МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

Інтелектуальна система визначення рівня критичності елементів організаційної системи

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу,
групи КН- 41

_____Набережний А.В._____
(прізвище та ініціали)

Керівник __Гнатієнко Г.М._____
(прізвище та ініціали)

_____канд. техн. наук,____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра інтелектуальних технологій
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____
"___"_____2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Набережному Артуру
Вячеславовичу_____
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)
Інтелектуальна система визначення рівня критичності елементів організаційної системи
_____
_____
___
_____
___
_____
___

затверджена протоколом засідання кафедри від «23» грудня 2021 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)

Інтелектуальна система визначення рівня критичності елементів організаційної системи створюється для визначення та кількісної оцінки рівня критичності елементів організаційної системи на основі аналізу прямих та непрямих даних_____
_____
___
_____
___
_____
___

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Вибір та аналіз предметної області, постановка задачі визначення рівня критичності елементів організаційної системи, огляд джерел інформації щодо функціонування елементів організаційної системи, проєктування архітектури системи, огляд обраних засобів, розробка системи, вибір тестових задач, тестування роботи системи_____
_____
___
_____
___

5. Перелік презентаційного матеріалу (з точним зазначенням обов’язкових презентацій)

Огляд джерел інформації щодо функціонування елементів організаційної системи, ілюстрація ситуації прийняття рішення, архітектура системи, програмна реалізація системи, презентація Power Point

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Вибір та аналіз предметної області		
2	Постановка задачі визначення рівня критичності елементів організаційної системи		
2	Огляд особливостей теорії управління організаційними системами		
2	Вибір інструментів для визначення рівня критичності елементів організаційної системи		
3	Проєктування архітектури системи		
3	Програмна реалізація системи		
3	Вибір тестових задач, тестування роботи системи		

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / / (підпис) (ПІБ)

Завдання прийняв до виконання _____ / / (підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Вибір та аналіз предметної області	25.01-01.02	
2.	Постановка задачі визначення рівня критичності елементів організаційної системи	01.02-07.02	
3.	Огляд особливостей теорії управління організаційними системами	08.02-14.02	
4.	Вибір інструментів для визначення рівня критичності елементів організаційної системи	15.02-25.02	
5.	Проєктування архітектури системи	26.02-07.03	
6.	Програмна реалізація системи	08.03-25.04	
7.	Вибір тестових задач, тестування роботи системи	26.04-07.05	

Студент-дипломник _____ / / (підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / / (підпис) (ПІБ)

Анотація

Набережний Артур Вячеславович виконав випускну кваліфікаційну роботу на тему «Інтелектуальна система визначення рівня критичності елементів організаційної системи» за спеціальністю 122 – «Комп'ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних методів оцінки критичності об'єктів організаційних систем, розглянуто особливості різних типів організацій, розроблено інформаційне та програмне забезпечення, що виконує процедури оцінки критичності об'єктів.

Ключові слова: організаційні системи, критичність об'єктів, експертна оцінка.

Summary

The degree project: «Intellectual system for determining the level of criticality of elements in the organizational system» has completed by **Naberezhnyi Artur** specialty 122 – «Computer Sciences».

At the graduation qualification work, an analysis of modern methods for assessing the criticality of objects of organizational systems was carried out, the features of various types of organizations were examined, information and software security was disaggregated, which includes the procedures for assessing the criticality of objects.

Keywords: organizational systems, criticality of objects, expert review.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС — інтелектуальна система

ОргС — організаційна система

КЕ — критичний елемент

БД — база даних

DAO — Data Access Object

FMCG — Fast Moving Consumer Products

ВСТУП

Сучасне суспільство збільшується зі зростаючою швидкістю. Задля підтримки власного функціонування, ОргС набувають все більш складних форм. Вище сказане стосується всіх рівнів організації, починаючи з невеликих приватних компаній, закінчуючи державами. Нерідко зі збільшенням складності структури виникають елементи, вихід із ладу яких може причинити великі збитки, а інколи навіть ставити під загрозу функціонування системи загалом. Виявлення подібних КЕ та забезпечення їхньої безпеки є однією з основних цілей збільшення стійкості ОргС.

На сьогоднішній день більшість робіт розглядають методи оцінки критичності елементів державних систем. У цьому випадку, об'єктами є заводи, автомагістралі, електростанції, музеї тощо. Така ситуація склалася через непопулярність проблеми, через що вона була докладно розглянута тільки у найбільшому масштабі та на найважливіших рівнях.

Проте це питання може зацікавити представників великих і середніх компаній, оскільки дослідження критичності їх елементів, що представляються у вигляді посад та окремих співробітників, може виявити слабкі місця цих ОргС Усунення вразливостей може збільшити стійкість функціонування, що може мати ефект під час криз або надзвичайних ситуацій.

Метою цієї роботи є створення проведення аналізу стратегій функціонування компаній та підготовка теоретичної бази для створення інтелектуальної системи, здатної оцінювати ступінь критичності елементів, на основі прямих та непрямих даних, таких як:

- Особи, яким у тому чи іншому вигляді підпорядковується цей працівник
- Особи, які у тому чи іншому вигляді підпорядковуються даному працівнику
- Потоки, що проходять через цього працівника, такі як інформаційні, фінансові та ін.

- Обов'язки цього працівника
- Залежність інших посад від якості роботи, яку виконує даний працівник
- Витрати з підготовки та адаптації нових працівників на цю посаду

Об’єктом дослідження цієї роботи є структурний аналіз елементів ОргС

Предметом дослідження цієї роботи є методи експертного оцінювання критичності структурних елементів ОргС.

Не передбачається, що описана система надаватиме вичерпну аналітику. Вона розробляється саме як інструмент для допомоги менеджеру в аналітиці стійкості та потенційних слабких місць ОргС.

РОЗДІЛ 1. Аналітичний огляд організаційних структур та задачі визначення критичності їх об’єктів

1.1. Вибір та аналіз предметної області

Для цієї роботи, в якості демонстраційного прикладу, можна взяти абсолютно будь-яку компанію. В даному випадку, була обрана компанія FMCG сектору. Ця абревіатура розшифровується як «товари з високим оборотом» (Fast moving consumer goods). Також її можна інтерпретувати як «повсякденні товари» або «товари, розраховані на широкого споживача». Такі товари включають: продукти харчування, напої, напівфабрикати; косметику; предмети особистої гігієни; засоби для прання; ліки, що продаються без рецепта лікаря; електроніка, на кшталт лампочок та батареї; канцелярія та ін.

Виділяють п'ять основних відмінних рис цього сегмента ринку: [1]

- Висока оборотність товару — є основною характеристикою та говорить про високу частоту купівлі товарів цієї категорії;
- Відносно низький рівень чистого прибутку — товари цього сегмента являють собою товари з низькою маржинальністю, що означає низьку виручку з продажу одиниці товару через низьку собівартість;
- Високий рівень попиту — оскільки в цей сегмент входять товари першої необхідності, люди їх купуватимуть незалежно від ситуації на ринку;
- Низька залученість споживачів — через купівлю товарів цієї категорії практично кожен день, покупці не загострюють увагу на особливості товару і роблять вибір «за звичкою», купуючи товар компанії, якій вони довіряють;
- Простота замінності — через перенасиченість ринку та низьку різноманітність характеристик самого продукту, знайти аналог,

який не поступається ні за якістю, ні за ціною. Зазвичай, досить просто.

Низька залученість споживачів, простота замінності свідчать, що у разі втрати довіри, повернути покупців, які перейшли на продукт конкуруючої марки, буде досить складно. Так само відтік споживачів може спровокувати неможливість купити товар знайомої марки через зниження продуктивності.

Подібні умови ринку підвищують вплив КЕ на рентабельність системи загалом. Оскільки, у цьому сегменті ринку, короткочасна втрата функціональної повноти виробництва призводить до набагато серйозніших довготривалих наслідків, ніж у сегментах ринку, де товари мають більш високу маржинальність.

Компанії у сфері FMCG можна розділити на три групи, за кількістю типів товарів, якими вони торгують: [1]

- Монопродуктні
- Сфокусовані на 2-3 видах продуктів
- Мультипродуктні

Приналежність до однієї з цих груп впливає внутрішню структуру організації підприємства і її складність. Оскільки мультипродуктні фірми більш схильні до складної комбінації різних типів стратегій управління. Це робиться задля збільшення зручності управління структурою, спрямованої на задоволення різноманітних запитів клієнтів, що перебувають у різних фокус-групах.

1.1.1. Аналіз типів компаній у сфері FMCG

Процес продажу товару у сфері FMCG, за рідкісним винятком, поділяють на чотири рівні: [1]

1. Великі компанії. Їхніми завданнями є виробництво, просування та реклама;
2. Дистриб'ютори та компанії-склади. Перші займаються маркетингом продукції, і створюють клієнтську базу. Склади

лише зберігають вироби і передають їх роздрібним споживачам за запитом;

3. Роздрібні торгові точки. До цієї категорії входять, як мережеві магазини та супермаркети, так і індивідуальні підприємці. Суб'єкти цього рівня тісно взаємодію зі споживачем;
4. Кінцевий споживач. У сфері FMCG представлена фізичними особами, які купують товари для особистого користування. Саме на їхні потреби орієнтуються представники першого рівня, при розробці продукту.

Переважно використання описаної інтелектуальної системи орієнтоване на представників першого рівня, тобто на великі компанії. Проте вона все ще застосовна і корисна для членів другого класу. Для третього рівня вона майже не застосовна, але у випадку мереж супермаркетів все ще може бути корисною.

1.1.2. Аналіз потенційних стратегій організації компанії

Компанії в сфері FMCG, незалежно від їх рівня, є ієрархічною структурою побудованою по одному або комбінації кількох з наступних параметрів: [5]

- Географічне положення
- Категорії товарів
- Категорії клієнтів
- Функції збуту

Від вибору стратегії організації підприємства залежить складність аналізу опрацьованих даних. Географічна є найпростішою у всіх сенсах структурою. За такої організації виходить досить мала кількість рівнів ієрархії та відсутність переходу функціональних залежностей на сусідні гілки. На жаль, подібну структуру часто вибирають лише на великому масштабі, а в кожному регіоні організовують роботу за іншою стратегією. Також негативно впливає на популярність цієї стратегії стрімкий розвиток

інтернет-торгівлі, яка, по суті, нехтує віддаленістю кінцевого споживача від дистриб'ютора.

Стратегії розподілу за категорією товарів та категорією клієнтів, для даного завдання, не матимуть істотної різниці. Це відбувається через те, що обидві стратегії створюють досить гіллясту структуру, в якій гілки функціонально практично ізольовані одна від одної. Єдине, що їх пов'язує, так це використання спільних ресурсів, у разі розподілу за категоріями клієнтів, та спільні клієнти у разі розподілу за категорією товарів.

Стратегія розподілу за функцією збуту являє собою менш гіллясту у плані підпорядкованості структуру. Тим не менш, вона може виявитися однією з найскладніших для аналізу через сильне переплетення функціональних залежностей. Це відбувається через те, що виконані завдання одного відділу не є, по суті, завершеними, а лише підготовка матеріалів для початку роботи наступного відділу.

Також у більшості великих компаній є окремі співробітники, які займаються «ключовими» клієнтами. Під «ключовими» маються на увазі великі клієнти, як правило, великі дистриб'ютори.

1.1.3. Аналіз різних типів підпорядкованості

У компаніях, незалежно від сфери діяльності, може існувати два типи керівництва:

- Адміністративне - керівник підрозділу, який відповідає за загальну організацію діяльності співробітників підрозділу, за обсяг поставлених завдань підлеглим та їх результати, а також за виконання працівниками підрозділу всіх правил (вимог), прийнятих на підприємстві.
- Функціональне – керівник, який організовує діяльність працівника, який виконує функцію у зоні компетентності даного керівника. Функціональний керівник не володіє ресурсом якогось

часу співробітника, він змушений отримати цей ресурс від адміністративного керівника.

Найчастіше, функціональний та адміністративний керівники — це одна людина, але в деяких випадках ці посади можуть бути розділені. Подібне явище саме собою може погано позначатися на ефективності роботи підлеглих, але цей аспект у роботі не розглядається. Тим не менш, типи керівництва розрізняються за навантаженням і обов'язками, а отже система повинна як мінімум передбачати можливу різницю між функціональним і адміністративним керівниками.

