.getNodeID(node_name2, network_id);
            manager.addEdge(weight, linked_node1_id, linked_node2_id, network_id);
            edge_weight.setText("");
            if (counter == Integer.parseInt(edges_field.getText())) {
                JOptionPane.showMessageDialog(null, "Мережу створено!", "Нова мережа",
JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}
```

Файл NewNode.java

```
package com.company.functional;

import com.company.database.dbManager;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class NewNode implements ActionListener {

    private final JTextField network_name;
    private final JTextField nodes_field;
    private final JTextField node_name;
    private static int counter = 0;
```

```
    private static int network_id = 0;
    private final JPanel edge_panel;
    private final JComboBox<String> linked_node1;
    private final JComboBox<String> linked_node2;

    public NewNode(JTextField network_name, JTextField nodes_field , JTextField node_name,
JPanel edge_panel, JComboBox<String> linked_node1, JComboBox<String> linked_node2) {
        this.network_name = network_name;
        this.nodes_field = nodes_field;
        this.node_name = node_name;
        this.edge_panel = edge_panel;
        this.linked_node1 = linked_node1;
        this.linked_node2 = linked_node2;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(!node_name.getText().equals("")) {
            counter++;
            dbManager manager = new dbManager();
            if (counter == 1)
                network_id = manager.getNetworkID(network_name.getText());
            int source = 0;
            if (counter == 1)
                source = 1;
            String node_sname = node_name.getText();
            linked_node1.addItem(node_sname);
            linked_node2.addItem(node_sname);
            manager.addNode(network_id, node_sname, source);
            node_name.setText("");
            if (counter == Integer.parseInt(nodes_field.getText())) {
                edge_panel.setVisible(true);
            }
        }
    }
}
```

Файл NodeEdgeAdd.java

```
package com.company.functional;

import com.company.GUI.AddEdgeGUI;
import com.company.GUI.AddNodeGUI;
import com.company.objects.Network;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
public class NodeEdgeAdd implements ActionListener {

    private final JRadioButton rbutton;
    private final DefaultTableModel node_model;
    private final DefaultTableModel edge_model;
    private final Network network;

    public NodeEdgeAdd(JRadioButton rbutton, DefaultTableModel node_model,
DefaultTableModel edge_model, Network network) {
        this.rbutton = rbutton;
        this.node_model = node_model;
        this.edge_model = edge_model;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(rbutton.isSelected()){
            AddNodeGUI addNodeGUI = new AddNodeGUI(network);
        } else {
            AddEdgeGUI addEdgeGUI = new AddEdgeGUI(network);
        }
    }
}
```

Файл NodeEdgeDelete.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class NodeEdgeDelete implements ActionListener {

    private final JRadioButton rbutton;
    private final DefaultTableModel node_model;
    private final DefaultTableModel edge_model;
    private final JTable node_table;
    private final JTable edge_table;
    private final Network network;
```

```
        public NodeEdgeDelete(JRadioButton rbutton, DefaultTableModel node_model,
DefaultTableModel edge_model, JTable node_table, JTable edge_table, Network network) {
            this.rbutton = rbutton;
            this.node_model = node_model;
            this.edge_model = edge_model;
            this.node_table = node_table;
            this.edge_table = edge_table;
            this.network = network;
        }

        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            if(rbutton.isSelected()){
                if(node_table.getSelectedRowCount() > 0){
                    dbManager manager = new dbManager();
                    int[] selected = node_table.getSelectedRows();
                    int counter = 0;
                    for(int i : selected){
                        int node_id = (int)node_table.getValueAt(i, 0);
                        manager.deleteNode(node_id, network.getId());
                        node_model.removeRow(i - counter);
                        Node node = network.nodes.remove(network.getNodeIndex(node_id));
                        ArrayList<Edge> edges = new ArrayList<>();
                        for(Edge edge : network.edges){
                            if(!edge.getLinked_node_1().equals(node.getName())                    &&
!edge.getLinked_node_2().equals(node.getName())){
                                edges.add(edge);
                            }
                        }
                        network.edges = edges;
                        counter++;
                    }
                }
            } else {
                if(edge_table.getSelectedRowCount() > 0){
                    dbManager manager = new dbManager();
                    int[] selected = edge_table.getSelectedRows();
                    int counter = 0;
                    for(int i : selected){
                        int edge_id = (int)edge_table.getValueAt(i, 0);
                        manager.deleteEdge(edge_id, network.getId());
                        edge_model.removeRow(i - counter);
                        network.edges.remove(network.getEdgeIndex(edge_id));
                        counter++;
                    }
                }
            }
        }
```

