НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

004.9-047.36:627.25

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
_Болбот І. В., д. т. н., проф.____	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2024 р.	**____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему _«Експертна система моніторингу наслідків аварій на гідротехнічних спорудах»_____

Спеціальність _121 – Інженерія програмного забезпечення_____

(код і назва)

Освітня програма _Програмне забезпечення інформаційних систем_____

(назва)

Орієнтація освітньої програми _освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_д.т.н., проф._____ _____ _Семко В. В._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____ _____ _Руденський Р. А._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _Юзюк О. В._____
(підпис) (ПІБ студента)

КИЇВ-2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ....4
ВСТУП....5
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ....7
1.1 Опис предметної області....7
1.2 Аналіз наявних рішень....9
1.3 Постановка завдання....10
2 МОДЕЛЮВАННЯ СИСТЕМИ....12
2.1 Діаграма прецедентів....12
2.2 Use-Case та User Story....14
2.3 Діаграми діяльності та послідовності....17
2.4 Оперативна БД....20
3 РОЗРОБКА СИСТЕМИ....22
3.1 Архітектура системи....22
3.2 Використані технології....23
3.3 Алгоритми обробки інформації....27
4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ....30
4.1 Проектування СД та розгортання OLAP-куба....30
4.2 Побудова звітності та розрахунок KPI....34
4.3 Оцінка стану споруд в реальному часі....39
ВИСНОВКИ....45
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ....47
ДОДАТОК А....52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ГЕС – гідроелектростанція.

ВБР – Водонапірна Башта Рожновського.

БД – База Даних.

СД – Сховище Даних.

UML – Unified Modeling Language.

US Code – Use-Case Code.

СУБД – Система Управління Базами Даних.

SSAS – SQL Server Analysis Services.

SSIS – SQL Server Integration Services.

MDB – Material-Design Bootstrap.

BI – Business Intelligence.

HTML – HyperText Markup Language.

CSS – Cascade Style Sheets.

IDE – Integrated Development Environment.

OLAP – On-Line Analytical Processing.

KPI – Key Performance Indicator.

KDE – Kernel Density Estimation.

ВСТУП

Актуальність. Важливість води для людей важко переоцінити: без неї на Землі не зародилося б життя, яке ми знаємо зараз, можливо, навіть взагалі його тут ніколи і не було би, без води це життя просто не може існувати.

Якщо абстрагуватися від більш фундаментальних аспектів важливості води і звернути увагу на практичні аспекти, то без води не можна було би створювати низку матеріалів, наприклад, бетон, а також не можна було би використовувати силу скупчень води для генерації енергії за допомогою ГЕС.

Разом із такими можливостями з’являються і ризики: якщо сила води подавить греблю ГЕС, це може привести до серйозної катастрофи. Саме тому вкрай необхідно мати можливість визначити різні сценарії катастроф, щоб можна було завчасно підготувати як спеціалізовану рятувальну і ремонтну техніку та персонал, так і населення, яке проживає в зоні ризику.

Однак лише визначення сценаріїв катастроф недостатньо, необхідно ще й кількісно проаналізувати їх наслідки, щоб можна було кількісно оцінити, скільки потрібно буде засобів задля усунення цих наслідків, а також в реальному часі стежити за станом гідротехнічних споруд, щоб оперативно реагувати на загрозу ще до її настання.

Об’єкт дослідження: система моніторингу наслідків аварій на гідротехнічних спорудах, яка дозволяє не тільки розробити симуляції аварій, але й надає аналітичні відомості про їх наслідки та стежить за станом гідроспоруд.

Предмет дослідження: методи і технології розробки експертних систем для моніторингу, симуляції наслідків аварій та аналізу ризиків на гідротехнічних спорудах.

Мета дослідження: підвищення якості експертної системи моніторингу стану гідротехнічних споруд у реальному часі на основі поточної інформації та системи знань.

Методи дослідження. Задля реалізації моніторингової складової експертної системи були використані мова програмування Python та

непараметрична модель KDE з бібліотеки scipy. Для проведення аналізу даних та розрахунку ключових показників ефективності використовувалася технологія OLAP.

Наукова новизна. Було досліджено можливість застосування KDE задля визначення потенційної загрози на основі поточного стану гідротехнічних споруд, враховуючи вже наявну інформацію про симуляції аварій на них.

Апробація роботи. Під час розробки цієї експертної системи було прийнято участь у двох конференціях: XIV та XV МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ МОЛОДИХ ВЧЕНИХ «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА». На обох конференціях були представлені основні положення у вигляді тез.

Структура роботи. Магістерська робота складається зі вступу, чотирьох розділів (системного аналізу предметної області, моделювання системи, розробки системи та результатів дослідження), висновків, переліку використаних джерел та додатків.

Записка складається з 51-єї сторінки та одного додатку. У роботі використано 47 джерел.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Експертна система моніторингу наслідків аварій на гідротехнічних спорудах є наступною ланкою розвитку програмного симулятора наслідків аварій на гідротехнічних спорудах. Тому перш ніж описувати експертну систему, спочатку варто описати програмний симулятор.

Програмний симулятор був розроблений з метою розв’язання задачі визначення наслідків аварій на гідротехнічних спорудах різного масштабу (від ГЕС до відносно невеликих ВБР) та з різними параметрами. Програмний симулятор також реалізовує можливість графічного представлення наслідків цих аварій з різними параметрами на обраних гідротехнічних спорудах у вигляді інтерактивної мапи.

З технічної точки зору, це було реалізовано за допомогою створення спеціального веб-орієнтованого додатку, який надає наступні можливості:

- заносити в БД нові споруди разом з їх параметрами;
- запускати симуляції різноманітних варіантів аварій;
- переглядати та зберігати результати цих симуляцій;
- переглядати інформацію про збережені споруди та приналежні до них симуляції;
- керувати фундаментальною для системи інформацією із окремої панелі адміністратора (дані про користувачів, типи даних, карта висот тощо)

Цільовою аудиторією програмного симулятора є люди, які зацікавлені в оцінці наслідків таких аварій. Ними, в першу чергу, можуть бути оператори гідротехнічних споруд. Також такою системою можуть бути зацікавлені на державному рівні для власної оцінки кількості постраждалих під час таких катастроф або для інформування населення.

При формуванні вимог до програмного симулятора було визначено

декілька необов’язкових вимог, одна із найголовніших – це передача сигналів про аварії від споруд до локальних систем оповіщення. Вона сформульована на основі того, що, по-перше, усі сучасні великі гідротехнічні споруди мають власні сервера, а по-друге, на ринку присутні компанії, які надають послуги зі встановлення та налаштування локальних систем оповіщення [1], а тому немає необхідності розроблювати власну, достатньо виступити посередником між спорудами і системами оповіщення. Однак це не було реалізовано через наступний рад проблем:

- це не стосується відносно невеликих споруд, у яких може навіть не бути персоналу, що вже казати про сервери, наприклад, ті ж ВБР, а тому вони для реалізації такої можливості потребують встановлення додаткових датчиків, які і будуть сповіщати про загрозу;
- може бути складно поєднати сервер споруди і системи оповіщення, адже вони можуть не мати необхідних можливостей для такого обміну повідомленнями, а тому це може потребувати деякого часу і навіть коштів;
- може виявитися, що на державному рівні такий зв’язок із таким рівнем автоматизації вже повністю передбачено, а тому у цьому функціоналі немає необхідності.

Якщо з третім пунктом в рамках програмного симулятора нічого не можна зробити, то перший та другий цілком реально спростити, чи навіть зовсім вирішити, для цього і покликана наступна ітерація програмного симулятора – вже експертна система.

За визначенням, експертна система – це інтелектуальна система, призначена для вирішення слабо формалізованих задач, на основі накопиченого в базі знань досвіду роботи експертів в проблемній області. В даній ситуації у якості бази знань виступають результати симуляцій, які були оцінені кваліфікованими людьми, а задачею є як раз вирішення перших двох перепон, про які було сказано вище.

З технічної точки зору, це має бути вирішено наступним чином: замість того, щоб споруди самостійно сповіщали про аварію, ця експертна система має

періодично робити запити на сервери чи датчики споруд і на основі отриманої інформації, робити оцінку поточного стану споруди. Оцінка має заключатися не у проведенні нової симуляції, бо це може зайняти багато часу, вона повинна ґрунтуватися на основі раніше проведених та оцінених фахівцями симуляцій. Таким чином, інтеграція невеликих гідротехнічних споруд стає цілком реальною задачею, а їх поєднання із системами оповіщення набагато простішим.

1.2 Аналіз наявних рішень

Як вже було зазначено у вступі, попередній пошук в інформаційно-пошуковій системі Google показав, що у відкритому чи напіввідкритому доступі доволі легко знайти лише ті системи, які досліджують наслідки катастроф природного характеру, що і виступає їх головним недоліком, хоч і не робить їх безкорисними. Проте серед них все ж таки є система, яка могла б задовільнити потребу в аналізі наслідків аварій на гідротехнічних спорудах – це InfraWorks. Вона не спеціалізується на симуляціях, вона призначена для візуалізації різноманітних архітектурних концептів [2]. Тим не менш, в ній є можливість підключення спеціальних розширень – плагінів. Враховуючи, що ця програма – це продукт компанії Autodesk, яка є доволі популярною, можна сміливо припускати, що існує плагін, який дозволить симулювати аварії на гідротехнічних спорудах.

Важко казати про переваги чи недоліки програми, про існування якої достеменно невідомо. Однак, скоріш за все, це буде якісний і доволі продуктивний продукт, як і більшість плагінів для рішень Autodesk. А ще йому буде характерний один суттєвий недолік, як і більшості плагінів – ціна. Враховуючи ціни на той же InfraWorks, можна припустити, що необхідно буде заплатити додаткових 20-30% на плагін.

Для пошуку аналогів було використано також різноманітні портали та бази патентів. Так, пошук в порталі Національної бібліотеки України імені В. Вернадського [3] та вітчизняній базі патентів [4] не дав результатів, однак

пошук в європейській [5] та американській [6] базах патентів не був безрезультатним.

В європейській базі вдалося знайти багато релевантних патентів, серед яких можна виділити метод симуляції прориву дамби на основі штучного інтелекту [7], метод аналізу ризиків від прориву дамби [8], а також ще один метод симуляції прориву дамби, який містить в собі і систему раннього оповіщення [9]. Всі отримані результати є вузькоспеціалізованими, бо стосуються лише прориву дамб, загальний пошук, який включав би будь-які гідротехнічні споруди, результатів не дав.

В американській базі ситуація аналогічна: загальний пошук не дав результатів, а серед вузьконаправленого можна виділити метод оцінки глибини затоплення [10], а також систему контролю подачі води, яка симулюючи результати команд, запобігає затопленню [11].

Зі знайдених результатів стає зрозуміло, що комплексні підходи потребують поєднання декількох методів у єдину злагоджену систему, яка не може бути універсальною, а тому розроблюється з врахуванням унікальних обставин.

1.3 Постановка завдання

Головною задачею є створення моніторингових та аналітичних засобів, за допомогою яких можна було б провести аналіз даних в системі і надати відповіді на наступні питання:

- яка кількість населення постраждала внаслідок аварії у конкретний час після початку аварії?
- яка кількість систем оповіщення була задіяна для попередження населення про аварію у конкретний час після початку аварії?
- яка кількість пам’яток культурної спадщини постраждала внаслідок аварії у конкретний час після початку аварії?
- яку загрозу становить поточний стан гідротехнічної споруди?

Задля цього у системи повинні бути присутні наступні дані:

- інформацію про споруди, а конкретно, їх назву та географічне положення;
- інформацію про рівні загроз, також їх назву;
- інформацію про населені пункти, кількість їх населення;
- інформацію про культурні пам’ятки, їх кількість;
- інформацію про системи оповіщення, також їх кількість;
- інформацію про симуляції, а саме які їх рівні загрози та яким спорудам вони належать;
- інформацію про те, які населені пункти, пам’ятки і системи оповіщення постраждали чи були задіяні на конкретний час після початку симуляції.

Звісно, це необхідна та достатня інформація лише для проведення аналізу, її буде недостатньо для правильного функціонування системи, але уся інша інформація не є у фокусі даної магістерської кваліфікаційної роботи.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Після того, як було визначено конкретний набір кроків, тобто задач, які необхідно виконати для досягнення поставленої мети, варто зобразити наявні в системі функції, а також їх взаємодію у вигляді діаграм. Це потрібно для кращого розуміння того, які саме дані наявні у системі, як їх можна використати для аналізу і чого, можливо, не вистачає для того, щоб відповісти на питання, задані в постановці завдання. Для цього було вирішено обрати одну із поведінкових діаграм стандарту UML – діаграму прецедентів (use-case diagram) [12].