1.2. Аналіз підходів рішення до задачі визначення критичності елементів організаційної системи

1.2.1. Загальний огляд підходів

Є два можливих підходи до рішення поставленої задачі:

- Аналітичний
- Експертний

Аналітичний підхід має під собою деякий алгоритм або формулу, за якою підраховується критичність елементів системи. Такий підрахунок може базуватися на кількості підлеглих, впливу на фінансові та інформаційні потоки, тощо. Перевагою такого підходу є зрозумілість витоків результату та відсутність необхідності будь кого, окрім людини, яка введе Дані та запустить алгоритм. До недоліків входить неповність огляду. Якщо в формулі не враховується деякий параметр, специфічний для конкретно взятої галузі, або його характеристика не повна, то результати підрахунків можуть бути дуже далекими від істини.

Експертний метод виступає антиподом аналітичного. Він потребує участі кваліфікованих експертів, які розуміють специфіку устрою галузі, та конкретно взятої системи. Але це компенсується тим, що експерти можуть провести більш комплексну та розгорнуту оцінку. Також, через участь великої кількості експертів, майже повністю виключається людський фактор,

так як малоймовірно, що велика кількість експертів допустить одну й ту саму помилку оцінки. Він не є ідеально точним, так як при створенні питань поставлених експертам також може бути навмисно, або випадково зроблено акцент на деяких атрибутах системи.

В свою чергу, великі ОргС відрізняються великою складністю та взаємопов'язаністю елементів. Через це дуже складно розробити аналітичний метод, який би надійно та точно проводив оцінку КЕ.

У випадках, коли потрібно проводити автоматичний аналіз деяких об'єктів але виявити зв'язок між елементами стає занадто складно, використовують нейронні мережі. Нажаль в даному випадку їх застосування не є можливим через потребу в великій кількості навчальних даних, які дуже складно збирати.

Саме тому для вирішення задачі визначення критичності елементів ОргС використовують експертний метод.

1.2.2. Приклади використання експертного методу для визначення критичності елементів

Перш за все, слід зазначити, що навіть роботу менеджера з персоналу, який оцінює яких людей ще треба найняти, або які посади створити можна вважати процесом рішення задачі визначення критичності елементів за допомогою експертного методу. Але це також не повноцінний приклад, так як в даному випадку приймає рішення лише один експерт.

Випадки, в яких до цієї задачі підходять більш комплексно, з'являються лише при підвищенні ціни помилки до масштабів країн. Так у більшості держав світу є свої критерії визначення критичності об'єктів інфраструктури.

На їх прикладі можна помітити суб'єктивність такої оцінки. При складені критеріїв оцінки перевага віддається конкретним аспектам елементів. Так, наприклад, в Ізраїлі серед параметрів є «символічна (ідеологічна, історична або культурна) значущість об'єктів»[2]. Через це

музеї, пам’ятки та інші об’єкти культурної спадщини віднесені до найбільш критичних для країни. В цей час, більшість інших країн, наприклад, США та країни ЄС мають більш прагматичний підхід та до найважливіших відносять три категорії об’єктів: «життєво важливі (АЕС, великі гідроспоруди та ГЕС, сховища стратегічних запасів нафти та газу, небезпечні хімічні та нафтохімічні виробництва, сховища ядерних матеріалів та боєприпасів); вкрай важливі (великі системи енергозабезпечення, метрополітен, мережі водопостачання та каналізації, магістральні трубопроводи); важливі (морські порти, очисні споруди, магістральні автомобільні та залізничні дороги, великі аеропорти, центри зв’язку тощо)» [2].

Також використання найрозвиненішими державами світу експертного методу для оцінки рівня критичності об’єктів інфраструктури підтверджує його ефективність для вирішення поставленої задачі.

1.3. Постановка задачі

1.3.1. Загальний огляд застосунку

Інтелектуальна система призначена для допомоги в аналізі надійності, та виявленні вразливостей в орг.с. за допомогою експертної оцінки. Потенційні групи користувачів та їх взаємодії с системою представлені на рис 3.1. З системою взаємодіють три типи сутностей:

- Користувач
- Експерт
- Адміністратор

До можливостей користувача входить можливість створення проекту, що включає в себе введення всіх необхідних даних та його запуск. Також саме він буде одержувати кінцевий результат.

Експерт підключається на створене опитування та проводить оцінку, базуючись на даних про організацію представлену користувачем.

Адміністратор регулює взаємозв’язок користувачів з системою, базуючись на інформації яку система йому предоставляє.

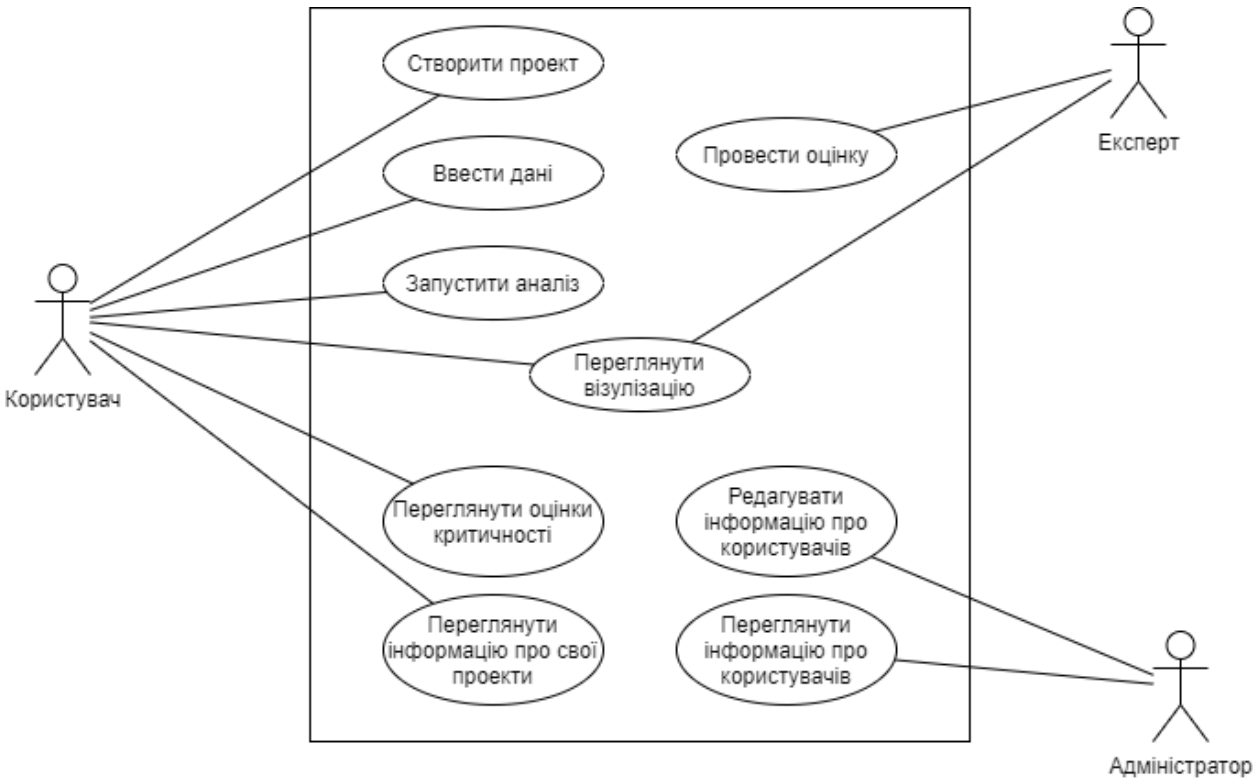

Рисунок 3.1. Діаграма прецендентів

До майбутньої системи можна висунути наступні вимоги.

Функціональні:

- візуалізувати введені Дані у вигляді графів;
- прийняти оцінку експертів;
- компілювати оцінки декількох експертів;
- формувати файл зі звітом.

Нефункціональні:

- повинен підтримуватися ввід у форматах excel;
- представлені візуалізація та оцінка повинні бути доступними для розуміння;
- система повинна підтримувати різні типи зв’язків між співробітниками;
- можливість роботи декількох користувачів паралельно;
- запрошувати експертів за допомогою листа на електронну пошту.

1.3.2. Опис вхідних даних

Позначимо як $A = \{a_1, \dots, a_n\}$ множину всіх працівників аналізованої області, яка може бути представлена компанією чи великим відділом певної компанії. При цьому кожен із зазначених працівників належить до однієї з множини посад цієї компанії. Позначимо цю множину як $B = \{b_1, \dots, b_m\}$, де $m \leq n$. Приналежність співробітника до конкретної посади визначається як R_1, відношення цих двох множин. Причому один співробітник може обіймати лише одну посаду.

$$\forall a \in A; \exists b \in B; \forall c \in B; c \neq b: aR_1b \Rightarrow a\widetilde{R_1}c$$

Кожній посаді відповідає оцінка витрат на заміну співробітника, що її обіймає, виражену у вигляді оцінки суми витрат із трьох типів джерел:

- Складність пошуку компетентного спеціаліста
- Тривалість періоду адаптації
- Ступінь зменшення продуктивності у період адаптації

Також у характеристики посад входить показник плинності кадрів, який характеризує оцінку ймовірності звільнення конкретно взятого співробітника із посади яку він обіймає.

Підпорядкованість працівників визначається відношеннями R_2 та R_3 ($R_2, R_3 \subset A \times A$), які описують функціональне та адміністративне керівництво відповідно. До того ж, у кожного співробітника може бути лише один функціональний та один адміністративний керівники. Ці дві посади можуть збігатися.

$$\forall a \in A; \exists b, c \in A; \forall d \in A; d \neq a, b, c: aR_2b \wedge aR_3c \wedge a\widetilde{R_2}d \wedge a\widetilde{R_3}d$$

Функціональні обов'язки підприємства визначається множиною завдань, які потрібні виконувати співробітники компанії, $C = \{c_1, \dots, c_k\}$. Обов'язки є унікальними, тобто $C^{i_1} \cap C^{i_2} = \emptyset, i_1, i_2 \in \{1, \dots, k\}$.

Кожній посаді відповідає перелік функцій, які представники цієї посади мають виконувати. Ця залежність описується відношенням R_4.

$$\forall a \in B; \exists b \in C: aR_4b$$

Також у функції можуть бути їй попередні та наступні, що виражаються відношенням R_5.

$$\exists a \in C; \exists b \in C: aR_5b$$

У характеристики функції можуть входити різні типи потоків, оперування якими необхідно для виконання цієї функції.

Варто зазначити, що перелік посад складається за характером робіт, що виконуються даним робітником, а не за підпорядкованістю тим чи іншим особам. Таким чином, представники однієї посади можуть мати різних керівників, але не можуть мати різних функціональних обов'язків.

Всі з перелічених вище відношень не мають не одну з властивостей теорії множин (симетрія, рефлективність, транзитивність тощо).

1.4. Висновок до першого розділу

У ході роботи над першим розділом дипломної роботи було обрано предметну область та проведено дослідження актуальності задачі визначення критичності елементів орг.с. Детально розглянуто специфіку функціонування компаній в сфері FMCG.

Проаналізовано сучасні підходи до вирішення задач визначення критичності елементів систем. Описано приклади використання методів експертного оцінювання для висвітлення КЕ на рівні світових держав.

Чітко окреслені вхідні дані задачі, сформульовано перелік функціональних та не функціональних вимог до ІС, визначено ролі доступу до веб-застосунку.

РОЗДІЛ 2. Проектування системи визначення критичності елементів організаційних систем

2.1. Проектування архітектури системи

2.1.1. Визначення типу системи

Перш за все треба визначити чи буде програмна реалізація інтелектуальної системи представлена у вигляді мобільного додатку, додатку для персональних комп'ютерів або веб-застосунком. Тому розглянемо переваги кожного з них в таблиці 2.1.

Таблиця 2.1. Переваги різних типів застосунків

Мобільний додаток	Додаток для ПК	Веб-застосунок
Портативність	Працює офлайн	Контроль користувачів
Працює офлайн	Середня потужність	Швидкий доступ
		Велика потужність
		Портативність

Тепер розглянемо кожну з переваг та її важливість для проекту.

Портативність застосунку, як і можливість працювати офлайн є гарними характеристиками, але не планується використання ІС в повсякденному житті. Саме тому вони не є важливими для проекту.

Даний застосунок не вимагає дуже великих потужностей, тому персонального комп'ютера для виконання задач повинно вистачити, а тим паче сервера.