Файл NodesEdgesDisplay.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.ArrayList;

public class NodesEdgesDisplay implements ItemListener {

    private final String option;
    private final DefaultTableModel model;
    private final JPanel nodes_table_panel;
    private final JPanel edges_table_panel;
    private final JPanel node_status_panel;
    private final JPanel edge_status_panel;
    private final Network network;

    public NodesEdgesDisplay(String option, DefaultTableModel model, JPanel nodes_table_panel,
JPanel edges_table_panel, JPanel node_status_panel, JPanel edge_status_panel, Network network) {
        this.option = option;
        this.model = model;
        this.nodes_table_panel = nodes_table_panel;
        this.edges_table_panel = edges_table_panel;
        this.node_status_panel = node_status_panel;
        this.edge_status_panel = edge_status_panel;
        this.network = network;
    }

    @Override
    public void itemStateChanged(ItemEvent itemEvent) {
        if(option.equals("nodes")){
            if(itemEvent.getStateChange() == ItemEvent.SELECTED){
                edges_table_panel.setVisible(false);
                nodes_table_panel.setVisible(true);
                edge_status_panel.setVisible(false);
                node_status_panel.setVisible(true);
                if(network.nodes == null) {
                    dbManager manager = new dbManager();
                    network.nodes = manager.getNodes(network.getId());
                }
                int counter = 0;
                while(model.getRowCount() > 0) model.removeRow(0);
                for(Node node : network.nodes){
                    model.insertRow(counter, node.getArray());
                    counter++;
                }
```

```
            }
        } else {
            if(itemEvent.getStateChange() == ItemEvent.SELECTED){
                nodes_table_panel.setVisible(false);
                edges_table_panel.setVisible(true);
                node_status_panel.setVisible(false);
                edge_status_panel.setVisible(true);
                if(network.edges == null) {
                    dbManager manager = new dbManager();
                    network.edges = manager.getEdges(network.getId());
                }
                int counter = 0;
                while(model.getRowCount() > 0) model.removeRow(0);
                for(Edge edge : network.edges){
                    model.insertRow(counter, edge.getArray());
                    counter++;
                }
            }
        }
    }
}
```

Файл NodesEdgesSearch.java

```
package com.company.functional;

import com.company.database.dbManager;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class NodesEdgesSearch implements ActionListener {

    private final JRadioButton rbutton;
    private final JTextField field;
    private final DefaultTableModel nodes_model;
    private final DefaultTableModel edges_model;
    private final Network network;

    public NodesEdgesSearch(JRadioButton rbutton, JTextField field, DefaultTableModel
nodes_model, DefaultTableModel edges_model, Network network) {
        this.rbutton = rbutton;
        this.field = field;
        this.nodes_model = nodes_model;
```