На цій діаграмі зображені окремі системи, підсистеми чи модулі, які називаються суб’єктами, користувачі, які називаються акторами, та доступні для них функції системи, які називаються прецедентами [13]. Актори і прецеденти пов’язуються за допомогою зв’язків чи асоціацій. Суб’єкти позначаються прямокутниками, всередині яких розміщуються прецеденти у вигляді овалів. Ззовні суб’єкта розміщуються актори, зазвичай, у вигляді чоловічків, хоча вони можуть мати будь-яке відображення, навіть у вигляді класу із діаграми класів UML [14]. Для даної системи ця діаграма показана на рис. 1.

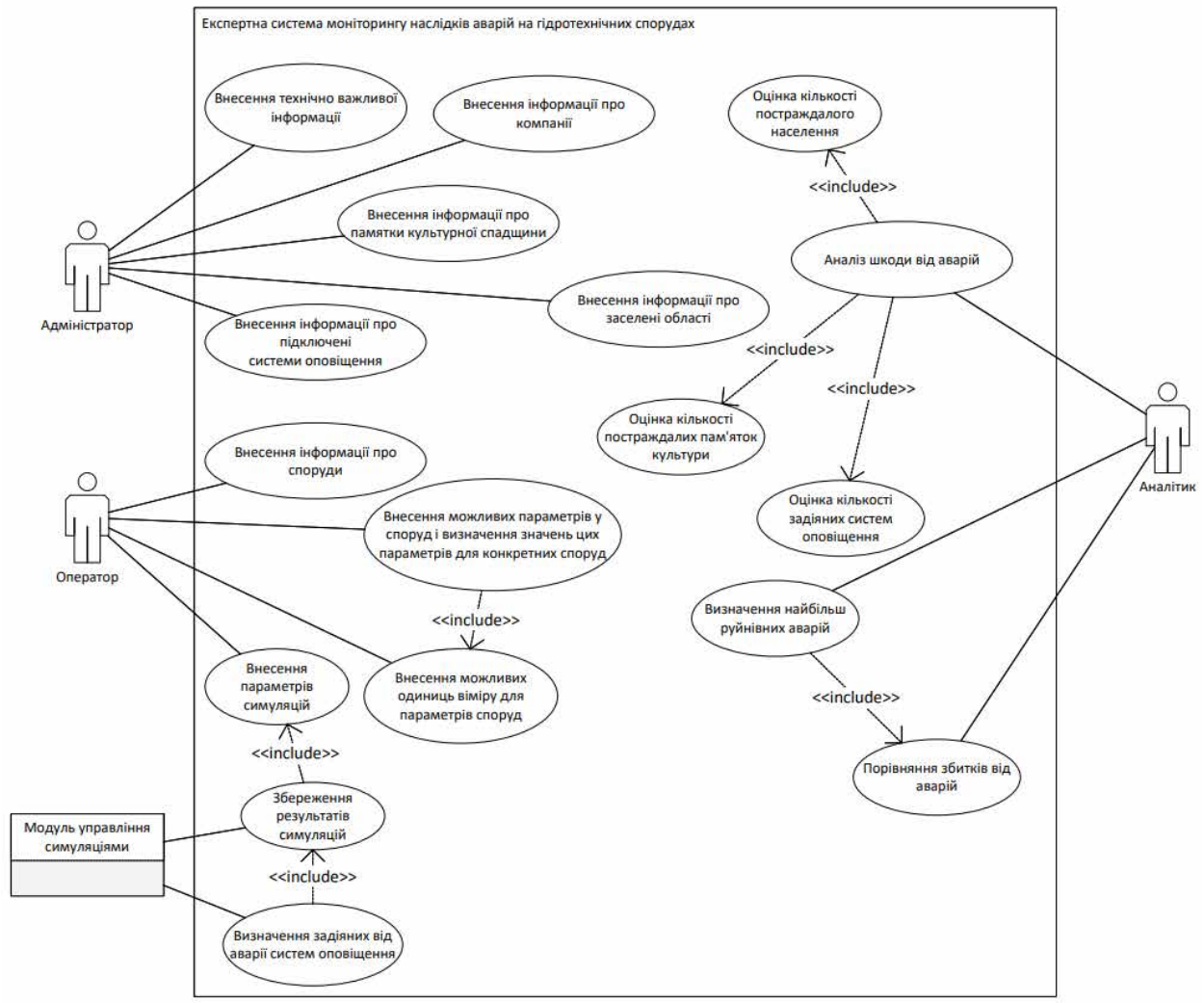


Рис. 1 Діаграма прецедентів

На даній діаграмі прецедентів зображені наступні актори:

- адміністратор – це людина, яка має певні навички програмування, а тому займається внесенням більше службової інформації, яка необхідна для функціонування системи;
- оператор – це людина, яка має спеціалізовані знання про гідротехнічні споруди і їх параметри і яка використовує систему за її безпосереднім призначенням – задля проведення симуляцій;
- модуль управління симуляціями – це клас, який відповідає за технічне керування інформацією про симуляції: їх проведення, збереження і визначення наслідків;

- аналітик – це людина, яка на основі усієї інформації, яку внесли інші користувачі системи, робить певні висновки і відповідає на ті питання, які були задані у постановці завдання.

В контексті цієї магістерської роботи головна увага буде спрямована здебільшого на аналітика, на створенні та налаштуванні засобів, які йому необхідні для проведення аналітичної роботи і, як наслідок, надання відповідей на питання із постановки завдання.

2.2 Use-Case та User Story

Маючи вимоги до системи, недостатньо просто сказати, що вони потрібні якомусь користувачу і все, необхідно ще визначити, для чого це треба, яку конкретну проблему чи побажання користувача вирішує дана вимога. Для цього є User story (історія користувача). User story – це короткий опис бажаної функціональності системи, написаний з точки зору користувача. Вони використовуються для збору та документування вимог до системи, а тому є цінним інструментом для забезпечення того, щоб система відповідала потребам користувачів.[15].

Нижче наведено табл. 1 із прикладом User story. Звісно, щоб повноцінно описати усю систему, потрібно набагато більше, але для демонстрації цього буде достатньо.

Таблиця 1

US Code	5
As a	Адміністратор
I want	Мати можливість повноцінно керувати системою із графічного інтерфейсу
In order to	Для управління системою
	Acceptance Criteria
Working **When** **Then**	Маючи облікові дані Виконав вхід у окремій формі логіну Отримав доступ до панелі адміністратора

Для написання User story у якості критеріїв якості було використано акронім INVEST, який розшифровується наступним чином:

- I – Independent (Незалежна): Кожна user story повинна бути незалежною від інших user story, тобто її можна реалізувати в будь-якому порядку. Це робить user story модульними та легкими для керування.
- N – Negotiable (Обговорювана): User story – це лише опис бажання користувача, а не детальна специфікація. Деталі реалізації повинні обговорюватися між користувачами, розробниками та іншими зацікавленими сторонами.
- V – Valuable (Цінна): User story повинна описувати цінність для користувача. Це означає, що вона повинна вирішувати реальну проблему або задовольняти реальну потребу.
- E – Estimable (Оцінювана): User story повинна бути досить детальною, щоб можна було оцінити зусилля, необхідні для її реалізації. Це дозволяє командам планувати та відстежувати хід роботи.
- S – Sized (Невелика): User story повинна бути невеликою та чітко

визначеною. Це робить їх легшими для розуміння, розробки та тестування.

- T – Testable (Перевірювана): User story повинна бути перевірюваною, тобто повинні бути чіткі критерії визначення того, чи виконана вона. Це допомагає гарантувати, що user story реалізована правильно [16].

В табл. 2 наведено відповідність вищезазначених User story критерію INVEST.

Таблиця 2

I	N	V	E	S	T
●	●	●	●	●	●

Як можна побачити, User story для даної експертної системи майже повністю відповідають цьому критерію.

User story – це лише загальний опис функціональності, який підходить для спілкування між менеджерами та замовниками програмних продуктів. Однак для розробників цих продуктів потрібні набагато конкретніші описи, із покроковим зазначенням дій користувача у системі. Для цього використовуються **Use-case** (сценарії використання).

Use-case також є інструментом аналізу та опису функціональності системи з точки зору її користувачів. Однак він покроково описує, як користувачі взаємодіють з системою для досягнення певної мети. Основна мета сценарію використання полягає в описі послідовності подій, що відбуваються між користувачем і системою, від початку до досягнення бажаного результату. Сценарії використання допомагають розуміти, як система буде використовуватися в реальних умовах і дозволяють здійснювати аналіз вимог до програмного забезпечення. [17].

Як і для User story, для даної експертної системи було розроблено один сценарій використання, хоча, звісно, для повноцінного її опису потрібно набагато більше. Він зазначений у таблиці нижче.

Use Case Code	3
Use Case Name	Запуск симуляції
Trigger	Перейдено на сторінку споруди
PreConditions	• Виникла необхідність просимулювати новий сценарій аварії
Normal Scenario	1. Перейдено на сторінку запуску симуляції 2. Визначено стартову точку симуляції на мапі 3. Обрано параметри споруди серед доступних та введено їх значення 4. Запущено симуляцію 5. Результати симуляції збережено
Result	Нова симуляція з’явилися у переліку симуляцій даної споруди

2.3 Діаграми діяльності та послідовності

Після діаграми прецедентів можна зобразити послідовність виконання дій під час загального функціонування системи за допомогою ще одної поведінкової діаграми UML – діаграми діяльності. На ній у вигляді прямокутників зображені певні дії, часто, прецеденти, які називаються активностями. Активності містять в собі об’єкти, часто, це актори, а також відповідні ним потоки відповідальності, які називаються розділами або доріжками, бо вони нагадують доріжки для плавання. Всередині доріжок розміщуються атомарні кроки в ході виконання якоїсь активності, які називаються діями і позначаються прямокутниками з округлими кутами, а також інші допоміжні компоненти: розгалуження у вигляді ромбів, граничні події у вигляді кругів тощо [18]. Для даної системи діаграма діяльності зображена на рис. 2.

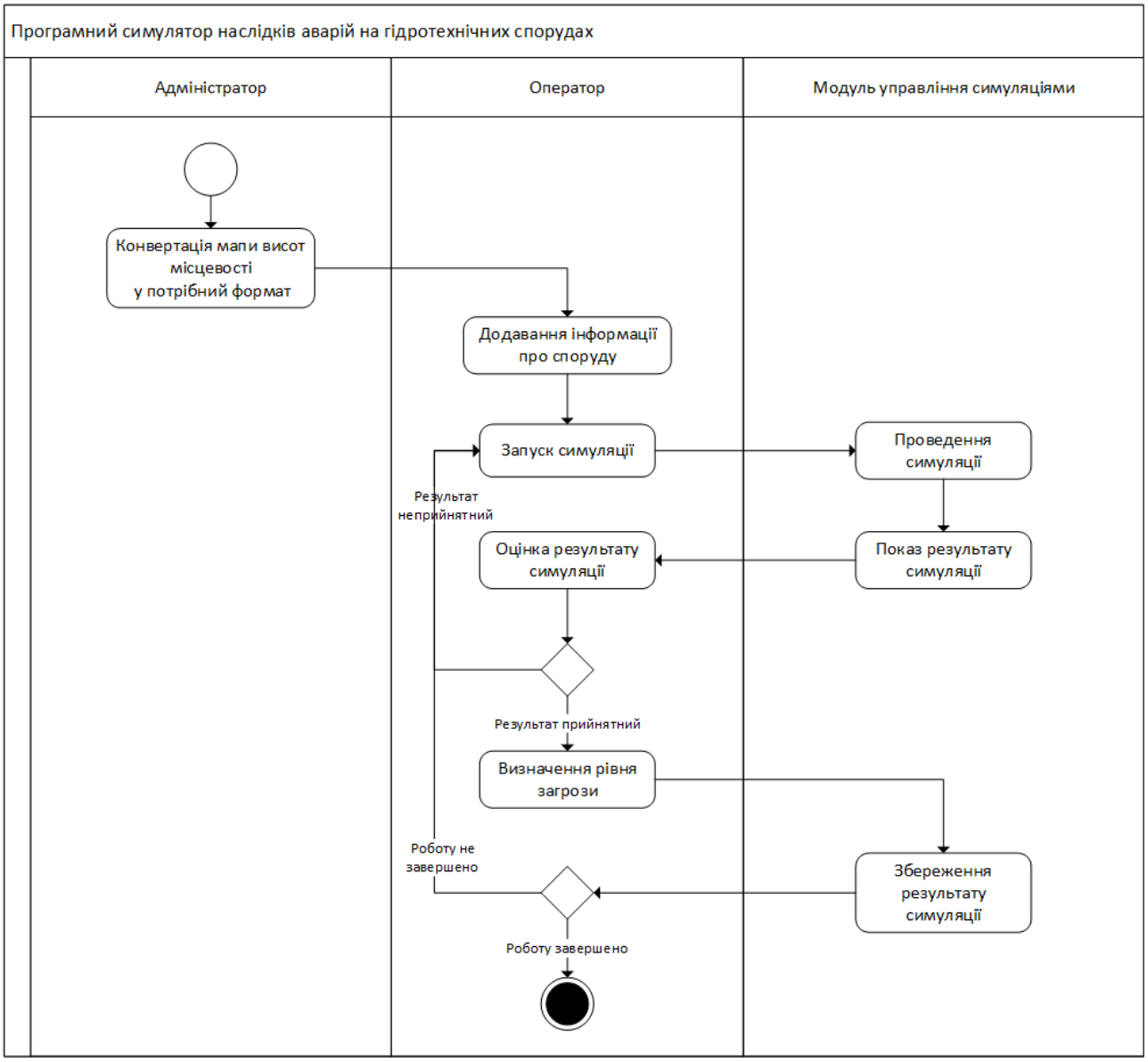


Рис. 2 Діаграма діяльності

Як можна побачити, на ній зображено загальний процес того, як працює система. Спочатку адміністратор підготовлює мапу висот, адже для цього необхідні певні навички програмування. Потім оператор додає інформацію про споруду і запускає симуляцію. Коли модуль управління симуляціями виконає цю симуляцію і покаже її результат, оператору необхідно вирішити, чи прийнятним є результат. Можливо, під час симуляції виникли якісь помилки, і варто спробувати знову, може, оператор проводить експеримент і намагається краще визначити, як розмежовувати симуляції тощо. Якщо результат не прийнятний, то симуляція запускається знову, а якщо прийнятний, то визначається рівень загрози, результати зберігаються і робота завершується. Очевидно, що реальна