Через епізодичність необхідності створення візуалізації орг.с. та визначення критичності елементів дуже гарною є можливість швидкого доступу. Також це полегшить доступ до системи для експертів, які раніше не працювали з нею.

Контроль користувачів є гарним доповненням до попереднього пункту. Тобто, це полегшить потенційну монетизацію структури, а також дозволить

покращити швидкодію системи за рахунок збору та аналізу звітів використання.

Таким чином, найбільш доречним буде представити інтерфейс до інтелектуальної системи у вигляді веб-додатку.

2.1.2. Функціональний аналіз

Тепер для конкретизації вимог до модулів проекту та більш чіткого представлення особливостей реалізації побудуємо дерево функцій (рис 2.1), де у вершині буде веб-додаток з визначення критичності об’єктів орг.с. Функції можна поділити на клієнтські, адміністративні та експертні.

Основна частина клієнтських функцій — це керування своїми проектами, а саме: створення та видалення. Також повинна бути можливість переглянути результати обрахунків. Для компенсації недоліку веб-додатку про необхідність доступу к інтернету додаймо можливість завантажувати результати обробки, а також зберігати в хмарному сховищі. Варто зазначити, що вигідніше буде зберігати результати обробки, а не початкові дані. Такий підхід дозволить досягти ідемпотентності застосунку, тобто не потрібно буде обробляти Дані при кожному перезавантаженні сторінки. Система просто буде відправляти вже готові результати.

Функціями які доступні експертам є лише перегляд візуалізації орг.с. та проведення оцінки критичності елементів на її основі.

До адміністративних функцій відносяться різноманітні можливості керування акаунтами користувачів. Наприклад перегляд даних користувачів, які мають включати: логін, пароль, контакти та кількість створених проектів. Процес реєстрації нових користувачів або видалення вже існуючих проводиться саме через адміністраторів. Також повинна бути можливість призначати нових адміністраторі та не повністю видаляти акаунт, а тимчасово його заблокувати, зі збереженням всіх даних.

Рисунок 2.1. Дерево функцій

Далі, для конкретизації процесу роботи з додатком, представимо його функції у вигляді процесів за допомогою VAD діаграми, зображеної на рисунку 2.3.

На схемі видно, що весь процес роботи с застосунком можна поділити на три частини. Першою є авторизація, так як неавторизовані користувачі не мають доступу до функцій додатку. Далі процес можна розбити на два підпроцеси які йдуть паралельно та визначається рівнем доступ.

У випадку, коли людина авторизувалася як простий користувач, вона матиме доступ до створення нового проекту. Це включає в себе завантаження даних про структуру організації та задання даних експертів, а саме їх пошти, що система могла з ними зв’язатися. Наступним кроком є проведення оцінки експертами на основі візуалізації структури системи. Далі користувач зможе переглянути результати праведної оцінки та завантажити їх. Коли дані йому будуть не потрібні він зможе видалити його зі свого профілю.

Діяльність адміністраторів в цей час представляє з себе процеси створення, налаштування та видалення акаунтів користувачів. Все це правдиться на основі перегляду інформації про користувачів.

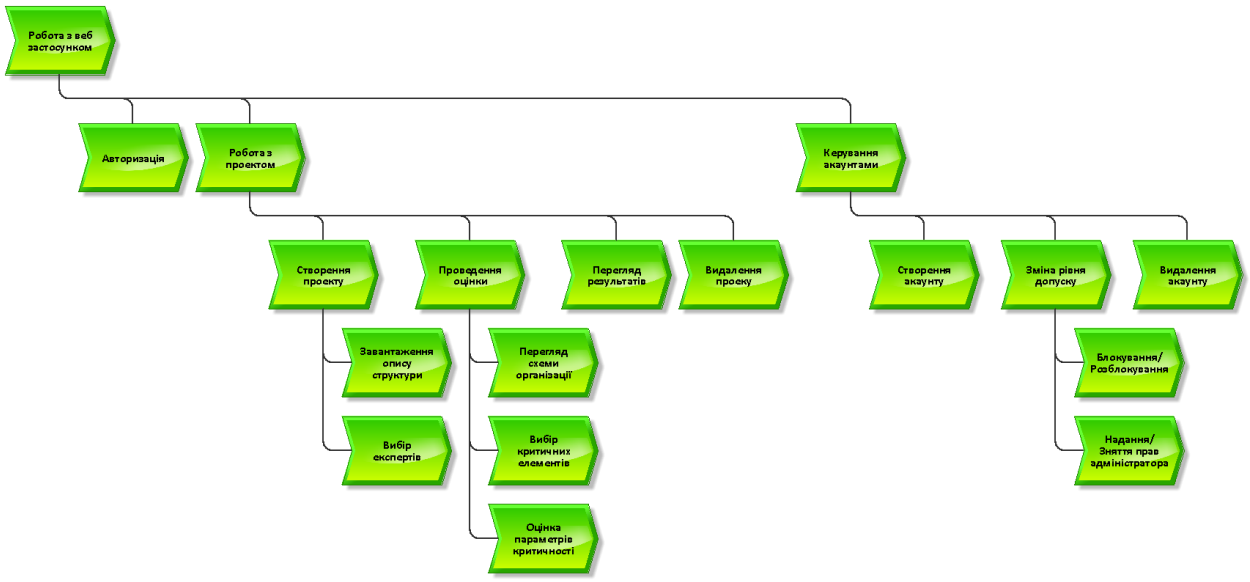

Рисунок 2.3. VAD діаграма використання веб-застосунку

2.1.3. Модулювання процесів

Для більш чіткого представлення процесу роботи з застосунком зобразимо його на Event-Driven Process Chain діаграмі (рис 2.2), яка починається з самого відкриття сайту та проходить по всім етапам роботи з проектом, окрім видалення.

На початку користувач авторизується на сайті. У випадку якщо він є адміністратором, його буде перенесено на адмін-панель, що в даній схемі не розглядається. Далі користувач створює переходить на сторінку створення нового проекту та вводить всі необхідні дані. Далі іде етап обробки, на якому створюється візуалізація структури проекту та відправляються запрошення експертам. Наступним є проведення опитування експертів. Після чого результати агрегуються та виводяться початковому користувачу.

Також, як вже було зазначено вище, користувач зможе скачати звіт про отримані результати, в якому буде представлена візуалізація системи та агреговані оцінки експертів.

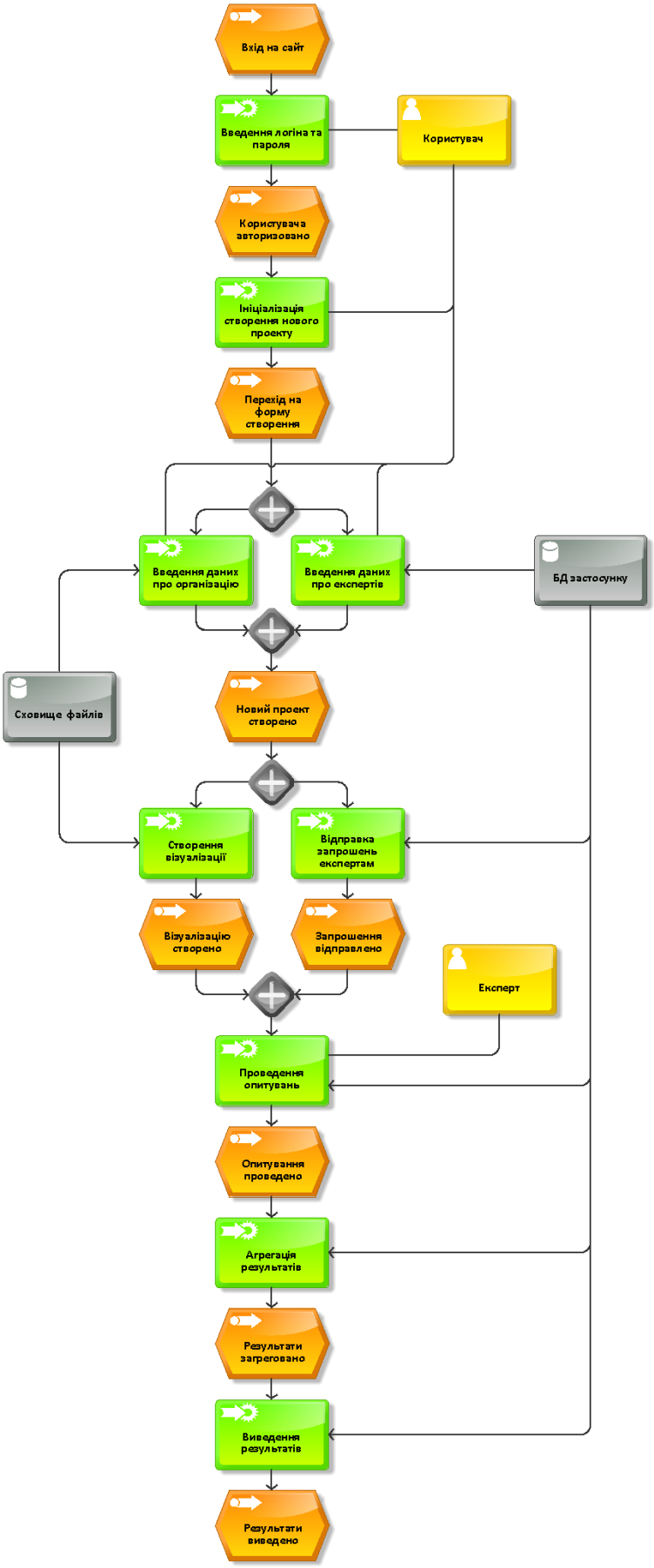

Рисунок 2.2. Event-Driven Process Chain діаграма взаємодії користувача з веб-додатком

Формалізуємо зображений процес та представимо його у вигляді діаграми в нотації IDEF0. Перш за все побудуємо контекстну діаграму (рис. 2.4).

Вхідними даними системи є файл, який описує структуру організації, перелік КЕ та оцінка параметрів критичності від різних експертів, а також логін та пароль. На виході система повинна повертати візуалізацію структури організації та оцінку критичності її елементів.

Механізмами є користувач та експерти. Об’єктами керування для системи виступають вагові коефіцієнти, згідно з якими проводиться агрегація оцінок різних експертів, та закон України «Про захист персональних даних», так як на всіх етапах роботи виконується обробка в тій чи іншій мірі конфіденційних даних.

Рисунок 2.4. Контекстна діаграма IDEF0 веб-додатку

Наступним кроком буде проведення декомпозиції описаної вище схеми (рис 2.5).

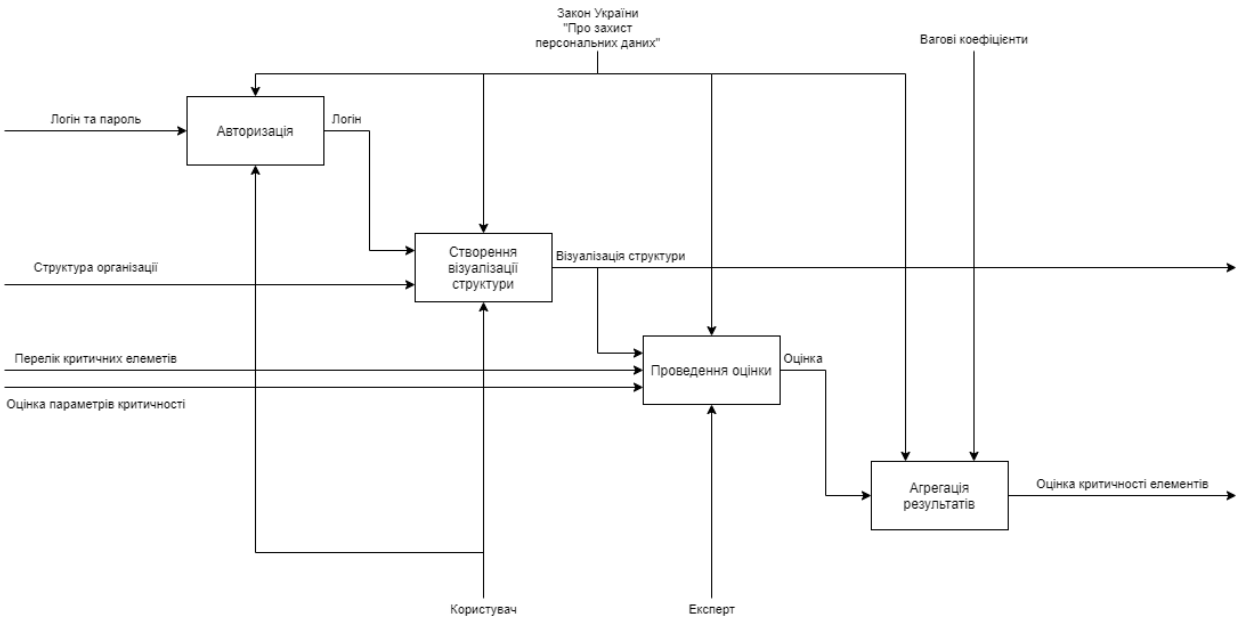

Рисунок 2.5. Декомпозиція IDEF0 першого рівня

Першим етапом є авторизація, яка дозволить користувачу отримати доступ необхідних функцій системи. Цей процес далі передає ім’я користувача, яке необхідне для коректного збереження файлів в системі.