```
        this.edges_model = edges_model;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(rbutton.isSelected()){
            String text = field.getText();
            int counter = 0;
            while(nodes_model.getRowCount() > 0) nodes_model.removeRow(0);
            for(Node node : network.nodes){
                nodes_model.insertRow(counter, node.getArray());
                counter++;
            }
            if(!text.equals("")){
                int i = 0, j = 0;
                for(Node node : network.nodes){
                    if(!node.getName().contains(text) && !node.getStatus().contains(text)){
                        nodes_model.removeRow(i - j);
                        j++;
                    }
                    i++;
                }
            }
        } else {
            String text = field.getText();
            int counter = 0;
            while(edges_model.getRowCount() > 0) edges_model.removeRow(0);
            for(Edge edge : network.edges){
                edges_model.insertRow(counter, edge.getArray());
                counter++;
            }
            if(!text.equals("")){
                int i = 0, j = 0;
                for(Edge edge : network.edges){
                    if(!edge.getLinked_node_1().contains(text)                    &&
!edge.getLinked_node_2().contains(text) && !edge.getStatus().contains(text)){
                        edges_model.removeRow(i - j);
                        j++;
                    }
                    i++;
                }
            }
        }
    }
```

Файл UpdateNetwork.java

```
package com.company.functional;
```

```
import com.company.database.dbManager;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UpdateNetwork implements ActionListener {

    private final JRadioButton nodes_button;
    private final JTable nodes_table;
    private final JTable edges_table;
    private  final Network network;

    public UpdateNetwork(JRadioButton nodes_button, JTable nodes_table, JTable edges_table,
Network network) {
        this.nodes_button = nodes_button;
        this.nodes_table = nodes_table;
        this.edges_table = edges_table;
        this.network = network;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(nodes_button.isSelected()){
            dbManager manager = new dbManager();
            if(nodes_table.getSelectedRowCount() == 0){
                for(Node node : network.nodes){
                    manager.updateNode(node.getStatus(), node.getId());
                }
            } else {
                int[] selected = nodes_table.getSelectedRows();
                for(int i : selected){
                    manager.updateNode(nodes_table.getValueAt(i,                3).toString(),
(int)nodes_table.getValueAt(i, 0));
                }
            }
            JOptionPane.showMessageDialog(null,          "Оновлення          виконано!",          "",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            dbManager manager = new dbManager();
            if(edges_table.getSelectedRowCount() == 0){
                for(Edge edge : network.edges){
                    manager.updateEdge(edge.getStatus(), edge.getId());
                }
            } else {
                int[] selected = edges_table.getSelectedRows();
                for(int i : selected){
```

```
                    manager.updateEdge(edges_table.getValueAt(i, 4).toString(),
(int)edges_table.getValueAt(i, 0));
                }
            }
            JOptionPane.showMessageDialog(null, "Оновлення виконано!", "",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

Файл UpdateNetworksAndAccidents.java

```
package com.company.functional;

import com.company.database.dbManager;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UpdateNetworksAndAccidents implements ActionListener {

    private final JRadioButton networks;
    private final JTable networks_table;
    private final JTable accidents_table;

    public UpdateNetworksAndAccidents(JRadioButton networks, JTable networks_table, JTable
accidents_table) {
        this.networks = networks;
        this.networks_table = networks_table;
        this.accidents_table = accidents_table;
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if(networks.isSelected()){
            dbManager manager = new dbManager();
            if(networks_table.getSelectedRowCount() == 0){
                for(int i = 0; i < networks_table.getRowCount(); ++i){
                    manager.updateNetwork(networks_table.getValueAt(i, 1).toString(),
networks_table.getValueAt(i, 4).toString(), (int)networks_table.getValueAt(i, 0));
                }
            } else {
                int[] selected = networks_table.getSelectedRows();
                for(int i : selected){
                    manager.updateNetwork(networks_table.getValueAt(i, 1).toString(),
networks_table.getValueAt(i, 4).toString(), (int)networks_table.getValueAt(i, 0));
                }
            }
```

```
                JOptionPane.showMessageDialog(null, "Оновлення виконано!", "",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                dbManager manager = new dbManager();
                if(accidents_table.getSelectedRowCount() == 0){
                    for(int i = 0; i < accidents_table.getRowCount(); ++i){
                        manager.updateAccident(accidents_table.getValueAt(i, 1).toString(),
(int)accidents_table.getValueAt(i, 0));
                    }
                } else {
                    int[] selected = accidents_table.getSelectedRows();
                    for(int i : selected){
                        manager.updateAccident(accidents_table.getValueAt(i, 1).toString(),
(int)accidents_table.getValueAt(i, 0));
                    }
                }
                JOptionPane.showMessageDialog(null, "Оновлення виконано!", "",
JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
```