робота системи буде відрізнятися від цієї моделі, адже, наприклад, оператор може банально закрити вкладку браузера. Такі нюанси не зображені на діаграмі, щоб зайвий раз не ускладнювати її.

Отримавши діаграму прецедентів та активності, вже можна зрозуміти, що система має робити і в якому порядку. Але якщо придивитися, то для повної картини опису системи все ще не вистачає однієї складової – опису того, як саме взаємодіють об’єкти. Для цього в UML призначена одна із діаграм взаємодії – діаграма послідовності. На ній зображені окремі системи чи прецеденти, які називаються обмеженнями. Як і на інших діаграмах, на діаграмі послідовності зображені об’єкти, але тепер у них є лінії життя – лінії, які представляють сутності, які приймають участь в процесі. На лініях життя схематично позначається процес роботи об’єктів у вигляді прямокутника, а між цими прямокутниками позначається саме те, для чого ця діаграма і робиться – повідомлення. Для додаткової деталізації процесу в діаграмі діяльності можуть бути присутні різні фрагменти, позначені аналогічно системі, але з іншою назвою: альтернативи, цикли тощо [19]. Для даної системи діаграма послідовності зображена на рис. 3.

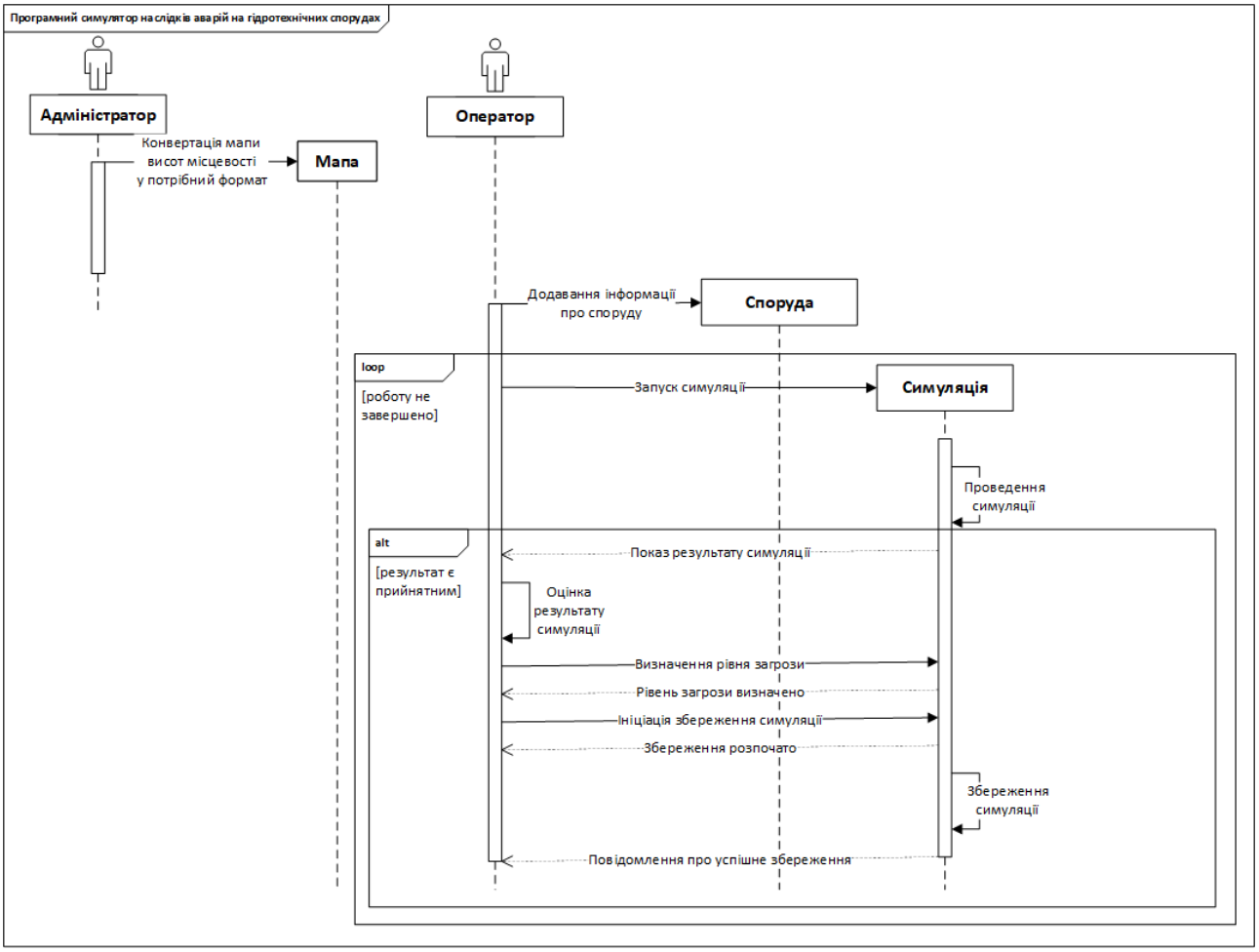


Рис. 3 Діаграма послідовності

Як можна побачити, на ній знову ж таки зображено загальний процес того, як працює система, однак цього разу ще й видно, що і коли буде задіяно і задля чого.

2.4 Оперативна БД

Дана експертна система моніторингу наслідків аварій розроблюється на базі програмного симулятора наслідків аварій на гідротехнічних спорудах, а тому вона не потребує великої кількості додаткової інформації, лише інформацію про конкретні об’єкти, які можуть постраждати в результаті аварії, і їх параметри, а саме населені пункти з відомим розташуванням і населенням і розташування культурних пам’яток. БД основної системи, а також таблиці, які містять додаткову інформацію, разом формують так звану оперативну БД, схему якої можна побачити в додатку А.

На цій схемі можна побачити, що центральною сутністю є Simulations – конкретні симуляції. Кожна із симуляцій належить конкретній споруді, які репрезентовані сутністю Constructions разом із такими допоміжними сутностями, як ConstructionParams та ParamsOfConstruction, які репрезентують усі можливі параметри споруд і значення конкретних параметрів у конкретної споруди відповідно, а також деякими іншими. Кожній симуляції оператором призначається рівень загрози, які репрезентовані сутністю DangerLevels. Окрім того, у кожної симуляції є перелік своїх параметрів, ParamsOfSimulation, які відповідають переліку параметрів споруди, якій належить ця симуляція, але мають інші значення.

Для відповіді на задані в постановці завдання питання в оперативній БД є сутності NotificationSystems (системи оповіщення), HabitableAreas (населені пункти) та CulturalMonuments (культурні пам’ятки). Вони пов’язуються із конкретною симуляцією за допомогою таблиць AffectedSystems (для систем оповіщення), AffectedHabitableAreas (для населених пунктів) і AffectedCulturalMonuments (для культурних пам’яток), а також населені пункти та культурні пам’ятки пов’язані із допоміжними таблицями, які містять їх типи.

Окремо варто виділити сутність SimulationTimestamps, яка репрезентує конкретну точку часу після початку аварії. Це потрібно для того, щоб відслідковувати прогрес аварії, тобто оцінювати та порівнювати шкоду від аварії станом на різні моменти часу відносно початку.

3 РОЗРОБКА СИСТЕМИ

3.1 Архітектура системи

Для того, щоб краще розуміти, що саме потрібно зробити для того, щоб системою могли користуватися усі користувачі, необхідно оцінити, як ця система має функціонувати в уже розгорнутому вигляді, тобто, що і на яких серверах буде знаходитися, і які з користувачів будуть взаємодіяти з якими частинами системи. Це можна зробити за допомогою діаграми топології системи, на якій зображується потік даних між користувачами і вузлами системи [20]. Для даної системи діаграма топології показана на рис. 4.

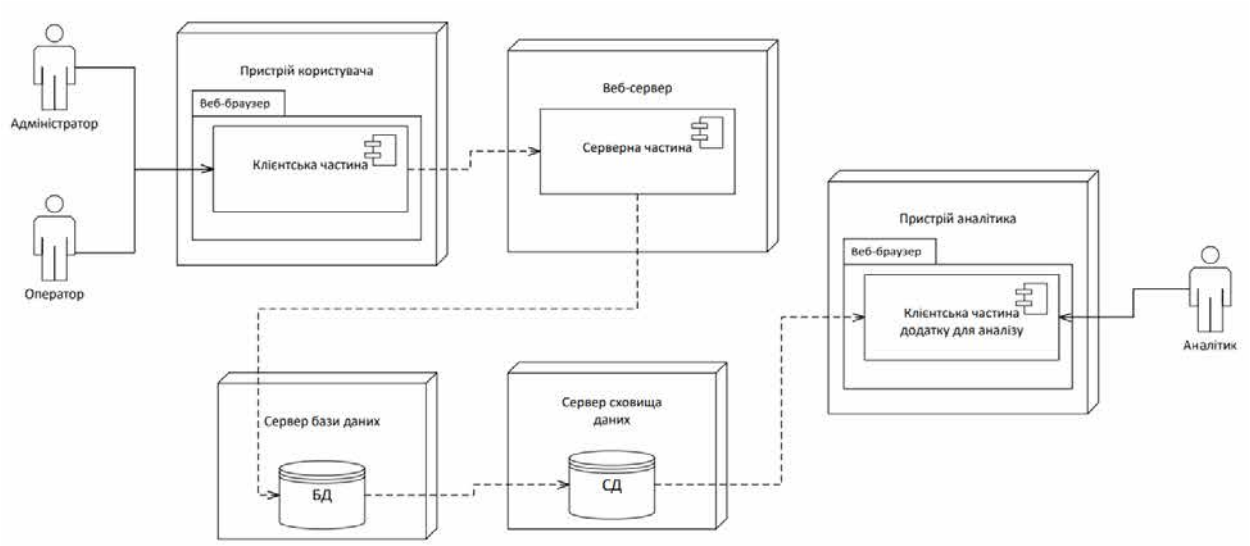


Рис. 4 Топологія системи

На ній присутні усі актори із діаграми прецедентів, окрім модуля управління симуляціями, адже це все-таки частина самої системи, хоч він і був виділений окремо в діаграмі прецедентів задля підкреслення того, як саме дані про симуляцію зберігаються і попередньо оброблюються.

Також на ній зображені усі вузли, які приймають участь в процесі передачі даних між користувачами, а також конкретні компоненти чи об’єкти, які функціонують на цих вузлах. Варто зазначити, що веб-сервер та сервер БД та СД належать системі, хоча і розміщені окремо. Це необхідно для покращення

продуктивності окремих компонентів, а також ізоляції їх допоміжної інформації, яка накопичується протягом функціонування системи, наприклад, бекапи БД.

Для аналітика тут є важливим СД, адже воно займається агрегацією інформації із БД, при чому, лише необхідну для аналізу, що дозволяє аналітику за допомогою окремого користувацького інтерфейсу маніпулювати графічним зображенням цих даних і, власне, проводити аналіз. Детальніше про СД буде сказано у наступному підрозділі.