Наступним етапом є обробка даних про структуру організації для створення візуалізації, на основі якої експерти зможуть зробити висновок про критичність тих чи інших її елементів. Також результати візуалізації йдуть на вихід системи, щоб користувач міг сам переглянути її.

Далі проводиться процес опитування експертів, які були запрошені до проекту за допомогою листів на електронну пошту.

Після проведення оцінок дані від різних експертів агрегуються та виводяться користувачу.

2.1.4. Архітектура веб-додатку

На основі проведеного аналізу можна виділити перелік модулів розроблюваного веб-додатку (рис 2.6).

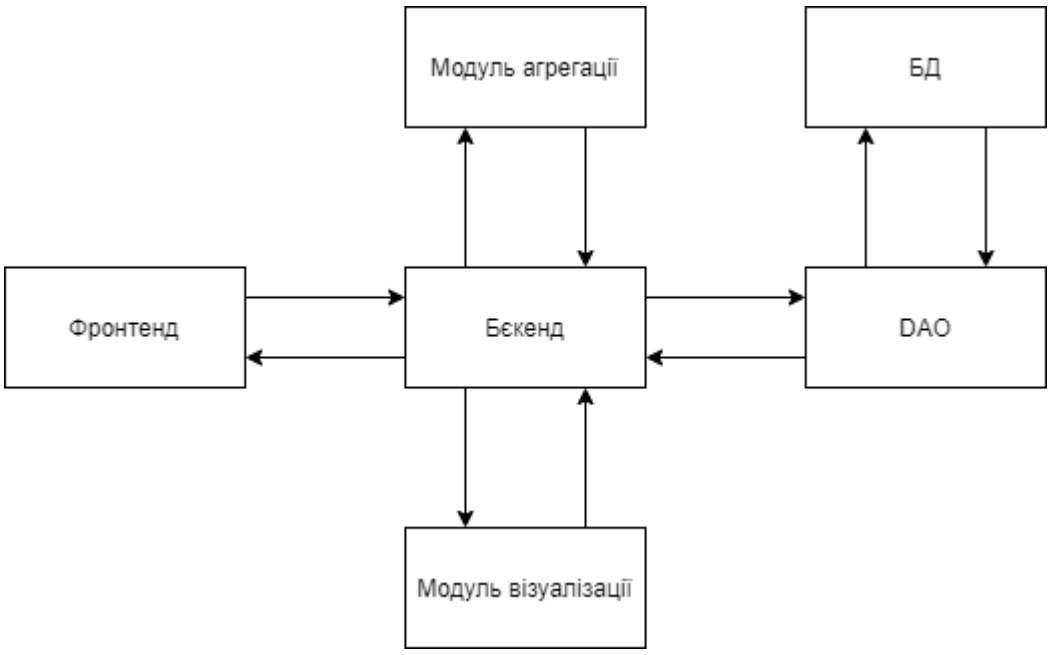

Рисунок 2.6. Архітектура додатку

Основними модулями є:

- Фронтенд — включає в себе різноманітні сторінки додатку та їх оформлення;
- Бєкенд — підтримує обіг даними та керує іншими процесами;
- БД — сховище даних, проектування якого було проведено вище;
- DAO (Data Access Object) — організовує взаємодію з базою даних, на основі запитів, які було викладено вище;
- Модуль візуалізації — проводить візуалізацію структури орг.с.;
- Модуль агрегації — проводить обробку оцінок експертів.

2.2. Математичне забезпечення інтелектуальної системи з визначення критичності елементів організаційних систем

Для вирішення задачі визначення критичності елементів орг.с. застосуємо експертний підхід. Визначимо критерії оцінки та метод агрегації результатів.

Слід зазначити, що системи, представлені експертам для аналізу можуть налічувати десятки, а може і сотні посад. Тому заради облегшення

процесу оцінки можна розділити оцінку на два тури. Так перший тур буде ставити більш загальне питання, а саме назвати найкритичніші, на думку експерта, елементи без пояснення конкретних аспектів. Для урегулювання цього аспекту обмежимо кількість елементів, які може відмітити експерт 10% від загальної кількості елементів.

На основі голосування експертів відбираються КЕ для проходження в другий тур. Їх кількість, як і на попередньому кроці, дорівнює 10% від загальної кількості елементів. У випадку, якщо на границі поділу буде декілька елементів з однаковими оцінками, то кількість елементів буде збільшено, щоб вмістити всі пограничні елементи.

Всі елементи, що пройшли у другий тур тепер будуть оцінені за конкретними параметрами за бальною системою від 1 до 5, де 1 — це зовсім не впливає, а 5 — вплив критичний. Перелік параметрів наступний: [3, 4]

α_1 – вплив на ресурси системи;

α_2 – потоки, які контролює елемент;

α_3 – вплив на прийняття рішень в системі.

Слід зазначити, що цей список можу бути розширено в майбутньому, в цілях розвитку додатку.

Далі іде етап агрегування результатів оцінки. Для цього використаємо метод лінійної (адитивної) згортки. Цей метод полягає в визначенні зваженого середнього оцінок від різних експертів для кожного параметра, після чого підрахунок зваженого середнього різних середніх значень параметрів кожного елемента. На початку розвитку додатку виставимо вагу всіх критеріїв та всіх експертів однаковою. Таким чином загальна формула розрахунку критичності буде наступною:

$$k = \sum_{m}^{j=1} \left(\sum_{n}^{i=1} g_{ij} a_i \right) * b_j, \quad (2.1)$$

де

n — кількість експертів

m — кількість параметрів (в даному випадку 3)

g — оцінка параметра деяким експертом

a — вага даного експерта (в даному випадку a = 1/n)

b — вага даного параметра (в даному випадку b = 1/m = 1/3)

В даному випадку, через відомість деяких параметрів, можна спросити формулу до наступного вигляду:

$$k = \frac{\sum_{3}^{j=1} \quad \sum_{n}^{i=1} \quad g_{ij}}{3n}.$$

(2.2)

2.3. Інформаційне забезпечення системи з визначення критичності елементів організаційних систем

2.3.1. Аналіз інформаційних потоків

Для проведення аналізу інформаційних потоків побудуємо Data Flow Diagram (в подальшому DFD діаграма). Для початку побудуємо контекстну діаграму (рис 2.6), на якій буде зображено взаємодію системи з зовнішніми сутностями.

З системою взаємодіють три зовнішні сутності, а саме користувач, експерт та адміністратор. Користувач відправляє логін та пароль для авторизації. Також, він може відправляти дані про організацію та експертів для створення проектів. У відповідь він отримує дані про раніше створені проекти та результати обробки відправлених даних.

Взаємодія експертів з системою зводиться лише до двох дій. А саме, отримання візуалізації орг.с. та відправки оцінки критичності елементів створеної на її основі.

Адміністратор, в свою чергу, також відправляє логін та пароль для авторизації. Крім цього він повинен отримувати інформацію про клієнтів та їх проекти, на основі якої від буде приймати рішення про відправку в систему команд для управління акаунтами.

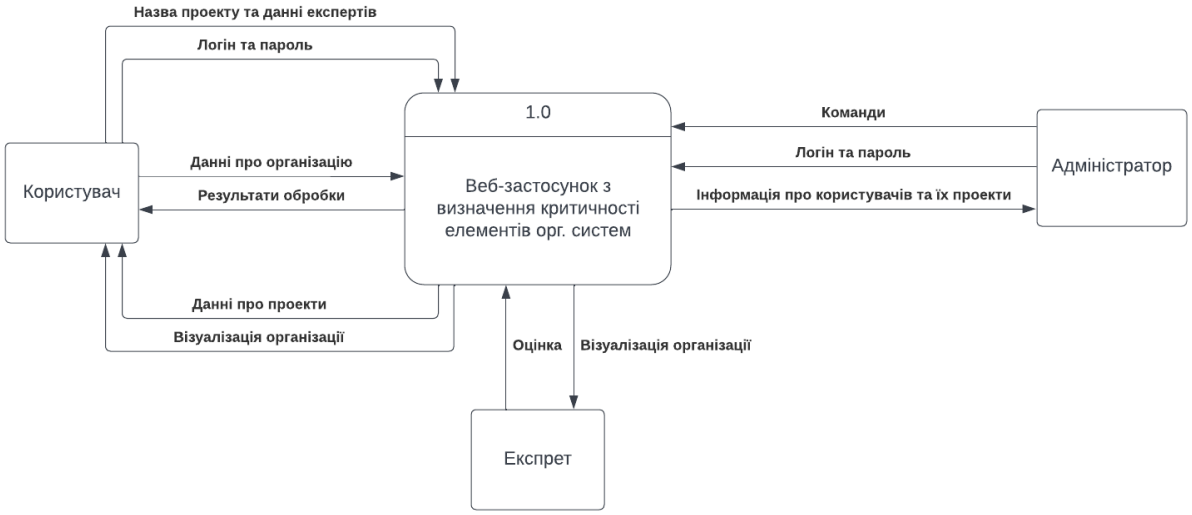

Рисунок 2.7. Контекстна DFD діаграма

Проведемо декомпозицію побудованої DFD діаграми (рис 2.7). Для зручності сприйняття, розташуємо всі вхідні потоки зліва, а всі вихідні справа. Також об’єднаємо вхідні потоки логіну та паролю від користувача та адміністратора в один потік даних, так як вони представляють однотипні дані для одного процесу авторизації. Теж саме зробимо і з вихідними потоками візуалізації організації, за тієї ж причини.

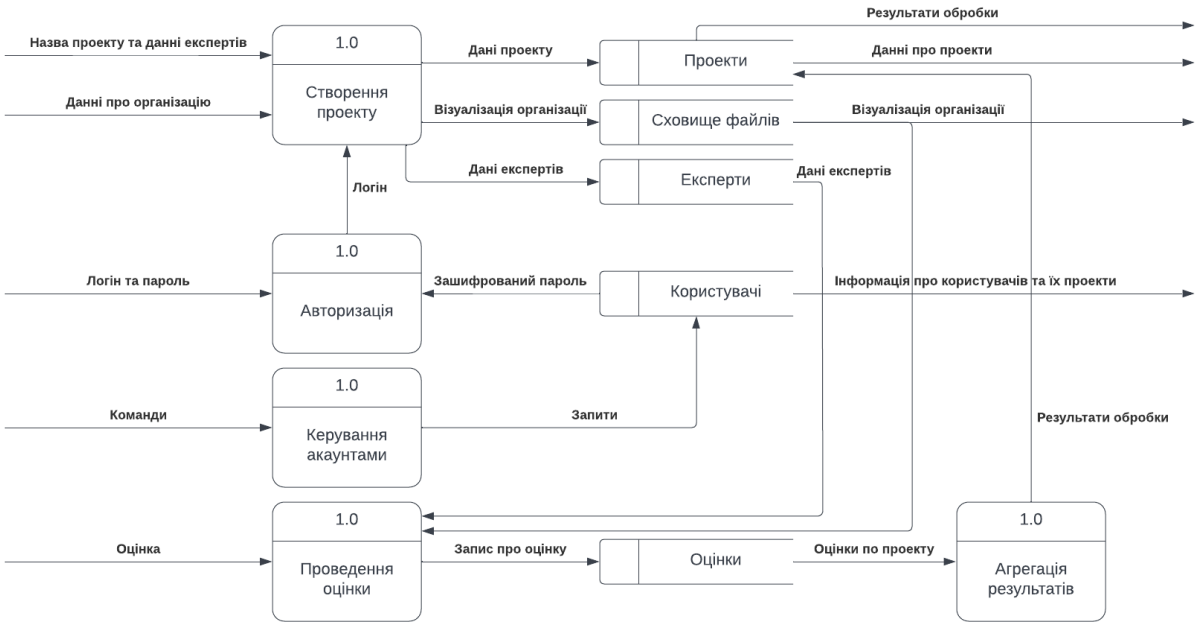

Рисунок 2.8. DFD діаграма першого рівня

Логін та пароль поступають на процес авторизації. В процесі вони порівнюються з зашифрованими даними з бази даних користувачів. Після цього логін користувача передається для подальшої роботи, зокрема збереження даних на ім’я користувача. Далі система приймає дані про організацію та експертів і після поучення логіну від процесу авторизації починає процес створення нового проекту та створення візуалізації. На виході ми маємо дані про створений проект, візуалізацію орг.с. та дані експертів. Всі перелічені дані зберігаються та йдуть на етап проведення оцінки.