Файл dbManager.java

```
package com.company.database;

import com.company.objects.Accident;
import com.company.objects.Edge;
import com.company.objects.Network;
import com.company.objects.Node;

import java.sql.*;
import java.util.ArrayList;

public class dbManager {
    private final static String url =
"jdbc:mysql://localhost:3306/electrical_networks?autoReconnect=true";
    private final static String username = "root";
    private final static String password = "root";
    private static Connection conn;
    private static Statement statement;

    static {
        try {
            conn = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
        static {
            try {
                statement = conn.createStatement();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public ArrayList<Network> getNetworks(){
            try {
                ResultSet resset = statement.executeQuery("SELECT * FROM networks");
                ArrayList<Network> networks = new ArrayList<>();
                while (resset.next()){
                    networks.add(new Network(resset.getInt(1), resset.getString(2), resset.getInt(3),
resset.getInt(4), resset.getString(5)));
                }
                return networks;
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return null;
        }

        public ArrayList<Accident> getAccidents(){
            try{
                String query = "SELECT a.id, a.info, a.date, n.name FROM accidents a inner join networks
n on a.id_nw = n.id";
                ResultSet resset = statement.executeQuery(query);
                ArrayList<Accident> accidents = new ArrayList<>();
                while (resset.next()){
                    accidents.add(new Accident(resset.getInt(1), resset.getString(2), resset.getString(3),
resset.getString(4)));
                }
                return accidents;
            } catch (SQLException e){
                e.printStackTrace();
            }
            return null;
        }

        public ArrayList<Node> getNodes(int network_id){
            try{
                String query = "SELECT id, name, if(weight is null, 0, weight), node_status FROM nodes
WHERE id_nw = " + network_id + " order by source desc";
                ResultSet resset = statement.executeQuery(query);
                ArrayList<Node> nodes = new ArrayList<>();
                while (resset.next()){
                    nodes.add(new Node(resset.getInt(1), resset.getString(2), resset.getInt(3),
resset.getString(4)));
                }
                return nodes;
            } catch (SQLException e){
```

```
                e.printStackTrace();
            }
            return null;
        }

        public ArrayList<Edge> getEdges(int network_id){
            try{
                String query = "SELECT e.id, e.weight, n.name, nn.name, e.edge_status FROM edges e inner join nodes n on e.linked_node1_id = n.id inner join nodes nn on e.linked_node2_id = nn.id WHERE e.id_nw = " + network_id;
                ResultSet resset = statement.executeQuery(query);
                ArrayList<Edge> edges = new ArrayList<>();
                while (resset.next()){
                    edges.add(new Edge(resset.getInt(1), resset.getInt(2), resset.getString(3), resset.getString(4), resset.getString(5)));
                }
                return edges;
            } catch (SQLException e){
                e.printStackTrace();
            }
            return null;
        }

        public void createNetwork(String name, int edges, int nodes){
            String query = "INSERT INTO networks (id, name, edges_count, nodes_count, status) VALUES (NULL, '" + name + "', " + edges + ", " + nodes + ", 'INACTIVE')";
            updateTable(query);
        }

        public int getNetworkID(String network_name){
            String query = "SELECT id FROM networks WHERE name = '" + network_name + "'";
            return getID(query);
        }

        public void addNode(int network_id, String name, int source){
            String query = "INSERT INTO nodes (id, id_nw, weight, node_status, name, source) VALUES (NULL, " + network_id + ", NULL, 'ACTIVE', '" + name + "', " + source + ")";
            updateTable(query);
            query = "UPDATE networks SET nodes_count = nodes_count + 1 WHERE id = " + network_id;
            updateTable(query);
        }

        public int getNodeID(String node_name, int network_id){
            String query = "SELECT id FROM nodes WHERE name = '" + node_name + "' and id_nw = " + network_id;
            return getID(query);
        }