3.2 Використані технології

Дану експертну систему було розроблено на мові програмування Python. Python – це високорівнева інтерпретована мова програмування загального призначення із динамічною типізацією і «збирачем сміття». Вона підтримує різні парадигми програмування: структурне, об’єктно-орієнтоване та функціональне. Вона містить доволі обширну стандартну бібліотеку, а також надзвичайно велику кількість бібліотек open-source спільноти, адже сама мова програмування має відкритий вихідний код [21].

Так як дана система є веб-орієнтованою, для полегшення та пришвидшення розробки доцільно використовувати веб-фреймворк. В даному випадку було використано Django. Як зазначають самі його розробники, це веб-фреймворк для перфекціоністів із дедлайнами. Його перевагами є те, що він поставляється із величезною кількістю функцій, наприклад, власна ORM-система для роботи з БД, гнучка панель адміністратора, робота із формами, підтримка кешування запитів і багато чого іншого [22]. Високорівнева архітектура Django складається із проектів і додатків [23]. Проект – це вся розроблювана система разом, а додатки – це окремі самодостатні частини проекту, які розроблюються таким чином, щоб їх можна було перевикористовувати. Завдяки цьому існує велика кількість різних бібліотек-розширень для Django, які також використовувалися під час розробки.

Серед них можна виділити:

- django-phonenumbers, яка призначена для зручної роботи із номерами телефонів [24];
- django-phonenumber-field, яка призначена для зручного відображення поля із номером телефона [25];
- django-leaflet, яка призначена для зручної інтеграції JavaScript-бібліотеки для мапи Leaflet [26];
- django-slick-reporting, яка призначена для формування звітів на основі вже існуючих моделей і даних, асоційованих із ними [27].

Однією із головних бібліотек, які були використані для розробки, окрім Django, є ANUGA. Вона призначена для моделювання наслідків гідрологічних катастроф, наприклад, проривів дамб, розливів річок, цунамі тощо. Вона розроблена Національним Австралійським Університетом та державним агентством Австралії Geoscience Australia, звідки і походить назва. Для моделювання ANUGA використовує рівняння мілкої води на неструктурованій триангулярній сітці, який розв’язується за допомогою метода кінцевого об’єму. Як би дивно це не здавалося, але рівняння мілкої води підходять для вирішення досить великої кількості задач, а тому ANUGA ідеально підходить для розроблюваної інформаційної системи [28].

У якості СКБД система використовує PostgreSQL. Вона є об’єктно-реляційною, що поєднує в собі як звичайні характеристики реляційних БД, так і можливості створення класів та їх наслідування. Завдяки цьому можна створювати власні типи даних [29].

Одними із таких типів є типи для спрощення роботи із, переважно, географічними даними: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection, різні типи для растрових зображень тощо. Всі вони та багато інших входять до розширення PostGIS [30].

Python за замовчуванням не має підтримки підключення до БД PostgreSQL, для цього необхідна спеціальна бібліотека-адаптер. Їх існує велика кількість, однак серед них було обрано psycopg, адже вона підтримує велику кількість розширень PostgreSQL, включаючи необхідне для збереження геоданих

PostGIS [31].

Так як дана система є веб-орієнтованою, її графічними інтерфейсами є веб-сторінки. Найчастіше, вони розроблюються за допомогою HTML, стилізуються за допомогою CSS чи її бібліотек, наприклад, MDB, і робляться інтерактивними за допомогою мови програмування JavaScript. В Django інтерфейси називаються шаблонами, для їх компоновки і відображення веб-фреймворк поставляється із власною системою шаблонів [32].

Python дозволяє створювати спеціальні віртуальні середовища виконання. Це потрібно для того, щоб не засмічувати бібліотеками головний інтерпретатор, а також щоб розділяти бібліотеки різних проектів, які виконуються в одній ОС. Під час розробки даної інформаційної системи також було використане таке середовище і платформу – Anaconda. Вона містить в собі велику кількість бібліотек, переважно, наукового призначення, в тому числі і ANUGA [33].

Розробка системи велася у IDE PyCharm – середовищі, спеціалізованому, переважно, для мови програмування Python [34], а впровадження системи зроблене з використанням Docker – спеціальної платформи для віртуалізації на рівні ОС за допомогою створення так званих контейнерів [35].

Під час реалізації системи було також використано OLAP – технологію, що використовується для аналізу великих обсягів даних. OLAP-системи дозволяють користувачам швидко отримувати підсумкові дані з різних точок зору, а також візуалізувати ці дані за допомогою діаграм, графіків та інших графічних елементів. OLAP-системи використовують багатовимірну модель даних, яка дозволяє зберігати дані в структурі, що відповідає їхньому природному змісту [36].

Для застосування OLAP потрібні, власне, дані, а тому вони часто використовуються спеціальних репозиторіїв даних – сховищ даних. СД – це специфічний тип системи управління даними, яка призначена для уможливлення та підтримки бізнес-аналітики. СД призначені виключно для виконання запитів та аналізу і часто містять великі обсяги історичних даних. Дані в СД зазвичай отримують з широкого кола джерел, таких як файли журналів додатків і

транзакційні додатки.

Добре спроектоване СД виконує запити дуже швидко, забезпечує високу пропускну здатність даних і надає кінцевим користувачам достатню гнучкість для "нарізки" або зменшення обсягу даних для більш детального вивчення [37].

Центральна частина OLAP – це так званий куб – багатовимірна (не обов'язково саме трьохвимірна, як зазначає назва «куб») структура даних, яка використовується для швидкого та ефективного аналізу великих наборів даних. Куби створюються на основі якогось джерела даних, яким, найчастіше, виступає саме СД [38].

SSAS – це потужний інструмент у складі пакету Microsoft SQL Server, який надає дві основні функціональні можливості:

- OLAP: дозволяє аналізувати великі обсяги даних з різних джерел, організовуючи їх у OLAP-куби;
- інтелектуальний аналіз даних: SSAS також може похвалитися можливостями інтелектуального аналізу даних, що дозволяє отримувати приховану інформацію і знання з даних. Це включає в себе такі методи, як аналіз асоціацій, кластеризація і дерева рішень.

SSAS є основою багатьох BI-рішень, надаючи користувачам можливість досліджувати та аналізувати дані для прийняття обґрунтованих рішень [39].

SSIS – це ще один інструмент від Microsoft, який допомагає переміщувати та перетворювати дані між різними джерелами, тобто він з'єднує різні інформаційні сховища і робить їх придатними для аналізу та звітності.

Основна функція SSIS – це інтеграція даних: він чудово справляється з процесами ETL (Extract, Transform, Load). Він може збирати дані з різних джерел, таких як бази даних, файли, веб-сервіси і навіть соціальні мережі. Потім він очищає, форматує і впорядковує дані, перш ніж завантажити їх у місце призначення, наприклад, у СД або іншу БД.

Аналізувати агреговані дані із заповненого СД можна за допомогою різних методів. Для виконання даної магістерської роботи було вирішено обрати середовище BI.

Microsoft Power BI – це набір інструментів бізнес-аналітики, які використовуються для аналізу даних і обміну висновками. Це потужний набір інструментів, який дозволяє перетворити дані, в тому числі і великі масиви даних, на зрозумілі й дієві історії [40].

3.3 Алгоритми обробки інформації

3.3.1 KPI. KPI (ключові показники ефективності) – це спеціальні метрики, які застосовуються для того, щоб оцінити та виміряти успіх усієї організації або конкретних співробітників у досягненні стратегічних та оперативних цілей бізнесу. Кожний ключовий показник ефективності дозволяє визначити, наскільки ефективно компанія або окремий співробітник виконує поставлені завдання, і служить інструментом для прийняття управлінських рішень [41].

Серед прикладів KPI можна виразити:

- чистий прибуток;
- рівень конверсії;
- продуктивність праці;
- відсоток браку тощо.

В різних випадках, на різних підприємствах ці показники розраховуються по-різному, тому універсальної формули не існує. Тим не менш, знаючи, що потрібно розрахувати в якомусь конкретному випадку і оцінивши наявні дані, можна самостійно розробити формулу, і потім розрахувати показники. Так, для даної системи можна виділити два ключових показники:

- густота населення в регіоні, який потерпає від катастрофи. Він розраховується дуже просто: загальна кількість постраждалих внаслідок аварії ділиться на кількість задіяних систем оповіщення. Така формула надає змогу попередньо оцінити, наскільки концентровано повинні бути розташовані сили та засоби служб надзвичайних ситуацій: чим більший показник, тим менша за площею постраждала область, але більша концентрація населення в ній. Зі

зменшенням цього показника, ситуація зворотна: менша концентрація населення, але на більший площі, тому варто подумати над визначенням найоптимальнішого розташування служб;

- густота пам’яток архітектури в регіоні, який потерпає від катастрофи. Цей показник розраховується так само, лише замість кількості населення береться кількість пам’яток архітектури. Цей показник не покаже концентрацію населення, однак продемонструє наскільки важливим є якийсь регіон з культурної точки зору та може свідчити про підвищену кількість грошових збитків, які буде отримано внаслідок аварії, адже в них може бути включено вартість реконструкції пам’яток, недоотримані кошти через тимчасову зупинку туризму або взагалі невідворотно втрачений прибуток через знищення пам’ятки.

3.3.2 KDE. Оцінка щільності ядра – це непараметрична модель, яка використовується для оцінки розподілів ймовірностей. Саме така модель і буде використовуватися для того, щоб визначити потенційний рівень загрози на основі поточного стану споруди. Однак, перш ніж заглиблюватися в оцінку щільності ядра, корисно зрозуміти концепцію непараметричного оцінювання.

На відміну від традиційних методів оцінювання, непараметричне оцінювання не припускає, що дані взято з відомого розподілу. Навпаки, непараметричні моделі визначають структуру моделі на основі вихідних даних.

Прикладами непараметричних даних є:

- ранжовані або порядкові дані;
- дані без сильного теоретичного зв'язку з відомим розподілом;
- дані з аномаліями, такими як викиди, зсуви або «важкі хвости» [42].

Загальний принцип, що лежить в основі KDE, доволі простий – чим більше точок даних у вибірці, розташованих навколо певного місця, тим вища ймовірність того, що спостереження відбулося у цьому місці. Це реалізується за допомогою використання зважених відстаней всіх спостережень з різних місць на лінійно розташованому наборі точок. Формально KDE задається формулою,

яка зображена нижче:

$$d(x) = (1 / (nh)) * \sum K((x - xi) / h)$$

де X – певна точка, щільність для якої потрібно визначити, Xi – це точка з вибірки, отриманої з деякого розподілу з невідомою густиною, n – кількість значень у вибірці, K(...) – ядрова функція, h – параметр згладжування, який ще називають пропускною здатністю.

Математично, пропускна здатність – це параметр масштабування, який впливає на поширення функції ядра над даними. Він контролює, наскільки широким або вузьким є ядро навколо кожної точки даних, таким чином впливаючи на те, наскільки кожна точка робить внесок в оцінку щільності в будь-якій іншій точці. Вибір правильної пропускної здатності має вирішальне значення: занадто мала смуга пропускання призводить до того, що оцінка щільності є занадто нерівною, відображаючи шум, а не основний розподіл, тоді як занадто велика згладжує дані, потенційно приховуючи важливі особливості розподілу [43].

Функція ядра визначає, як обчислити щільність ймовірності, враховуючи відстань X - Xi. Існує багато функцій ядра, і методи вибору моделей на основі середньоквадратичної похибки можуть бути використані, щоб допомогти визначити, яка з них є найкращим вибором для базових даних.

Серед функцій ядра можна виділити:

- рівномірна;
- Епанечнікова;
- Гауса (нормальний розподіл) тощо.

Гаусовська функція ядра вважається однією із хороших стартових точок, адже вона дає непогані результати, при цьому не потребуючи великих комп’ютерних затрат, саме тому і було вирішено обрати її.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Проектування СД та розгортання OLAP-куба

4.1.1 Проектування СД. Як було зазначено в підрозділі 3.2, СД – є найчастішим джерелом даних для OLAP-куба, тому перш ніж починати аналіз, потрібно визначити структуру СД, а потім створити його. Структуру можна побачити на рис. 5.

Рис. 5 Сховище даних

Воно містить усю необхідну інформацію для аналізу, а саме:

- виміри DangerLevels (мінімально необхідна інформація про рівні загроз) та Constructions (мінімально необхідна інформація про споруди), які, будучи відфільтрованими в парі, дозволяють унікально ідентифікувати конкретну симуляцію, адже у однієї споруди може бути лише одна симуляція з кожним із рівнів загрози;
- вимір часу AccidentDuration_Dim, який містить опорні точки часу, який пройшов після початку симуляції;
- таблиця фактів Fact, яка містить ідентифікатори вимірів, звісно, і як раз головну інформацію для аналізу – кількість людей і культурних пам’яток, які постраждали внаслідок аварії, а також кількість задіяних систем оповіщення.