Результати оцінки критичності зберігаються та, коли всі експерти залишать свою голоси, йдуть на етап агрегації. Результат цього етапу зберігається та виводиться користувачу.

Взаємодія з системою адміністраторів теж проходить через авторизацію, але після цього вони можуть переглядати дані про користувачів та їхні проекти, які були з агреговані з баз даних користувачів та проектів. Також вони можуть відправляти команди про зміни в базі даних користувачів, які були описані вище.

2.3.2. Розробка бази даних

Коли всі потоки даних визначено, ми можемо спроектувати БД для зберігання цих даних. Перш за все створимо концептуальну модель (рис 2.8). На ній представлено п’ять сутностей сутності, а саме: користувачі, проекти, експерти, а також їх відповіді на різні етапи опитування.

Слід зазначити, що можна провести денормалізацію БД для пришвидшення запитів до неї, а саме додати до сутностей відповідей на тури опитувань атрибут, який вказує на проект, до якого належить це опитування. Але в решті решт було прийнято рішення, що в даному випадку денормалізація скоріше вповільнить роботу БД, через потребу в записі додаткового поля.

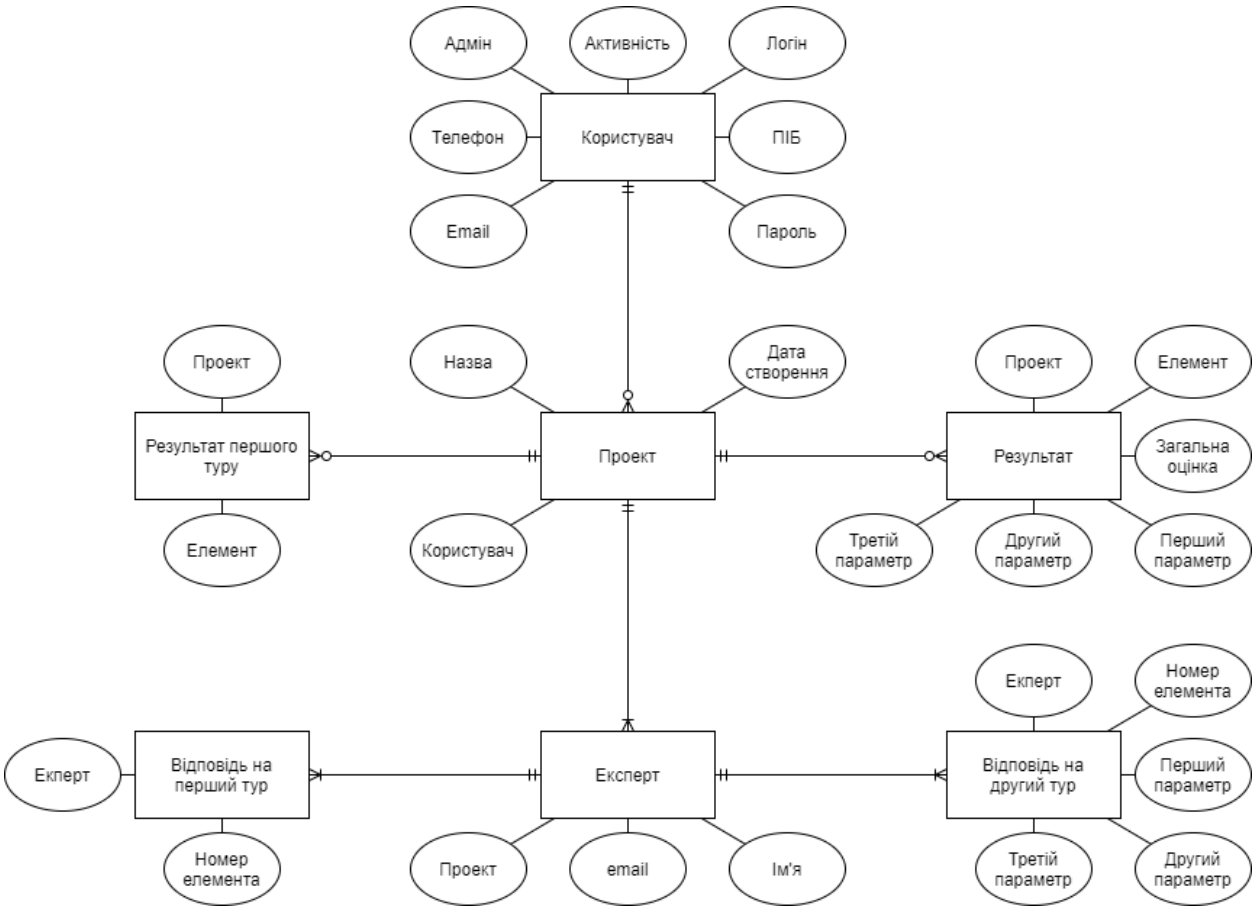

Рисунок 2.9. Концептуальна модель БД

Опишемо зв’язок між сутностями БД (табл. 2.2).

Таблиця 2.2. Зв’язки між сутностями

№	**Сутності, що утворюють зв’язок**	**Тип зв’язку**	**Пояснення**
1	Користувач — Проект	Один до багатьох (необов’язково)	Один користувач може мати декілька проектів, а може і не мати
2	Проект — Експерт	Один до багатьох	В проекті повинно бути декілька експертів
3	Експерт — Відповідь на перший тур	Один до багатьох	Експерт повинен дати декілька відповідей
4	Експерт — Відповідь на другий тур	Один до багатьох	Експерт повинен дати декілька відповідей

5	Проект — Результат першого туру	Один до багатьох (необов’язково)	У кожного проекту є результати по декільком елементам
6	Проект — Результат	Один до багатьох (необов’язково)	У кожного проекту є результати по декільком елементам

Деталізуємо визначені атрибути, визначивши їх типи та обмеження, а також ключі описаних сутностей (табл. 2.3). Також до вже описаних атрибутів додаймо id в таблиці проектів, експертів, відповідей та обох результатів для однозначної ідентифікації запису за одним атрибутом.

Таблиця 2.3. Визначення атрибутів сутностей

№ п/п	**Назва атрибуту**	**Тип атрибуту**	**Обов’язкове значення**	**Обмеження**	**Ключ**
Таблиця user «Користувач»					
1	login	Символьний	Так	<50 символів	ПК
2	password	Символьний	Так	<50 символів	
3	name	Символьний	Так	<100 символів	
4	email	Символьний	Так	<50 символів	
5	phone	Символьний	Ні	10 символів	
6	admin	Логічна змінна	Так	{0, 1}	
7	active	Логічна змінна	Так	{0, 1}	
Таблиця projects «Проекти»					
1	id	Ціле число	Так	>0	ПК
2	name	Символьний	Так	<50 символів	
3	user	Символьний	Так	<50 символів	ЗК
4	creation_date	Дата	Так	DATE()	

Таблиця experts «Експерти»					
1	id	Ціле число	Так	>0	ПК
2	project	Ціле число	Так	>0	ЗК
3	email	Символьний	Так	<50 символів	
4	name	Символьний	Так	<100 символів	
Таблиця first_tour «Відповідь на перший тур»					
1	id	Ціле число	Так	>0	ПК
2	expert	Ціле число	Так	>0	ЗК
3	element	Ціле число	Так	>0	
Таблиця second_tour «Відповідь на другий тур»					
1	id	Ціле число	Так	>0	ПК
2	expert	Ціле число	Так	>0	ЗК
3	element	Ціле число	Так	>0	
4	first_param	Ціле число	Так	[1;5]	
5	second_param	Ціле число	Так	[1;5]	
6	third_param	Ціле число	Так	[1;5]	
Таблиця first_tour_results «Результати першого туру»					
1	id	Ціле число	Так	>0	ПК
2	project	Ціле число	Так	>0	ЗК
3	element	Ціле число	Так	>0	
Таблиця results «Результати»					
1	id	Ціле число	Так	>0	ПК
2	project	Ціле число	Так	>0	ЗК
3	element	Ціле число	Так	>0	
4	final	Ціле число	Так	[1;5]	
4	first_param	Ціле число	Так	[1;5]	
5	second_param	Ціле число	Так	[1;5]	
6	third_param	Ціле число	Так	[1;5]	

Тепер, коли ми маємо остаточний перелік атрибутів, побудуємо фізичну модель БД (рис 2.9).

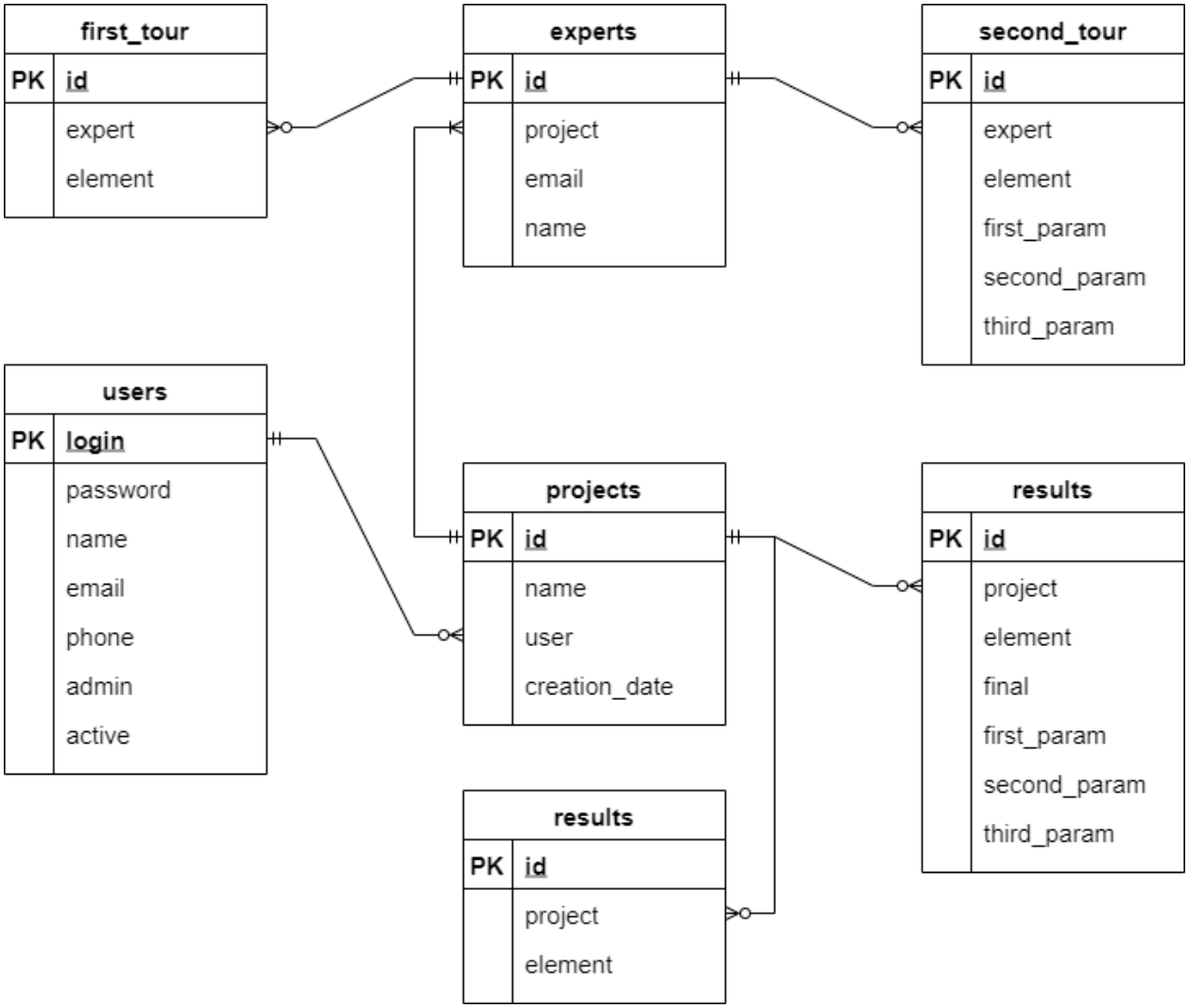

Рисунок 2.10. Фізична схема БД

2.3.3. Розробка запитів

Наступний крок це визначення та опис запитів до БД, які буде робити розроблюваний веб-додаток. Спочатку складемо словесний опис необхідних запитів для кожного DAO (табл. 2.4). Зазвичай їх кількість співпадає з кількістю сутностей БД. Якщо притримуватися цієї парадигми, у нас буде 6 об’єктів:

- UsersDao
- ProjectsDao
- ExpertsDao

- FirstTourDao
- SecondTourDao
- ResultsDao

Таблиця 2.4. Опис запитів до системи

№	Опис	Тип запиту
UserDao		
1	Знайти дані користувача для авторизації	SELECT
2	Вивести інформацію про всіх користувачів	SELECT
3	Встановити статус активності	UPDATE
4	Встановити статус адміністратора	UPDATE
5	Оновити інформацію користувача	UPDATE
6	Додати користувача	INSERT
7	Видалити користувача	DELETE
ProjectDao		
8	Створити новий проект	INSERT
9	Вивести дані проекту	SELECT
10	Повернути перелік експертів проекту та їх оцінки	SELECT
11	Перевірка чи завершено оцінку проекту	SELECT
12	Повернути посилання на візуалізацію проекту	SELECT
13	Видалити проект	DELETE
ExpertDao		
14	Додати експерта на проект	INSERT
FirstTourDao		
15	Додати оцінку першого туру	INSERT
SecondTourDao		
16	Додати оцінку другого туру	INSERT
ResultDao		
17	Додати результат проекту	INSERT

Таблиця 2.5. Об’єднання атрибутів

Назва атрибута	Об’єднання
users.login	користувач
users.password	
users.name	
users.email	
users.phone	
users.admin	активний
users.active	адмін
projects.id	проект
projects.name	
projects.user	
projects.creation_date	
expersts.id	експерт
expersts.project	
expersts.email	
expersts.name	
first_tour.id	перший тур
first_tour.expert	
first_tour.element	
second_tour.id	другий тур
second_tour.expert	
second_tour.element	
second_tour.effect	
results.id	результат
results.project	
results.element	
results.rating	

Таблиця 2.6. Використання атрибутів запитами

№ запиту	користувач	активний	адмін	проект	експерт	перший тур	другий тур	результат
Знайти дані користувача для авторизації	+	+	+					
Вивести інформацію про всіх користувачів	+	+	+	+				
Встановити статус активності		+						
Встановити статус адміністратора			+					
Оновити інформацію користувача	+							
Додати користувача	+							
Видалити користувача	+	+	+					
Створити новий проект	+			+				
Вивести дані проекту				+				
Повернути перелік експертів проекту та їх оцінки				+	+	+	+	+
Перевірка чи завершено оцінку проекту				+				+
Повернути посилання на візуалізацію проекту				+				
Видалити проект				+				
Додати експерта на проект					+			

Додати оцінку першого туру						+		
Додати оцінку другого туру							+	
Додати результат проекту								+

Наступним кроком буде визначення які атрибути буде використовувати кожен з перелічених запитів (табл. 2.6). Номера всіх запитів відповідні таблиці 2.3. Також, через те, що загальна кількість атрибутів занадто велика для відображення в таблиці, об’єднаємо ті, які використовуються разом (табл. 2.5).

Слід зазначити, що в даній моделі присутній ефект омонімії, тобто одна і та сама назва відповідає різним атрибутам, а саме user.name та project.name. Такий недолік не є критичним, але слід це мати на увазі при написанні запитів до БД.

2.4. Висновок до другого розділу

У ході роботи над другим розділом дипломного проекту було визначено тип системи, що проектується, проведено аналіз функцій, які повинен реалізовувати додаток. На основі цього було проведено планування загального алгоритму роботи, а також архітектури додатку.

Далі було проведено аналіз різних методів математичного вирішення задачі визначення критичності елементів орг.с. та сформовано алгоритм вирішення для цієї роботи.

Також було проведено аналіз потоків інформації та спроектовано структуру БД та запитів для покриття функцій додатку.

РОЗДІЛ 3. Програмна реалізація інтелектуальної системи з визначення рівня критичності елементів організаційних систем

3.1. Вибір інструментів реалізації

Для програмної реалізації ІС було обрано мову програмування Python [11]. Її простота та гнучкість дозволить полегшити процес написання коду. Також ця мова має доволі зручні методи побудови графів, які необхідні для візуалізації орг.с. В даному випадку було використано наступні бібліотеки:

- network — побудови та взаємодії з графами [12, 13];
- matplotlib — створення візуалізації даних [14];
- pandas — читання файлів користувача;
- numpy — обробка масивів даних.

Було розглянуто варіант написання бєкєнду також на Python, але врешті решт було прийнято рішення реалізувати цей модуль на більш об’єктно орієнтованій та швидкій мові, а саме на Java [6, 7]. Також для пришвидшення процесу розробки було використано фреймворк Spring [8], та його додаткові модулі для організації авторизації [9] та взаємодії з базою даних [10].

В якості візуалізатора Spring було обрано розширення HTML Thymeleaf [16-18]. Цей модуль дозволяє більш гнучко взаємодіяти з фронтендом та максимально зручно виводити дані користувачу.

Для структуризації коду бєкенду було використано декілька патернів програмування, а саме Front Controller та патерн Команда.

Front Controller — це патерн програмування, коли всі, або більшість запитів до серверу обробляє один і той самий клас, але сам їх не обробляє. Він делегує обов’язки іншим класам. Тобто Front Controller виконує функцію маршрутизації запитів і тільки її.

Патерн Команда — це петерн програмування, коли необхідні для виконання функції розбиваються на однотипні об’єкти, здатні виконувати

лише вузько поставлену задачу. Як правило, також створюється клас, який спеціалізується на зберіганні екземплярів цих команд та їх виклику.

3.2. Структура програмного забезпечення

В якості першого кроку зобразимо граф переходів між різними сторінками веб-додатку (рис 3.1). Слід зазначити, що з будь-якої сторінки можна потрапити на сторінки home та about, але це не було зображено на схемі, щоб не перенавантажувати її. Також не зображено входів в сторінку error, так як перехід на неї відбувається у випадку непередбачуваної ситуації, яка потенційно може статися в будь-якому місці додатку.

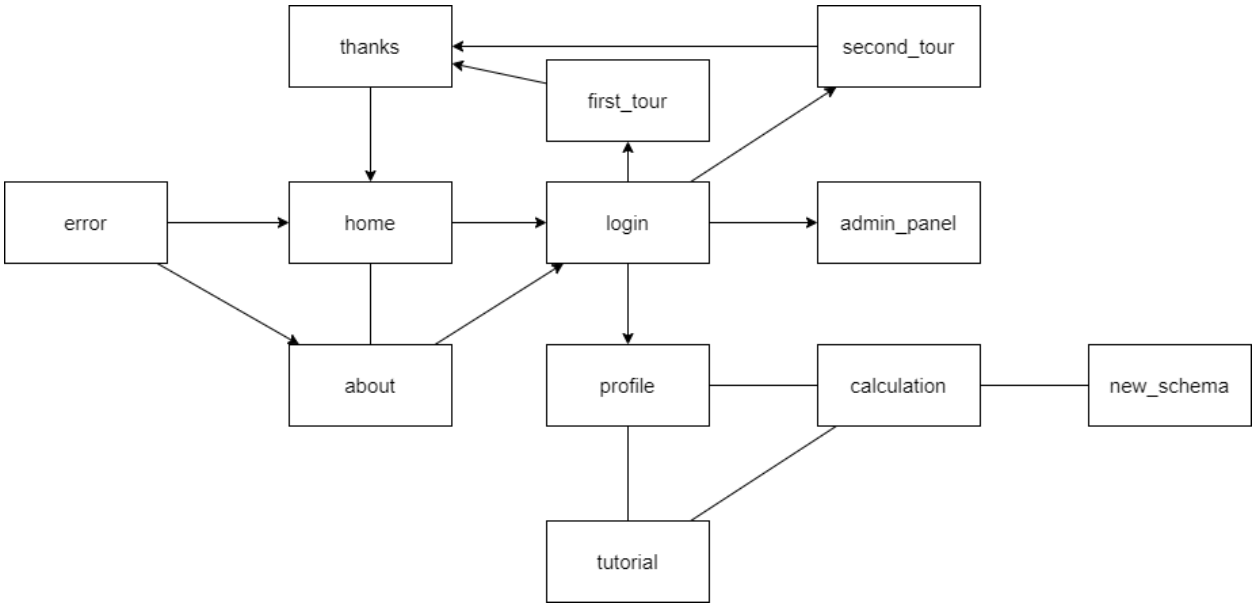

Рисунок 3.1. Граф переходів

Також слід зазначити, що всі описані сторінки відповідають однойменним html файлам, але в проекті приймає участь ще один файл — header. Він описаний окремо, є складовою частиною всіх інших файлів.

Тепер коротко опишемо кожну з перелічених на схемі сторінок (табл. 3.1)

Таблиця 3.1. Опис сторінок

№ сторінки	Назва	Опис

1	home	Домашня сторінка, на якій немає ніякої корисної інформації, окрім загального опису сайту
2	about	Сторінка про людей, які створили цей сайт та плани подальшої розробки
3	login	Сторінка входу в акаунт. Тут розташовані поля вводу логіну та паролю
4	admin_panel	Сторінка з інформацією про всіх користувачів та кнопками з їх адміністрування
5	profile	Сторінка з інформацією про всі проекти користувача та кнопками з їх адміністрування
6	tutorial	Сторінка з алгоритмом роботи з веб-застосунком
7	calculation	Сторінка відображення обчислень
8	new_schema	Форма вводу даних про новий проект
9	error	Резервна сторінка, яка повідомить користувача про помилку, та дозволить повернутися до роботи
10	first_tour	Сторінка проведення першого туру опитування
11	second_tour	Сторінка проведення другого туру опитування
12	thanks	Сторінка з подякою експерта за проходження опитування. Для варіанту сторінки після першого туру також повинно бути попередження по те, що опитування складається в двох турів

3.3. Приклад використання додатку

Для прикладу використання додатку пройдемо по процесам зображеним на Event-Driven Process Chain (рис. 2.2).

Перший пункт це авторизація користувача. Він потрапляє на сторінку введення логіну та паролю (рис. 3.2). Після введення сервер знаходить запис користувача в БД та порівнює його з введеними даними.

Рисунок 3.2. Сторінка авторизації

Якщо авторизація пройшла вдало то користувач потерпить в свій профіль (рис. 3.3). Тут відображені дані про проекти користувача. На зображені же продемонстрована тестова інформація для загального розуміння зовнішнього вигляду.