        public int getEdgeID(int linked_node1_id, int linked_node2_id, String op){
            String query = "SELECT id FROM edges WHERE linked_node1_id = " + linked_node1_id + " " + op + " linked_node2_id = " + linked_node2_id;
```

```
            return getID(query);
        }

        private int getID(String query){
            int id = 0;
            try{
                ResultSet resset = statement.executeQuery(query);
                resset.next();
                id = resset.getInt(1);
            } catch (SQLException e){
                e.printStackTrace();
            }
            return id;
        }

        public void addEdge(int weight, int linked_node1_id, int linked_node2_id, int network_id){
            String query = "INSERT INTO edges (id, weight, edge_status, linked_node1_id, linked_node2_id, id_nw) VALUES (NULL, " + weight + ", 'RESERVE', " + linked_node1_id + ", " + linked_node2_id + ", " + network_id + ")";
            updateTable(query);
            query = "UPDATE networks SET edges_count = edges_count + 1 WHERE id = " + network_id;
            updateTable(query);
        }

        public  void updateEdge(String status, int id){
            String query = "UPDATE edges SET edge_status = '" + status + "' WHERE id = " + id;
            updateTable(query);
        }

        public  void updateNode(String status, int id){
            String query = "UPDATE nodes SET node_status = '" + status + "' WHERE id = " + id;
            updateTable(query);
        }

        public void updateNetwork(String name, String status, int id){
            String query = "UPDATE networks SET name = '" + name + "', status = '" + status + "' WHERE id = " + id;
            updateTable(query);
        }

        public void updateAccident(String info, int id){
            String query = "UPDATE accidents SET info = '" + info + "' WHERE id = " + id;
            updateTable(query);
        }

        public void addAccident(String info, int network_id){
            String query = "INSERT INTO accidents (id, info, date, id_nw) VALUES (NULL, '" + info + "', NOW(), " + network_id + ")";
            updateTable(query);
        }
```

```
    private void updateTable(String query){
        try{
            statement.executeUpdate(query);
        } catch (SQLException e){
            e.printStackTrace();
        }
    }

    public ArrayList<Network> searchNetworks(String text){
        try {
            ResultSet resset = statement.executeQuery("SELECT * FROM networks WHERE name LIKE '%" + text + "%' OR status LIKE '%" + text + "%'");
            ArrayList<Network> networks = new ArrayList<>();
            while (resset.next()){
                networks.add(new Network(resset.getInt(1), resset.getString(2), resset.getInt(3), resset.getInt(4), resset.getString(5)));
            }
            return networks;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }

    public ArrayList<Accident> searchAccidents(String text){
        try{
            String query = "SELECT a.id, a.info, a.date, n.name FROM accidents a inner join networks n on a.id_nw = n.id WHERE a.info LIKE '%" + text + "%' OR n.name LIKE '%" + text + "%'";
            ResultSet resset = statement.executeQuery(query);
            ArrayList<Accident> accidents = new ArrayList<>();
            while (resset.next()){
                accidents.add(new Accident(resset.getInt(1), resset.getString(2), resset.getString(3), resset.getString(4)));
            }
            return accidents;
        } catch (SQLException e){
            e.printStackTrace();
        }
        return null;
    }

    public void deleteNetwork(int id){
        deleteRow("networks", id);
    }

    public void deleteAccident(int id){
        deleteRow("accidents", id);
    }


    public void deleteNode(int id, int network_id){
        deleteRow("nodes", id);
```

```
            String query = "UPDATE networks SET nodes_count = nodes_count - 1 WHERE id = " +
network_id;
            updateTable(query);
        }

        public void deleteEdge(int id, int network_id){
            deleteRow("edges", id);
            String query = "UPDATE networks SET edges_count = edges_count - 1 WHERE id = " +
network_id;
            updateTable(query);
        }

        private void deleteRow(String table, int id){
            try{
                String query = "DELETE FROM " + table + " WHERE id = " + id;
                statement.executeUpdate(query);
            } catch (SQLException e){
                e.printStackTrace();
            }
        }
    }
```