4.1.2 Побудова розгорнутого куба за допомогою служби SSAS. Після того, як було побудовано СД, можна приступати до підготовки та налаштування засобів для аналізу. Першим етапом підготовки є побудова OLAP-куба, що і було зроблено за допомогою служби SSAS.

Спочатку було створено та налаштовано джерело даних – це, по суті, просто зв'язок із СД, яке було створено раніше. Після цього потрібно було створити представлення даних, яке і дозволяє отримати інформацію про структуру сховища, і на основі цієї структури створити куб. Наступним кроком було створення вимірів на основі таблиць вимірів у СД. І, нарешті, після всіх цих приготувань можна створювати куб. Це робиться доволі схожим на створення виміру чином, однак замість атрибутів обираються групи мір, тобто факти, а також виміри. На рис. 6 можна побачити створений куб.

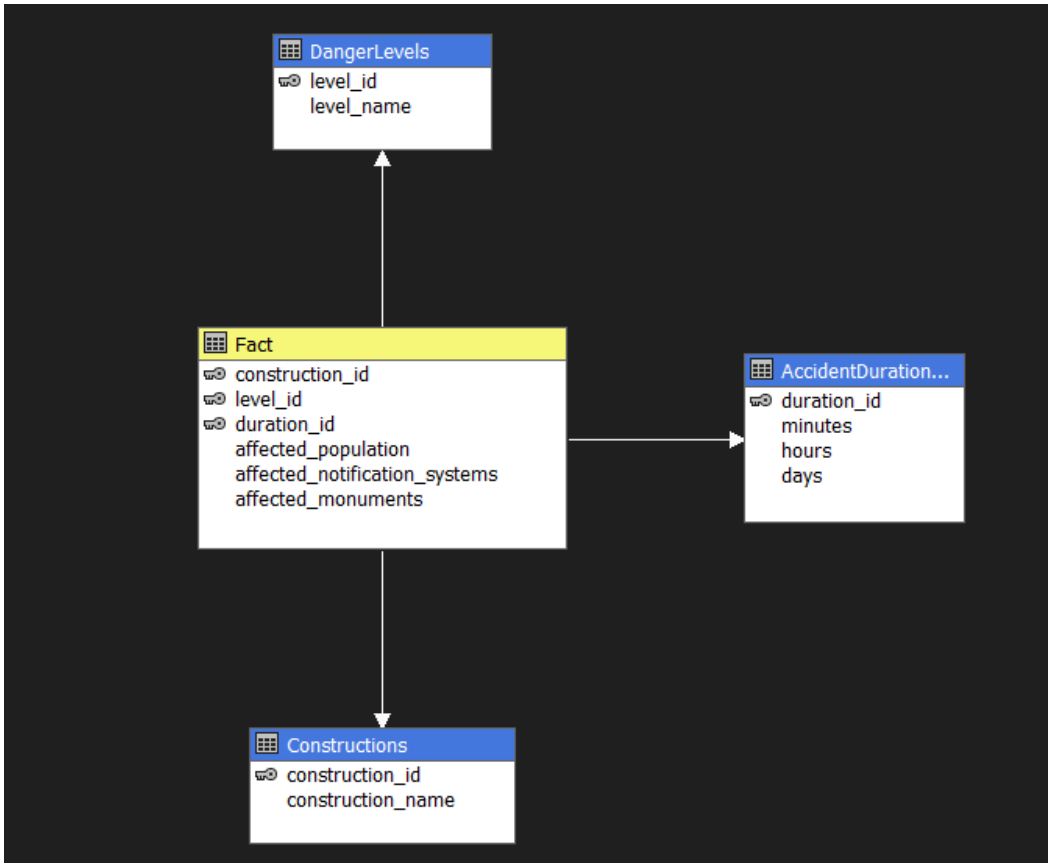


Рис. 6 Створений куб

На діаграмі куба таблиця фактів підсвічена жовтим, а також присутні стрілочки, які демонструють зв'язок між нею і вимірами, що говорить про те, що все було налаштовано правильно. Однак, щоб бути остаточно впевненим в

правильності створення куба, можна спробувати розгорнути його. Для цього було необхідно встановити Analysis Services для SQL Server, а також налаштувати режим роботи і надати користувачу права на роботу із сервером. І після цього розгортання дійсно пройшло успішно, а на рис. 7 можна побачити розгорнутий куб у середовищі SQL Server Management Studio.

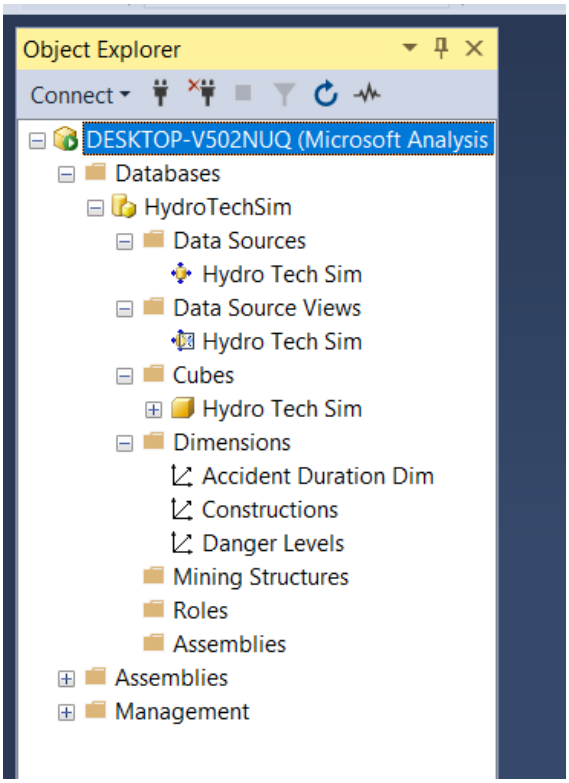


Рис. 7 Розгорнутий куб в SQL Server Management Studio

4.1.3 Реалізація отримання даних за допомогою Data Flow. В службі SSIS є інструмент Data Flow, який дозволяє графічно охарактеризувати процес перенесення потрібних даних із оперативної БД у СД [44]. Цей процес відбувається у два етапи: етап наповнення таблиць вимірів і етап наповнення таблиці фактів. Загальний процес наповнення із зазначенням етапів показаний на рис. 8.

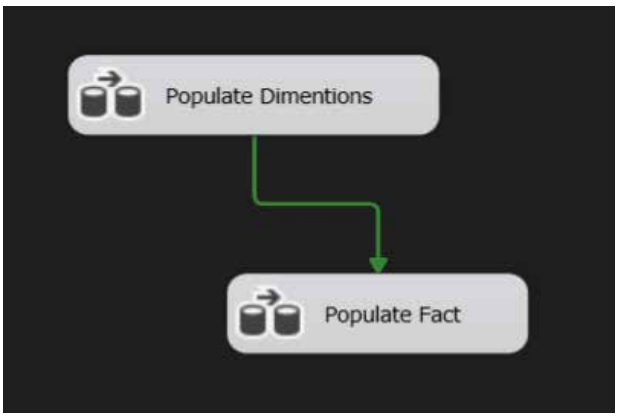


Рис. 8 Загальна схема процесу наповнення СД даними

Після цього необхідно визначити вхідний та вихідний зв’язок, вхідний – це зв’язок до БД, звідки будуть братися дані, а вихідний – це зв’язок до СД, куди потрібні дані будуть перенесені.

Отримавши валідні зв’язки із БД та СД можна приступати до безпосереднього визначення того, які саме дані і куди мають бути переміщені. Для цього створюється та налаштовується зв’язка джерел і призначень даних. Щоб така зв’язка була валідною, потрібно в джерелі і в призначенні вказати, який зв’язок їм використовувати, якої саме таблиці вони стосуються і які атрибути цих таблиць будуть приймати участь у передачі даних.

Таким же чином було налаштовано перенесення даних до інших двох вимірів, і в результаті було отримано схему, показану на рис. 9.

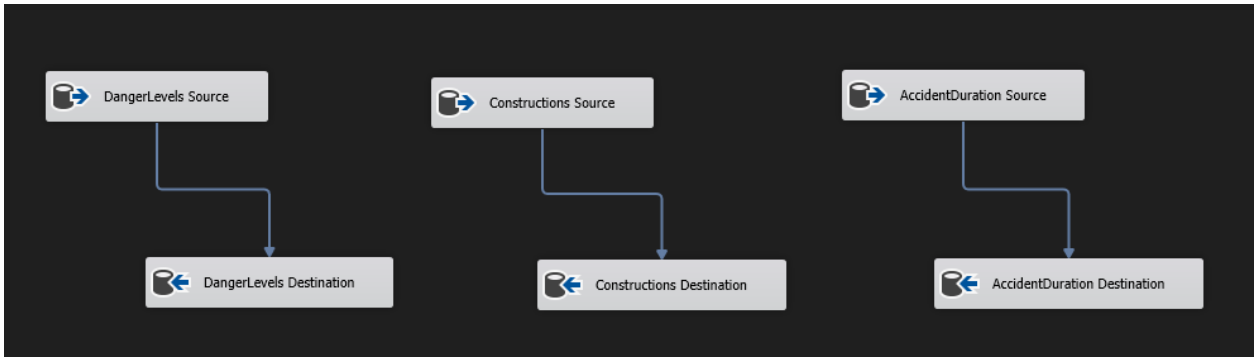
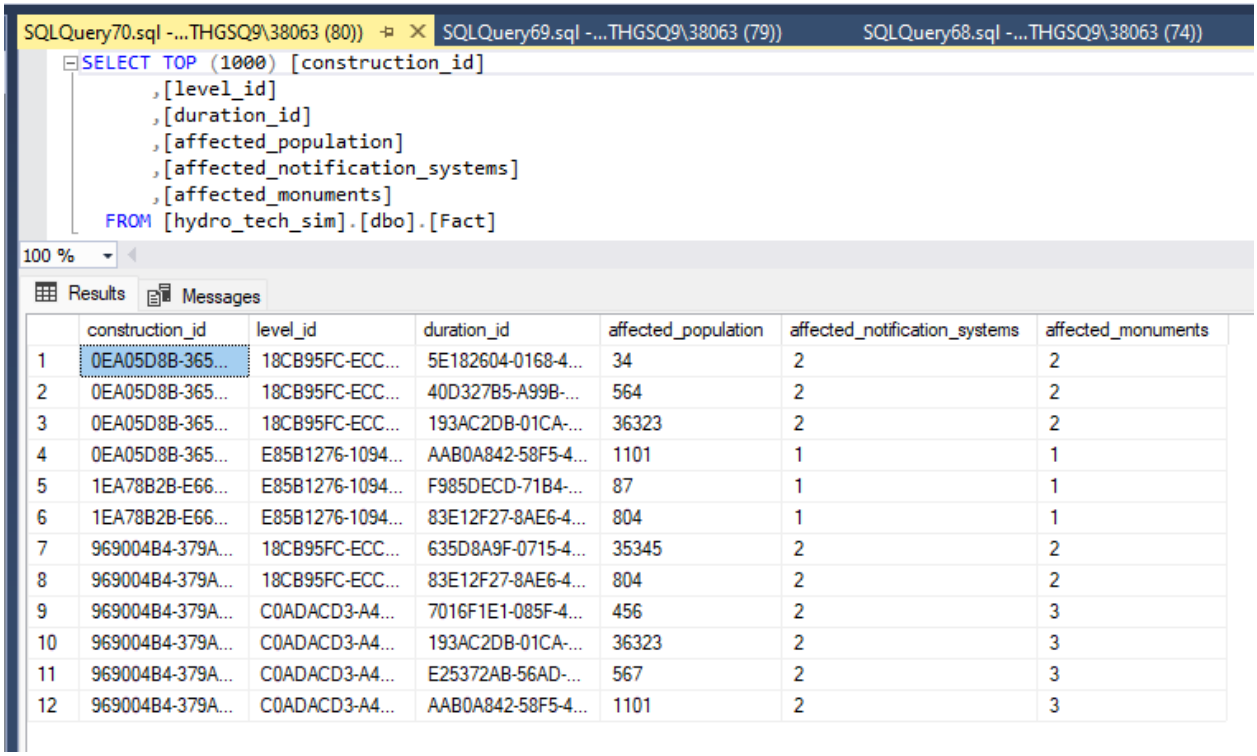


Рис. 9 Процес заповнення вимірів

Після цього можна визначити процес наповнення таблиці фактів. Загалом, він робиться аналогічним чином, тобто процес визначення передачі даних такий же, як і у вимірів. Головна відмінність між ними заключається в тому, що йому потрібен спеціальний SQL-запит, який може агрегувати потрібні дані із оперативної БД і записати їх у СД.