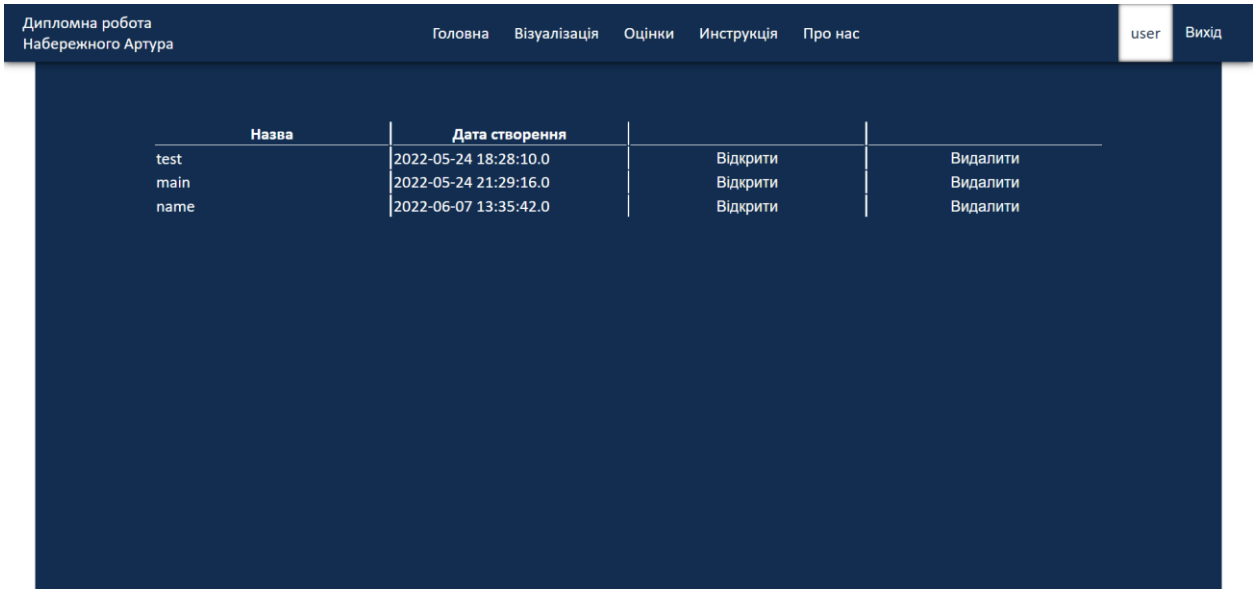

Рисунок 3.3. Сторінка профілю

Далі користувач може переглянути інформацію про вже створені проекти або перейти на нову вкладку «Розрахунки» (рис 3.4). На ній він може

натиснути кнопку «Новий проект», яка переправить його на форму створення нового проекту (рис. 3.5).

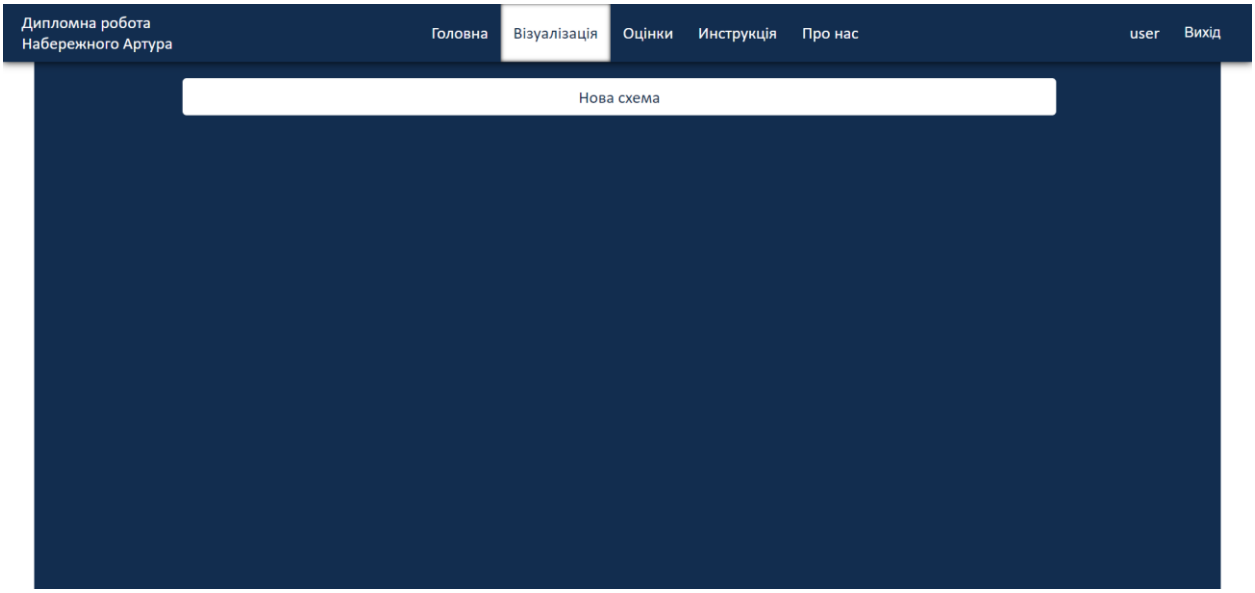

Рисунок 3.4. Сторінка розрахунків без проекту

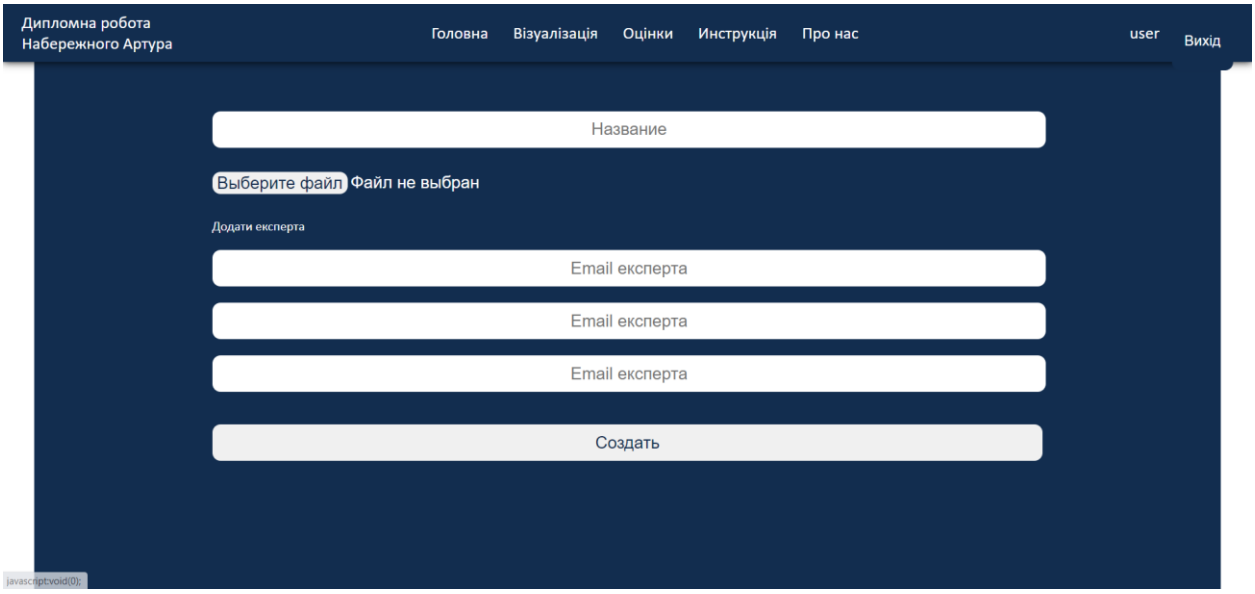

Рисунок 3.5. Форма створення проекту

Тут користувач вводить назву проекту, прикріпляє файл з описом структури, а також додає електронні адреси експертів, щоб система могла надіслати їм листи із запрошеннями на опитування.

Після цього в систему заходить експерт. Коли він переходить за посиланням із електронного листа, він потрапляє на сторінку проходження

першого туру голосування (рис. 3.6). На ній він може відмітити КЕ, на його погляд елементи.

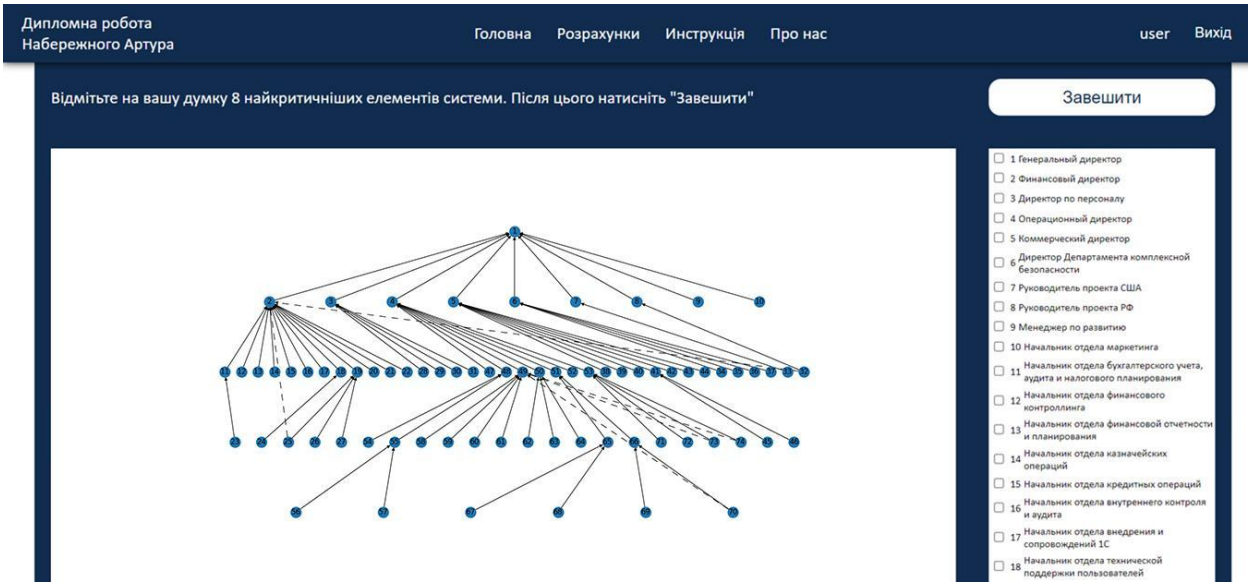

Рисунок 3.6. Форма опитування першого туру

Коли потрібна кількість елементів відмічена, експерт зможе натиснути кнопку «Завершити». Після цього він потрапляє на сторінку з подякою за участь в опитуванні та попередженням про те, що буде проведено другий тур (рис. 3.7).

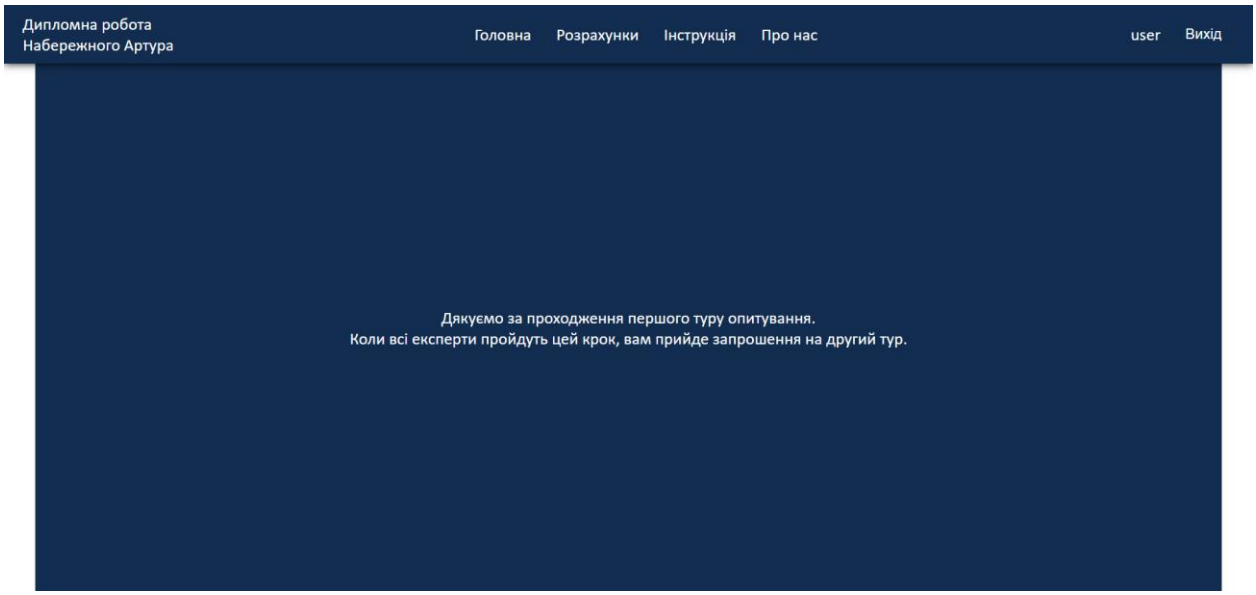

Рисунок 3.7. Сторінка з подякою після першого туру

Коли всі експерти пройдуть перший етап опитування, експертам прийде запрошення на проходження другого. Переходячи по посиланню, як в минулий раз вони потраплять на сторінку проходження голосування другого туру (рис. 3.8).