Файл GEdge.java

```
package com.company.algorithm.graph;

public class GEdge {
    public int source, dest, weight;

    public GEdge(int source, int dest, int weight)
    {
        this.source = source;
        this.dest = dest;
        this.weight = weight;
    }
}
```

Файл GNode.java

```
package com.company.algorithm.graph;

public class GNode {
    public int vertex, weight;

    public GNode(int vertex, int weight)
    {
        this.vertex = vertex;
        this.weight = weight;
```

```
    }
}
```

Файл Graph.java

```
package com.company.algorithm.graph;

import java.util.ArrayList;
import java.util.List;

public class Graph {
    public List<List<GEdge>> adjList = null;

    public Graph(List<GEdge> edges, int n)
    {
        adjList = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            adjList.add(new ArrayList<>());
        }

        for (GEdge edge: edges) {
            adjList.get(edge.source).add(edge);
        }
    }
}
```

Файл Route.java

```
package com.company.algorithm;

import com.company.algorithm.graph.GEdge;
import com.company.algorithm.graph.GNode;
import com.company.algorithm.graph.Graph;
import com.company.objects.Edge;
import com.company.objects.Network;

import java.util.*;

public class Route {

    private final Network network;
    private Graph graph;
    private final ArrayList<String> active_nodes;
    private final int nodes;
    private List<GEdge> edges;

    public Route(Network network) {
        this.network = network;
```

```
            nodes = network.getNodesCount(str -> str.equals("ACTIVE"));
            active_nodes = new ArrayList<>(Arrays.asList(network.getNodesNames(str -> str.equals("ACTIVE"))));

        }

        public boolean checkCurrentRoute(){
            for(int i = 1; i < active_nodes.size(); ++i){
                if(!network.getActiveLinked_Node2().contains(active_nodes.get(i))){
                    return false;
                }
            }
            return true;
        }

        public boolean checkPossibleRoute(){
            ArrayList<String> active_reserve_turn_on_linked_nodes2 = network.getAvailableLinked_Node2();
            for(int i = 1; i < active_nodes.size(); ++i){
                if(!active_reserve_turn_on_linked_nodes2.contains(active_nodes.get(i))){
                    return false;
                }
            }
            return true;
        }

        public void buildRoute(){
            edges = new ArrayList<>();

            for(Edge edge: network.edges){
                if(edge.getStatus().equals("RESERVE") || edge.getStatus().equals("ACTIVE") || edge.getStatus().equals("TURN_ON")){
                    int source = active_nodes.indexOf(edge.getLinked_node_1());
                    int dest = active_nodes.indexOf(edge.getLinked_node_2());
                    int weight = edge.getWeight();
                    edges.add(new GEdge(source, dest, weight));
                }
            }

            graph = new Graph(edges, nodes);
            findShortestPaths(network, graph, 0, nodes, active_nodes);
        }

        public void rebuildRoute(){
            edges = new ArrayList<>();

            for(Edge edge: network.edges){
                if(edge.getStatus().equals("ACTIVE")){
                    int source = active_nodes.indexOf(edge.getLinked_node_1());
                    int dest = active_nodes.indexOf(edge.getLinked_node_2());
                    int weight = edge.getWeight();
```

```
                edges.add(new GEdge(source, dest, weight));
            }
        }

        graph = new Graph(edges, nodes);
        findShortestPaths(network, graph, 0, nodes, active_nodes);
    }

    private static void getRoute(int[] prev, int i, List<Integer> route)
    {
        if (i >= 0)
        {
            getRoute(prev, prev[i], route);
            route.add(i);
        }
    }

    private static void findShortestPaths(Network network, Graph graph, int source, int n,
ArrayList<String> active_nodes) {
        PriorityQueue<GNode> minHeap;
        minHeap = new PriorityQueue<>(Comparator.comparingInt(node -> node.weight));
        minHeap.add(new GNode(source, 0));

        List<Integer> dist;
        dist = new ArrayList<>(Collections.nCopies(n, Integer.MAX_VALUE));

        dist.set(source, 0);

        boolean[] done = new boolean[n];