Після налаштування SSIS-проект було запущено і було отримано позитивний результат, що знаменує про успішне заповнення СД. Отримані дані в таблиці фактів можна побачити на рис. 10.



	construction_id	level_id	duration_id	affected_population	affected_notification_systems	affected_monuments
1	0EA05D8B-365...	18CB95FC-ECC...	5E182604-0168-4...	34	2	2
2	0EA05D8B-365...	18CB95FC-ECC...	40D327B5-A99B-...	564	2	2
3	0EA05D8B-365...	18CB95FC-ECC...	193AC2DB-01CA-...	36323	2	2
4	0EA05D8B-365...	E85B1276-1094...	AAB0A842-58F5-4...	1101	1	1
5	1EA78B2B-E66...	E85B1276-1094...	F985DECD-71B4-...	87	1	1
6	1EA78B2B-E66...	E85B1276-1094...	83E12F27-8AE6-4...	804	1	1
7	969004B4-379A...	18CB95FC-ECC...	635D8A9F-0715-4...	35345	2	2
8	969004B4-379A...	18CB95FC-ECC...	83E12F27-8AE6-4...	804	2	2
9	969004B4-379A...	C0ADACD3-A4...	7016F1E1-085F-4...	456	2	3
10	969004B4-379A...	C0ADACD3-A4...	193AC2DB-01CA-...	36323	2	3
11	969004B4-379A...	C0ADACD3-A4...	E25372AB-56AD-...	567	2	3
12	969004B4-379A...	C0ADACD3-A4...	AAB0A842-58F5-4...	1101	2	3

Рис. 10 Дані в таблиці фактів

4.2 Побудова звітності та розрахунок KPI

4.2.1 Побудова звітності. Для побудови звітності раніше було обрано середовище Microsoft Power BI. Для початку роботи з даними в Power BI необхідно визначити джерело даних, у даному випадку для цього є раніше розгорнутий куб. Після цього можна приступати до аналітики.

На рис. 11 можна побачити діаграму з кількістю постраждалих внаслідок аварії на кожній із споруд на кожному із можливих рівнів загрози станом на кінець симуляції.

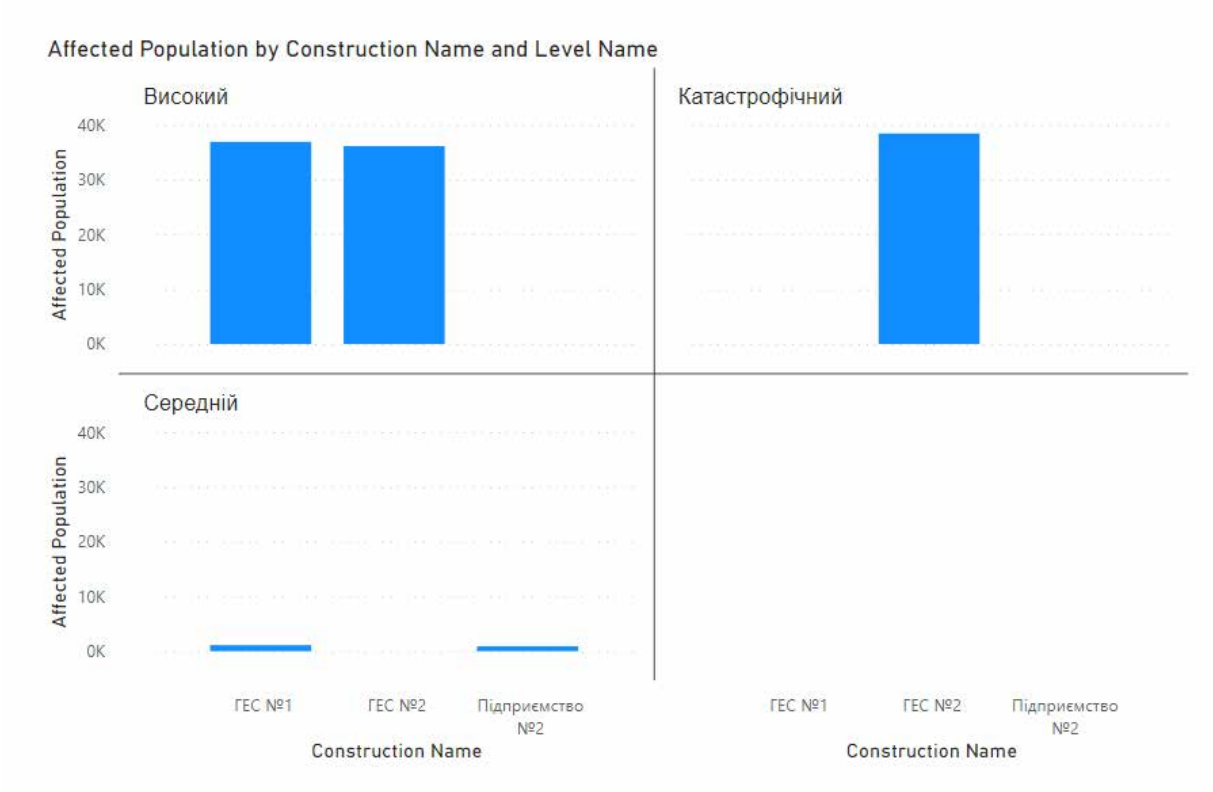


Рис. 11 Кількість постраждалих по спорудам за рівнями загрози

На цій діаграми відсутній низький рівень загрози, а також Підприємство №1 через те, що аварія на цьому підприємстві із цим рівнем загрози не потягла за собою ніяких збитків, а це зобразити неможливо.

Остаточні збитки не є безкорисними, вони дозволяють оцінити максимальну шкоду від різних аварій як на різних, так і на однакових спорудах, однак може виникнути необхідність оцінки прогресу того, як протягом аварії починає затоплювати все більшу і більшу територію, що і можна побачити на рис. 12. Задля контрасту наслідків, зліва та справа вказані дві різні симуляції двох різних споруд.

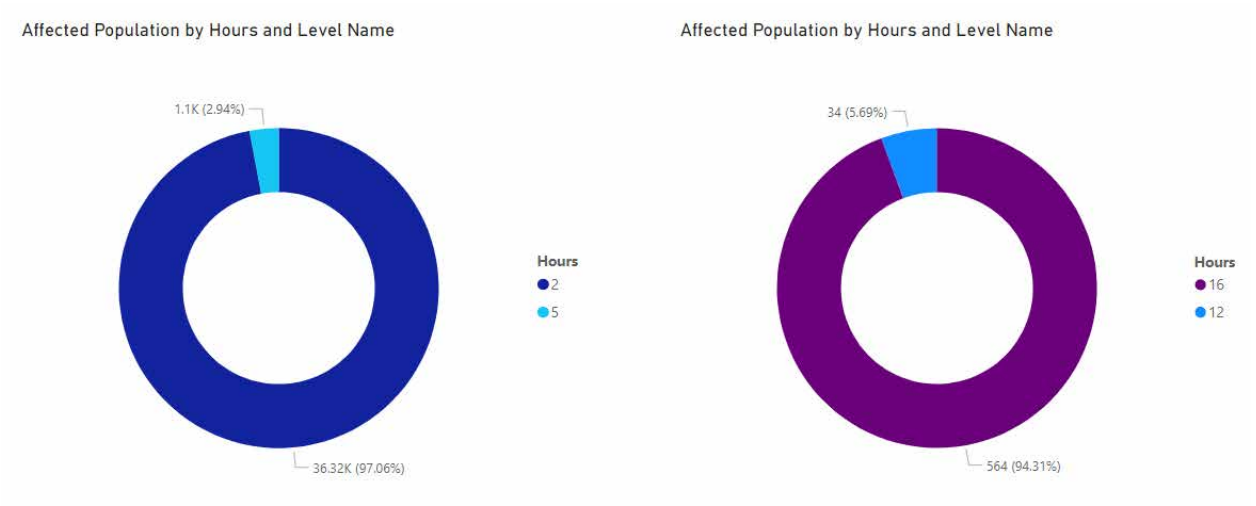


Рис. 12 Діаграма прогресу затоплення

На даній діаграмі показаний процес затоплення під час аварій на двох різних спорудах. На круговій діаграмі зліва можна побачити, наскільки швидко вода дійшла до першого і великого населеного пункту, що репрезентовано темно-синьою областю і кількістю постраждалих станом на другу годину після початку аварії, а через три години вода дійшла і до другого невеликого населеного пункту, що репрезентовано меншою і світло-синьою областю.

На круговій діаграмі справа можна побачити, що воді знадобилося аж 12 годин, щоб досягнути першого якогось дуже маленького населеного пункту (показано світло синім), і вже потім вона дійшла до другого, дещо більшого населеного пункту (показано фіолетовим).

4.2.2 Розрахунок KPI. Задля розрахунку KPI було використано засоби того ж SSAS-проекту, що і був створений задля побудови розгорнутого кубу. В ньому KPI представляються у вигляді набору розрахунків, пов'язаних з групою показників у кубі. До компонентів цього розрахунку відносяться:

- вираз значення – основний показник, який відстежується (наприклад, дохід від продажів, оцінка задоволеності клієнтів);
- вираз мети – цільове значення або поріг для KPI (наприклад, дохід в $1 млн, рівень задоволеності 85%);
- вираз стану – показує, чи досягає показник мети, перевищує її або не досягає (наприклад, за допомогою порогових значень, піктограм або кольорів)

[45].

Як вже було сказано в розділі 3.3, для даної системи у якості KPI було вирішено обрати два показники:

- PopulationToSystemsRatio_KPI – перший показник, який визначає середню кількість населення в кордонах задіяних систем оповіщення, показує, наскільки високою є середня густота населення в регіоні під загрозою;
- MonumentsToSystemsRatio_KPI – другий показник, який, аналогічно до попереднього, визначає середню густоту культурних пам’яток в регіоні під загрозою.

Налаштування відповідних цим показникам виразів значення, мети та стану показані на рис. 13 та 14 відповідно.

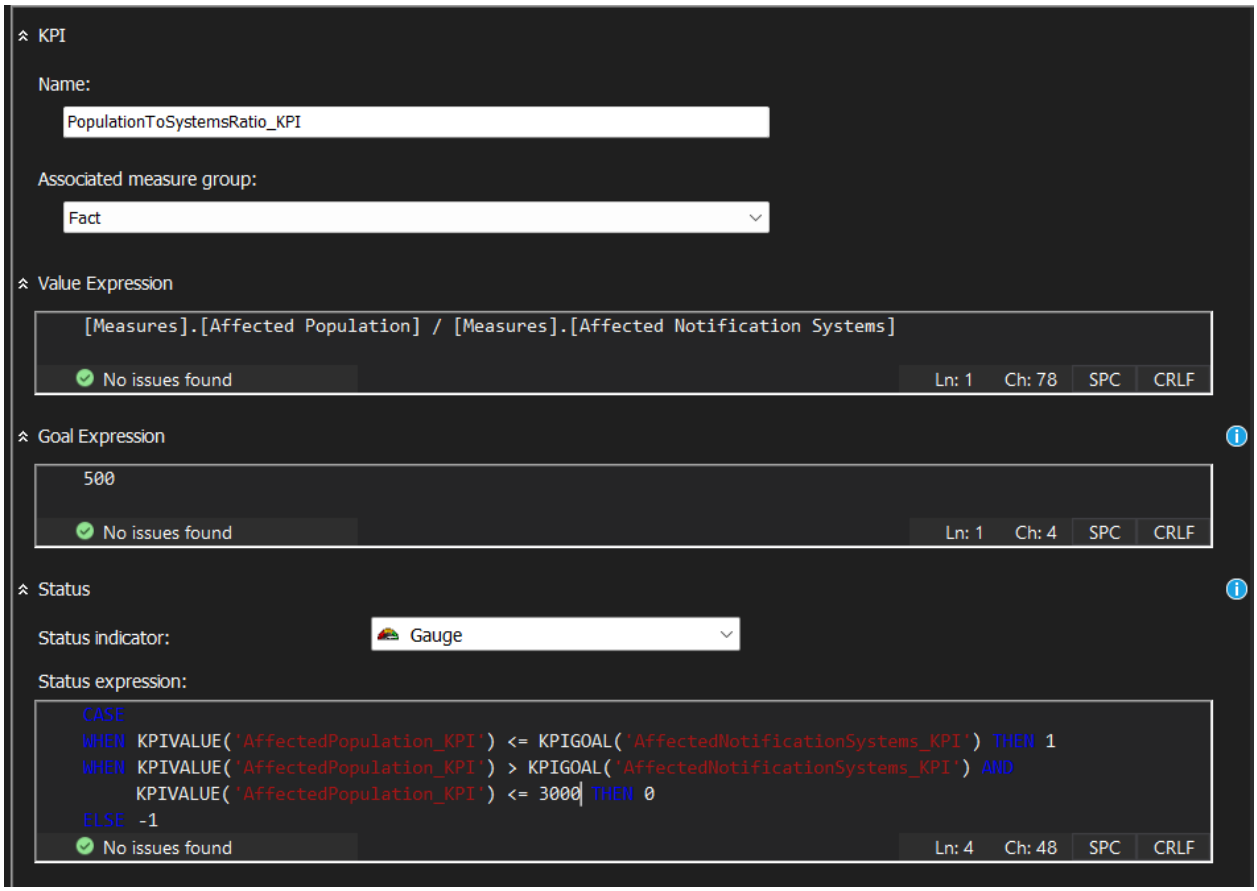
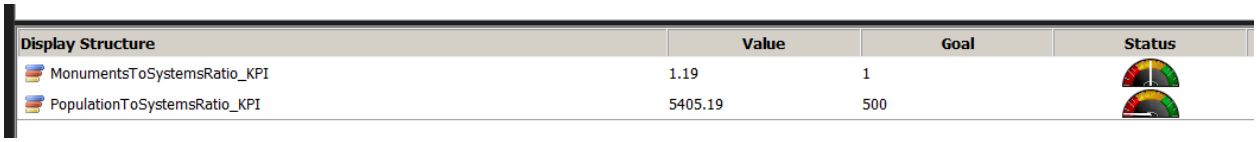


Рис. 13 PopulationToSystemsRatio_KPI

Рис. 14 MonumentsToSystemsRatio_KPI

Після того, як KPI було описано, їх потрібно опрацювати. Якщо опрацювання пройшло успішно, результат цього опрацювання можна побачити у вигляді агрегованого по всім параметрам значення, що і показано на рис. 15.



Display Structure	Value	Goal	Status
MonumentsToSystemsRatio_KPI	1.19	1	
PopulationToSystemsRatio_KPI	5405.19	500	

Рис. 15 Результат опрацювання KPI

Однак загальне значення KPI не є дуже корисним, а тому використовуючи той же Power BI було візуалізовано і його. На рис. 16 показано одночасно обидва KPI для тих же споруд і симуляцій, і за той же проміжок часу, що і на рис. 12.

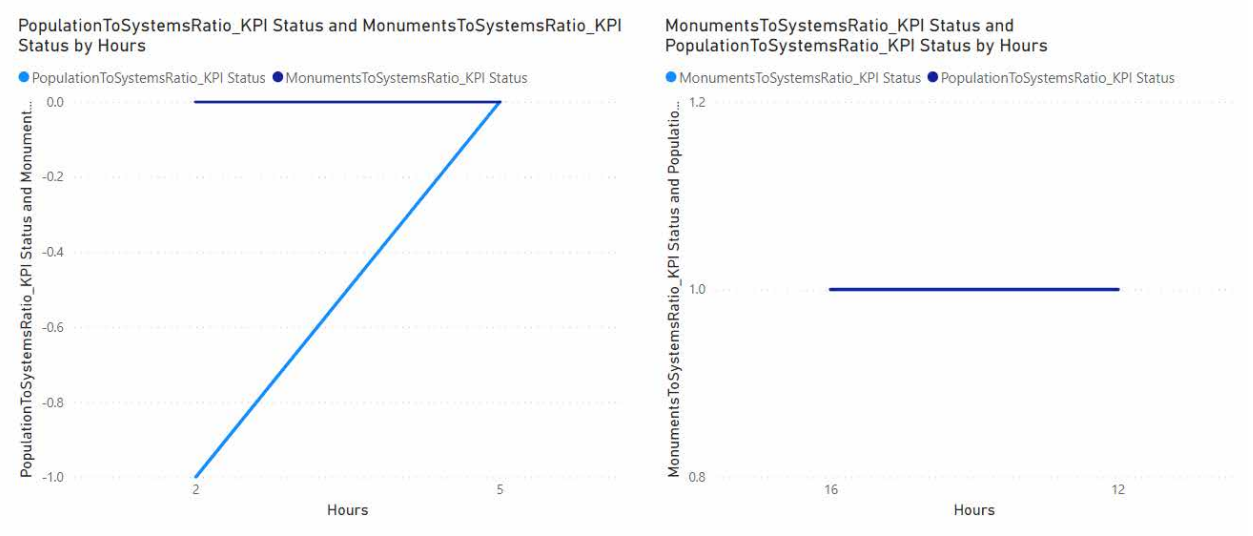


Рис. 16 Візуалізація KPI

Як можна побачити, KPI симуляції справа весь час мають значення 1, а це додатково підтверджує, що дана аварія не несе суттєвої загрози. А KPI симуляції зліва показує, що стосовно культурних пам'яток дана аварія на даному проміжку часу несе загрозу середнього рівня. Однак стосовно кількості населення, яка постраждала від аварії, KPI показує, що станом на дві години після аварії загроза суттєва, однак вона до п'ятої години після аварії загроза спадає також до середнього рівня. Це може бути пов'язано або з особливостями місцевості, які не дають воді розлитися на більшу площу, або з тим, що навколо тих населених пунктів, що постраждали, більше немає населених пунктів.

4.3 Оцінка стану споруд в реальному часі

В пункті 3.3.2 було зазначено, що для визначення рівня загрози на основі поточного стану гідротехнічних споруд буде використано непараметричний метод KDE. Він був реалізований у вигляді окремого модуля, якому потрібно передати усі наявні дані про споруду, тобто всі результати симуляцій і пов'язані з ними параметри споруд, а також поточні параметри цієї ж споруди.

Усі наявні дані про споруду отримуються із БД самої експертної системи, а поточні дані про цю ж споруду отримуються шляхом надсилання запиту на сервер споруди. Ці процеси не будуть висвітлені в рамках цієї магістерської

роботи, адже процес отримання усіх наявних даних, по суті, є лише комплексним запитом до БД, а отримання поточних даних залежить від специфіки роботи серверів споруд (тобто у різних споруд можуть бути різні запити і різні формати передачі даних), тому їх неможливо універсально описати. Що буде розглянуто, так це приклад того, як саме модуль з KDE визначає потенційний рівень загрози.

Тож, спочатку дані. Так як сама по собі експертна система не містить спеціалізованих даних, наприклад, можливі параметри споруд та потрібні одиниці виміру, бо для них потрібна експертна оцінка, було вирішено провести самостійне дослідження того, які параметри можуть бути у споруди, та які значення мають сенс [46][47]. Таким чином було створено класи, які описують потрібні дані, а потім було створено об’єкти класів, які містять конкретні значення як усіх наявних даних про споруди, так і поточних даних про ці ж споруди. Створені класи можна побачити на рис. 17.

```python
class StructuralCondition(IntEnum):  17 usages  Alex
    GOOD = 1
    MODERATE = 2
    POOR = 3


@dataclass  4 usages  Alex
class ConstructionParams:
    id: int
    max_volume: float
    max_water_level: float
    earthquake_resistance: float
    max_wind_speed: float | None = None
    max_inflow_rate: float | None = None
    min_inflow_rate: float | None = None


@dataclass  19 usages  Alex
class ConstructionState:
    id: int
    volume: float
    water_level: float
    wind_speed: float | None = None
    inflow_rate: float | None = None
    operational_temp: float | None = None
    earthquake_rating: float | None = None
    structural_condition: StructuralCondition | None = None


class DangerLevels(IntEnum):  10 usages  Alex
    LOW = 1
    MODERATE = 2
    HIGH = 3
    CRITICAL = 4
```

Рис. 17 Класи, які описують дані

Першим є перелік із структурних станів споруд, другий – це параметри споруд, третій – це інформація про поточний стан споруд, а четвертий – це перелік з рівнями загрози.

Серед параметрів присутні наступні: максимальний об’єм, максимальний рівень води, стійкість до землетрусів (максимальне допустиме значення по шкалі

Ріхтера), максимальна швидкість вітру, яку може витримати споруда, максимально допустимий об’єм вхідної води та мінімальний гарантований об’єм вихідної води. Останні три параметри можуть бути відсутні. Поточний стан споруди містить в собі точно такий же набір даних, однак замість того, щоб показувати можливі значення, він вже демонструє саме конкретні.

Заповнення наявних та поточних даних можна побачити на рис. 18.

```
constructions: list[ConstructionParams] = [
    ConstructionParams(id=1, max_volume=100, max_water_level=0.9, max_wind_speed=30, earthquake_resistance=6),
    ConstructionParams(id=2, max_volume=10000, max_water_level=0.8, max_inflow_rate=30, earthquake_resistance=8),
    ConstructionParams(id=3, max_volume=1000000, max_water_level=0.85, max_inflow_rate=500, min_inflow_rate=30, earthquake_resistance=8),
]
data: dict[DangerLevels, list[ConstructionState]] = {
    DangerLevels.LOW: [
        ConstructionState(id=1, volume=50, water_level=0.45, wind_speed=5, structural_condition=StructuralCondition.GOOD),
        ConstructionState(id=1, volume=90, water_level=0.9, wind_speed=15, structural_condition=StructuralCondition.GOOD),
        ConstructionState(id=1, volume=70, water_level=0.65, wind_speed=20, structural_condition=StructuralCondition.MODERATE),
        ConstructionState(id=2, volume=8000, water_level=0.7, inflow_rate=20, structural_condition=StructuralCondition.GOOD),
        ConstructionState(id=2, volume=3000, water_level=0.4, inflow_rate=10, structural_condition=StructuralCondition.POOR),
    ],
    DangerLevels.MODERATE: [
        ConstructionState(id=2, volume=10000, water_level=0.8, inflow_rate=20, structural_condition=StructuralCondition.GOOD),
        ConstructionState(id=2, volume=11000, water_level=0.9, inflow_rate=15, structural_condition=StructuralCondition.POOR),
        ConstructionState(id=3, volume=100000, water_level=0.1, inflow_rate=50, structural_condition=StructuralCondition.POOR),
        ConstructionState(id=3, volume=300000, water_level=0.3, inflow_rate=200, structural_condition=StructuralCondition.MODERATE),
    ],
    DangerLevels.HIGH: [
        ConstructionState(id=3, volume=700000, water_level=0.65, inflow_rate=100, structural_condition=StructuralCondition.MODERATE),
        ConstructionState(id=3, volume=300000, water_level=0.25, inflow_rate=150, structural_condition=StructuralCondition.POOR),
    ],
    DangerLevels.CRITICAL: [
        ConstructionState(id=3, volume=800000, water_level=0.75, inflow_rate=700, structural_condition=StructuralCondition.MODERATE),
        ConstructionState(id=3, volume=1200000, water_level=0.95, inflow_rate=300, structural_condition=StructuralCondition.POOR),
    ]
}
new_conditions: list[ConstructionState] = [
    ConstructionState(id=1, volume=115, water_level=0.95, wind_speed=15, structural_condition=StructuralCondition.POOR),
    ConstructionState(id=2, volume=9000, water_level=0.7, inflow_rate=100, structural_condition=StructuralCondition.MODERATE),
    ConstructionState(id=3, volume=500000, water_level=0.65, inflow_rate=200, structural_condition=StructuralCondition.MODERATE),
]
```

Рис. 18 Заповнення даних

Після цього відбувається процес розрахунку, який можна описати наступним чином: почергово всі наявні параметри, включаючи поточну кількість симуляцій з різним рівнем загроз, передаються в окрему Гаусовську модель KDE, і після цього на основі розрахунків ймовірностей всіх параметрів та поєднання їх разом визначається найбільш ймовірний потенційний рівень загрози.