        done[source] = true;

        int[] prev = new int[n];
        prev[source] = -1;

        while (!minHeap.isEmpty()) {
            GNode node = minHeap.poll();
            int u = node.vertex;

            for (GEdge edge : graph.adjList.get(u)) {
                int v = edge.dest;
                int weight = edge.weight;

                if (!done[v] && (dist.get(u) + weight) < dist.get(v)) {
                    dist.set(v, dist.get(u) + weight);
                    prev[v] = u;
                    minHeap.add(new GNode(v, dist.get(v)));
                }
            }

            done[u] = true;
        }
```

```
            List<Integer> route = new ArrayList<>();
            HashMap<Integer, Integer> target_source = new HashMap<>();

            for (int i = 0; i < n; i++) {
               if (i != source && dist.get(i) != Integer.MAX_VALUE) {
                  getRoute(prev, i, route);
                  for(int j = 0; j < route.size() - 1; ++j){
                     target_source.put(route.get(j + 1), route.get(j));
                  }
                  /*System.out.printf("Path (%s —> %s): Minimum cost = %d, Route = %s\n",
                        active_nodes.get(source), active_nodes.get(i), dist.get(i), route);*/
                  network.getNode(active_nodes.get(i)).setWeight(dist.get(i));
                  route.clear();
               }
            }
            for(int i = 0; i < network.edges.size(); ++i){
               String linked_node1 = network.edges.get(i).getLinked_node_1();
               String linked_node2 = network.edges.get(i).getLinked_node_2();
               int node1_index = active_nodes.indexOf(linked_node1);
               int node2_index = active_nodes.indexOf(linked_node2);
               if(target_source.get(node2_index) != null && target_source.get(node2_index) ==
node1_index){
                  if(network.edges.get(i).getStatus().equals("RESERVE") &&

!network.getActiveLinked_Node2().contains(network.edges.get(i).getLinked_node_2()))
                  {
                     network.edges.get(i).setStatus("TURN_ON");
                  }
               } /*else {
                  if(network.edges.get(i).getStatus().equals("ACTIVE"))
                     network.edges.get(i).setStatus("TURN_OFF");
               }*/
            }
         }
      }
```

Файл Accident.java

```
      package com.company.objects;

      public class Accident {
         private int id;
         private String info;
         private String date;
         private String network;

         public Accident(int id, String info, String date, String network) {
            this.id = id;
            this.info = info;
            this.date = date;
```

```
        this.network = network;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getInfo() {
        return info;
    }

    public void setInfo(String info) {
        this.info = info;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getNetwork() {
        return network;
    }

    public void setNetwork(String network) {
        this.network = network;
    }

    public Object[] getArray(){
        return new Object[]{id, info, date, network};
    }
}
```

Файл Edge.java

```
package com.company.objects;

public class Edge {
    private int id;
    private int weight;
    private String linked_node_1;
    private String linked_node_2;
    private String status;
```

```
public Edge(int id, int weight, String linked_node_1, String linked_node_2, String status) {
    this.id = id;
    this.weight = weight;
    this.linked_node_1 = linked_node_1;
    this.linked_node_2 = linked_node_2;
    this.status = status;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public int getWeight() {
    return weight;
}

public void setWeight(int weight) {
    this.weight = weight;
}

public String getLinked_node_1() {
    return linked_node_1;
}

public void setLinked_node_1(String linked_node_1) {
    this.linked_node_1 = linked_node_1;
}

public String getLinked_node_2() {
    return linked_node_2;
}

public void setLinked_node_2(String linked_node_2) {
    this.linked_node_2 = linked_node_2;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public Object[] getArray(){
    return new Object[]{id, weight, linked_node_1, linked_node_2, status};
}
```

```
}
```

Файл Network.java

```
package com.company.objects;

import java.util.ArrayList;
import java.util.function.Predicate;

public class Network {
    private int id;
    private String name;
    private int edges_count;
    private int nodes_count;
    private String status;

    public ArrayList<Node> nodes = null;
    public ArrayList<Edge> edges = null;

    public Network(int id, String name, int edges_count, int nodes_count, String status) {
        this.id = id;
        this.name = name;
        this.edges_count = edges_count;
        this.nodes_count = nodes_count;
        this.status = status;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getEdges_count() {
        return edges_count;
    }

    public void setEdges_count(int edges_count) {
        this.edges_count = edges_count;
    }
```

```
public int getNodes_count() {
    return nodes_count;
}

public void setNodes_count(int nodes_count) {
    this.nodes_count = nodes_count;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public Object[] getArray(){
    return new Object[]{id, name, edges_count, nodes_count, status};
}

public String[] getNodesNames(Predicate<String> predicate){
    String[] nodes_names = new String[getNodesCount(predicate)];
    int i = 0;
    for(Node node: nodes){
        if(predicate.test(node.getStatus())) {
            nodes_names[i] = node.getName();
            i++;
        }
    }
    return nodes_names;
}

public int getNodesCount(Predicate<String> predicate){
    int count = 0;
    for(Node node : nodes){
        if(predicate.test(node.getStatus()))
            count++;
    }
    return count;
}

public int getNodeIndex(int id){
    int counter = 0;
    for(Node node : nodes){
        if(node.getId() == id)
            return counter;
        counter++;
    }
    return counter;
}
```

```
        public int getEdgeIndex(int id){
            int counter = 0;
            for(Edge edge : edges){
                if(edge.getId() == id)
                    return counter;
                counter++;
            }
            return counter;
        }

        public ArrayList<String> getActiveLinked_Node2(){
            ArrayList<String> linked_nodes2_name = new ArrayList<String>();
            for(Edge edge : edges){
                if(edge.getStatus().equals("ACTIVE")){
                    linked_nodes2_name.add(edge.getLinked_node_2());
                }
            }
            return linked_nodes2_name;
        }

        public ArrayList<String> getAvailableLinked_Node2(){
            ArrayList<String> linked_nodes2_name = new ArrayList<String>();
            for(Edge edge : edges){
                if(edge.getStatus().equals("ACTIVE") || edge.getStatus().equals("RESERVE") ||
edge.getStatus().equals("TURN_ON")){
                    linked_nodes2_name.add(edge.getLinked_node_2());
                }
            }
            return linked_nodes2_name;
        }

        public Edge getEdge(int id){
            for(Edge edge : edges){
                if(edge.getId() == id){
                    return edge;
                }
            }
            return null;
        }

        public Node getNode(int id){
            for(Node node : nodes){
                if(node.getId() == id){
                    return node;
                }
            }
            return null;
        }

        public Node getNode(String name){
            for(Node node : nodes){
                if(node.getName().equals(name)){
```

```
                return node;
            }
        }
        return null;
    }
}
```

Файл Node.java

```
package com.company.objects;

public class Node {
    private int id;
    private String name;
    private int weight;
    private String status;

    public Node(int id, String name, int weight, String status) {
        this.id = id;
        this.name = name;
        this.weight = weight;
        this.status = status;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public String getStatus() {
        return status;
```

```
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public Object[] getArray(){
        return new Object[]{id, name, weight, status};
    }
}
```