Створення моделі KDE для рівнів загроз можна побачити на рис. 19, а створення моделей KDE для кожного з валідних параметрів споруд (тобто таких, що присутні хоча б в двох спорудах) на кожному з рівнів загрози можна побачити на рис. 20.

```python
# Historical danger levels
historical_danger_levels = [dl.value for dl in data for danger_level_lst in data[dl]]
random.shuffle(historical_danger_levels)
historical_danger_levels = np.array(historical_danger_levels)
# Fit a KDE model to the historical danger levels
kde_danger_levels = gaussian_kde(historical_danger_levels)

# Function to get probability for a specific danger level
def get_danger_level_probability(level):  1 usage  Alex
    density = kde_danger_levels.evaluate([level])[0]
    # Normalize to get a proper probability
    normalized_density = density / kde_danger_levels.integrate_box_1d( low: 1, len(DangerLevels))
    return normalized_density


# Calculate probabilities for each danger level (Low, Moderate, High)
danger_level_probs = [
    get_danger_level_probability(level) for level in DangerLevels
]

# Normalize to ensure the probabilities sum to 1
danger_level_probs = np.array(danger_level_probs)
danger_level_probs /= danger_level_probs.sum()
```

Рис. 19 Створення моделі KDE для рівнів загроз

```python
# Build KDE models with bandwidth adjustment for each parameter and danger level
for input_construction in new_conditions:
    # print("=========================================================")
    # print(f"Construction {input_construction.id}")
    kde_models = {}

    for danger_level in data:
        # print(f"Danger level {danger_level}")
        records: list[ConstructionState] = list(filter(lambda x: x.id == input_construction.id, data[danger_level]))

        if len(records) > 1:
            kde_models[danger_level] = {}

            for field in [f for f in fields(input_construction) if f.name != "id"]:
                field_data = [getattr(record, field.name) for record in records]
                valid_field_data = list(filter(lambda x: x is not None, field_data))

                if len(valid_field_data) > 1:
                    avrg = sum(valid_field_data) / len(valid_field_data)
                    kde = gaussian_kde(list(map(lambda x: x if x is not None else avrg, field_data)), bw_method='scott')
                    kde_models[danger_level][field.name] = kde
```

Рис. 20 Створення моделей KDE для кожного з параметрів

Визначення найбільш ймовірного потенційного рівня загрози відбувається у два етапи: спочатку результати розрахунку кожної з потрібних моделей KDE заносяться в матрицю ймовірностей, з неї розраховується спільна ймовірність усіх змінних в рамках кожного з рівнів загрози, яка потім множиться з результатами моделі KDE для рівнів загроз. Отримані ймовірності

нормалізуються, і з них обирається найбільша. Все це показано на рис. 21.

```
fields_lengths = [len(list(kde_models[danger_level].keys())) for danger_level in DangerLevels if danger_level in kde_models]
likelihood_matrix = np.zeros((len(DangerLevels), max(fields_lengths)))

for danger_level in DangerLevels:
    if danger_level in kde_models:
        # Compute the likelihood matrix
        field_names = list(kde_models[danger_level].keys())

        for index, field_name in enumerate(field_names):
            # Evaluate the KDE for each parameter and danger level
            kde = kde_models[danger_level][field_name]
            likelihood_matrix[danger_level.value - 1, index] = kde.evaluate([getattr(input_construction, field_name)])[0]

likelihood_matrix = np.array(likelihood_matrix)
# print("Likelihood Matrix:")
# print(likelihood_matrix)

# Calculate the joint likelihood for each danger level (product of row entries)
joint_likelihoods = likelihood_matrix.prod(axis=1)

# Compute unnormalized posteriors
unnormalized_posteriors = joint_likelihoods * danger_level_probs

# Normalize the posteriors
posterior_probabilities = unnormalized_posteriors / unnormalized_posteriors.sum()
```

Рис. 21 Визначення найбільш ймовірного потенційного рівня загрози

Звісно, що перш ніж застосовувати цю систему, потрібна експертна оцінка фахівців на правильність її роботи на основі справжніх даних. Якщо їхній висновок покаже, що вона працює коректно – результати її роботи можна застосовувати для завчасного попередження населення та підготовки сил надзвичайних служб.

ВИСНОВКИ

У ході виконання даної магістерської кваліфікаційної роботи було розроблено експертну систему моніторингу наслідків аварій на гідротехнічних спорудах. Вона була побудована шляхом розширення розробленого раніше програмного симулятору наслідків аварій на гідротехнічних спорудах.

Магістерська робота розділена на чотири частини. У першій частині було проведено системний аналіз предметної області, а саме було описано предметну область, проаналізовано наявні рішення, а також було поставлено задачі до виконання.

У другій частині було змодельовано систему, а саме було розроблено діаграми прецедентів, діяльності та послідовності, а також було описано взаємодію користувачів із системою через Use-Case та User Story.

У третій частині було розроблено саму експертну систему: було описано архітектуру системи, визначено технології, якими це реалізовано, а також було обрано та описано алгоритми, за допомогою яких в подальшому було реалізовано аналітичну та моніторингову складові системи.

У четвертій частині було продемонстровано технічні аспекти реалізації аналітичної та моніторингової частин експертної системи. У першому підрозділі було показано механізми вилучення, обробки та передачі даних за допомогою OLAP-куба, заповнення СД даними за допомогою інструменту SSIS Data Flow. У другому підрозділі було показано налаштування засобів для розрахунку KPI, а також побудову аналітичних звітів у середовищі Microsoft Power BI. У другому розділі У третьому підрозділі було продемонстровано роботу модуля для визначення потенційного рівня загрози на основі синтезованих даних про поточний стан споруди.

У результаті було створено та налаштовано ефективне рішення для проведення подальшого аналізу даних, і для демонстрації можливостей, було самостійно проведено попередній аналіз вже наявних у системі даних. Окрім

цього, було реалізовано рішення для моніторингу поточного стану споруд в реальному часі, що значно розширює користь від використання даної експертної системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система оповіщення цивільного захисту. Купити, встановити та налаштувати автоматизацію [Електронний ресурс]. – Режим доступу: https://leater-ict.com/catalog/sistema-spovischennya

2. InfraWorks | Get Prices & Buy InfraWorks 2023 | Autodesk [Електронний ресурс]. – Режим доступу: https://www.autodesk.com/products/infraworks/overview?term=1-YEAR&tab=subscription

3. Національна бібліотека України імені В. І. Вернадського [Електронний ресурс]. – Режим доступу: nbuv.gov.ua

4. Спеціалізована БД "Винаходи (корисні моделі) в Україні" [Електронний ресурс]. – Режим доступу: http://base.uipv.org/searchINV/

5. Espacenet - Home page [Електронний ресурс]. – Режим доступу: http://worldwide.espacenet.com/?locale=en_EP

6. Patent Public Search Basic | USPTO [Електронний ресурс]. – Режим доступу: https://ppubs.uspto.gov/pubwebapp/static/pages/ppubsbasic.html

7. Espacenet - Bibliographic data [Електронний ресурс]. – Режим доступу: https://worldwide.espacenet.com/publicationDetails/biblio?II=2&ND=3&adjacent=true&locale=en_EP&FT=D&date=20240419&CC=CN&NR=117910365A&KC=A

8. Espacenet - Bibliographic data [Електронний ресурс]. – Режим доступу: https://worldwide.espacenet.com/publicationDetails/biblio?II=1&ND=3&adjacent=true&locale=en_EP&FT=D&date=20240719&CC=CN&NR=118364749A&KC=A

9. Espacenet - Bibliographic data [Електронний ресурс]. – Режим доступу: https://worldwide.espacenet.com/publicationDetails/biblio?II=3&ND=3&adjacent=true&locale=en_EP&FT=D&date=20240416&CC=CN&NR=117893015A&KC=A

10. FLOOD DEPTH ESTIMATION APPARATUS, FLOOD DEPTH ESTIMATION METHOD, COMPUTER READABLE MEDIUM, AND TRAINING APPARATUS; DOCUMENT ID US 20240345287 A1; Int. Cl. G01W1/00; DATE PUBLISHED 2024-10-17;

11. Water supply control system that implements safety controls and uses simulation to prevent commands that would cause or worsen flooding; DOCUMENT ID US 12116740 B1; Int. Cl. G05B17/02; DATE PUBLISHED 2024-10-15;

12. Unified Modeling Language (UML) description, UML diagram examples, tutorials and reference for all types of UML diagrams - use case diagrams, class, package, component, composite structure diagrams, deployments, activities, interactions, profiles, etc. [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/

13. Use case diagrams are UML diagrams describing units of useful functionality (use cases) performed by a system in collaboration with external users (actors). [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/use-case-diagrams.html

14. UML actor is a role played by a human user of the designed system, some other system or hardware that interacts with the subject by using services of the subject. [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/use-case-actor.html

15. User Stories | Examples and Template | Atlassian [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/agile/project-management/user-stories

16. How INVEST helps team write effective user stories… | by Saba | Medium [Електронний ресурс]. – Режим доступу: https://sabaimam.medium.com/investing-in-user-stories-c7cfb1fc5e85

17. What is a Use Case? [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchsoftwarequality/definition/use-case

18. UML activity diagrams are UML behavior diagrams which show flow of control or object flow with emphasis on the sequence and conditions of the flow.

[Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/activity-diagrams.html

19. Explore the UML sequence diagram - IBM Developer [Електронний ресурс]. – Режим доступу: https://developer.ibm.com/articles/the-sequence-diagram/

20. Web Server Components Deployment Scenarios - Business Central | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/deployment-scenarios

21. About Python™ | Python.org [Електронний ресурс]. – Режим доступу: https://www.python.org/about/

22. The web framework for perfectionists with deadlines | Django [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/

23. Overview of Online Analytical Processing (OLAP) - Microsoft Support. [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/office/overview-of-online-analytical-processing-olap-15d2cdde-f70b-4277-b009-ed732b75fdd6

24. django-phonenumbers · PyPI [Електронний ресурс]. – Режим доступу: https://pypi.org/project/django-phonenumbers/

25. django-phonenumber-field · PyPI [Електронний ресурс]. – Режим доступу: https://pypi.org/project/django-phonenumber-field/

26. Welcome to Django Leaflet's documentation! — Django Leaflet 0.20 documentation [Електронний ресурс]. – Режим доступу: https://django-leaflet.readthedocs.io/en/latest/

27. Django Slick Reporting – Welcome [Електронний ресурс]. – Режим доступу: https://django-slick-reporting.com/

28. ANUGA [Електронний ресурс]. – Режим доступу: https://anuga.anu.edu.au/

29. PostgreSQL: About [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/about/

30. PostGIS [Електронний ресурс]. – Режим доступу: https://postgis.net/

31. PostgreSQL driver for Python — Psycopg [Електронний ресурс]. – Режим доступу: https://www.psycopg.org/

32. The Django template language | Django documentation | Django [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/4.2/ref/templates/language/

33. Anaconda | The World's Most Popular Data Science Platform [Електронний ресурс]. – Режим доступу: https://www.anaconda.com/

34. PyCharm: the Python IDE for Professional Developers by JetBrains [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/pycharm/

35. Docker: Accelerated, Containerized Application Development [Електронний ресурс]. – Режим доступу: https://www.docker.com/

36. Overview of Online Analytical Processing (OLAP) - Microsoft Support. [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/office/overview-of-online-analytical-processing-olap-15d2cdde-f70b-4277-b009-ed732b75fdd6

37. What Is a Data Warehouse | Oracle. [Електронний ресурс]. – Режим доступу: https://www.oracle.com/database/what-is-a-data-warehouse/

38. Overview of OLAP cubes for advanced analytics | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/system-center/scsm/olap-cubes-overview?view=sc-sm-2022

39. SQL Server Analysis Services overview | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/analysis-services/ssas-overview?view=asallproducts-allversions

40. What is Power BI? - Power BI | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview

41. KPI. Що таке KPI і ключові показники ефективності або страшний сон менеджера [Електронний ресурс]. – Режим доступу: https://www.dsnews.ua/ukr/economics/strashnyy-son-menedzhera-chto-takoe-kpi-18062021-428687

42. The Fundamentals of Kernel Density Estimation | Aptech [Електронний ресурс]. – Режим доступу: https://www.aptech.com/blog/the-fundamentals-of-kernel-density-estimation/

43. Essential Math for Machine Learning: Kernel Density Estimation | by Dagang Wei | Medium [Електронний ресурс]. – Режим доступу: https://medium.com/@weidagang/essential-math-for-machine-learning-kernel-density-estimation-d014df073770

44. Data Flow Task - SQL Server Integration Services (SSIS) | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/sql/integration-services/control-flow/data-flow-task?view=sql-server-ver16

45. Key Performance Indicators (KPIs) in Multidimensional Models | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/analysis-services/multidimensional-models/key-performance-indicators-kpis-in-multidimensional-models?view=asallproducts-allversions

46. (PDF) Program of hydropower potential assessment as an effective possibilities in Upper Vistula water region in Poland [Електронний ресуср]. – Режим доступу: https://www.researchgate.net/publication/332673903_Program_of_hydropower_potential_assessment_as_an_effective_possibilities_in_Upper_Vistula_water_region_in_Poland

47. (PDF) Assessment of flood hazard areas at a regional scale using an index-based approach and Analytical Hierarchy Process: Application in Rhodope–Evros region, Greece [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/281291980_Assessment_of_flood_hazard_areas_at_a_regional_scale_using_an_index-based_approach_and_Analytical_Hierarchy_Process_Application_in_Rhodope-Evros_region_Greece