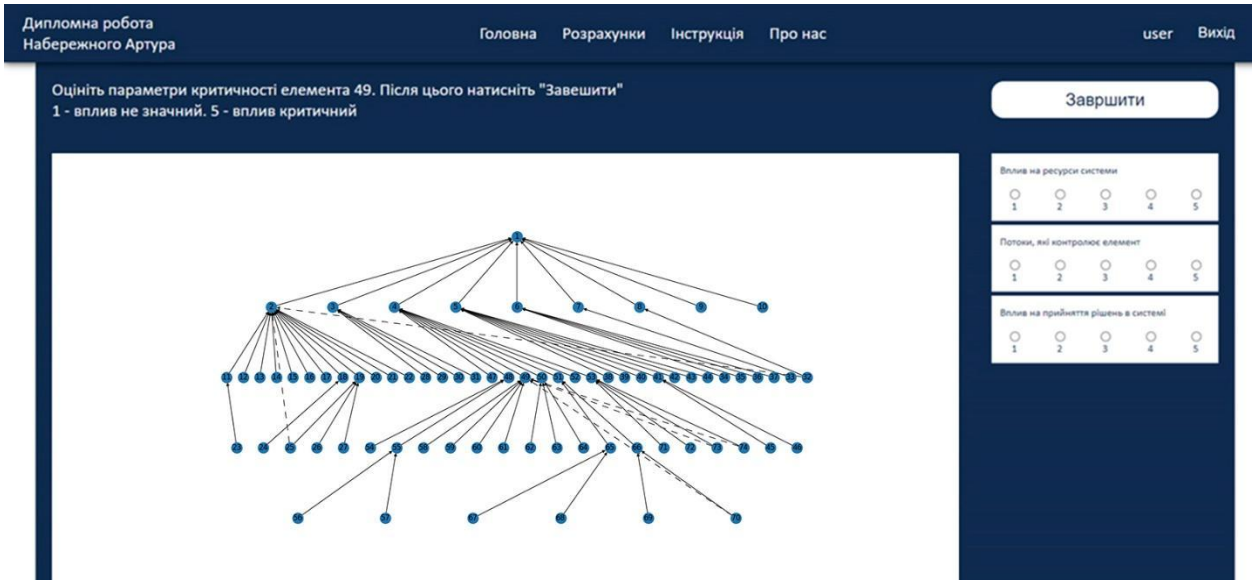

Рисунок 3.8. Форма опитування другого туру

На цій сторінці експерт вже оцінює ступінь критичності елементів за конкретними параметрами. Він відмічає оцінки для одного елемента, після чого переходить к наступному по натиску відповідної кнопки. Коли всі елементи оцінені, експерт потрапляє на сторінку з подякою, схожу на зображену на рисунку 3.7, але з трохи іншим повідомленням (рис. 3.9).

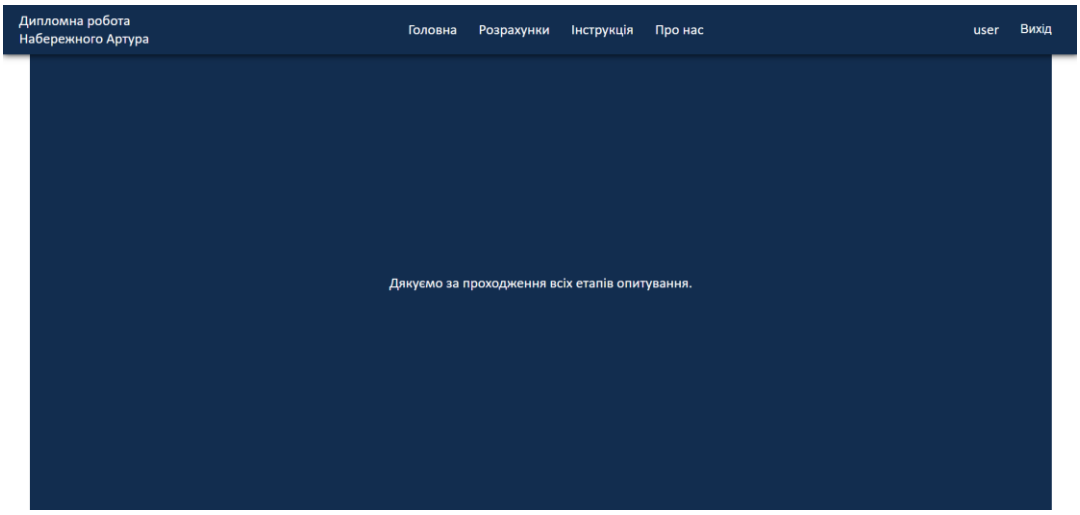

Рисунок 3.9. Сторінка з подякою після другого туру

Одразу після створення проекту користувачу буде доступна сторінка з візуалізацією проекту (рис 3.10, 3.11). Після проходження всіма експертами опитування стане доступна сторінка з результатами опитування (рис. 3.12), на якій приведено не тільки оцінки конкретних елементів, а і оцінки кожного з експертів.

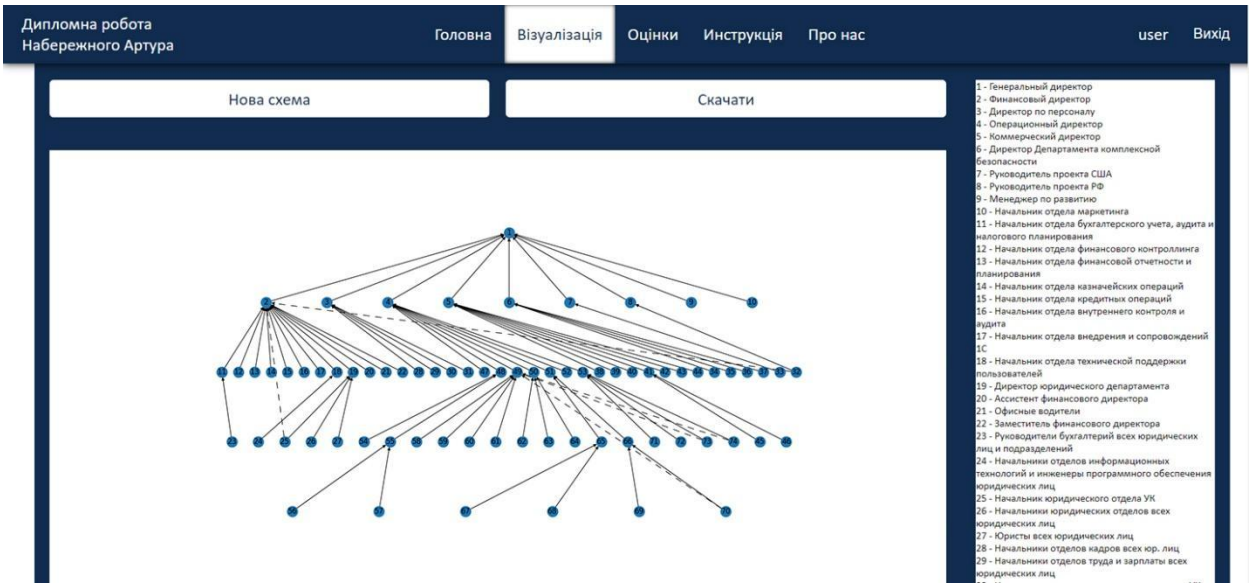

Рисунок 3.10. Сторінка перегляду візуалізації адміністративної структури

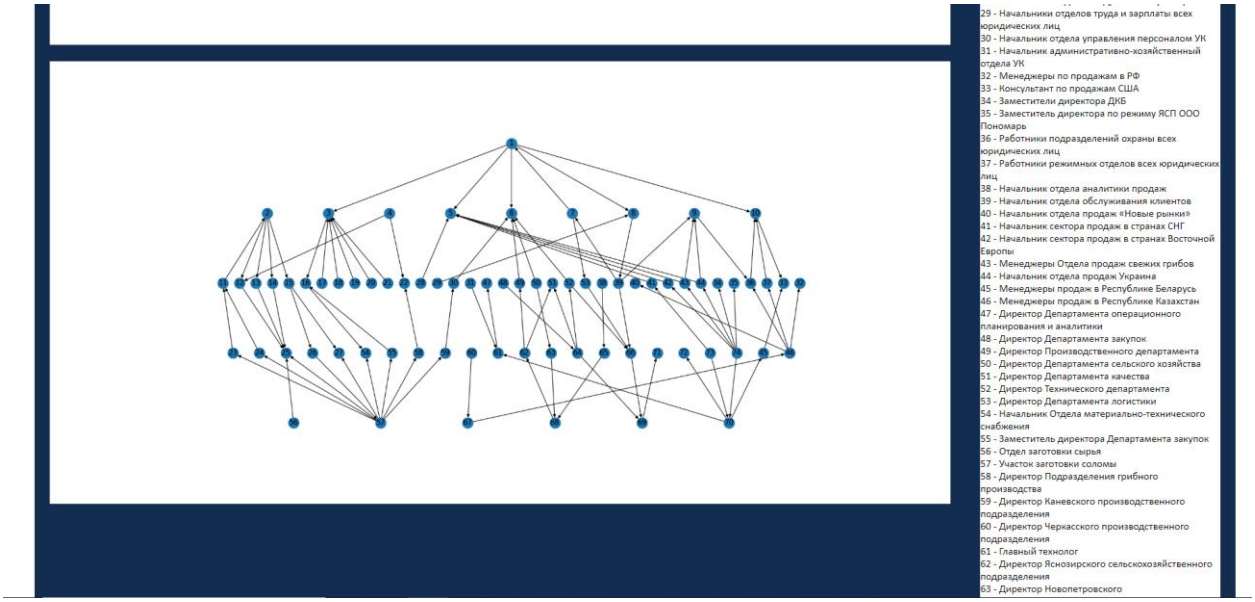

Рисунок 3.11. Сторінка перегляду візуалізації функціональної структури

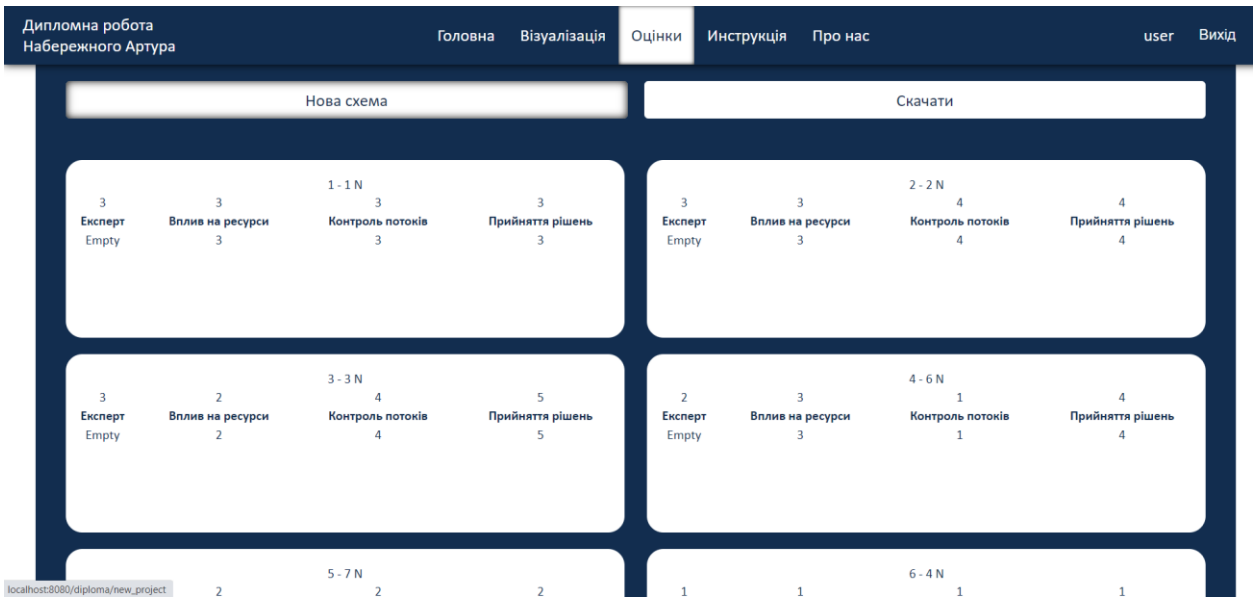

Рисунок 3.12. Сторінки перегляду результатів

В якості завершальної стадії користувач може звантажити результат у форматі pdf та видалити проект на сторінці профілю.

3.4. Висновок до третього розділу

Було обрано та аргументовано вибір інструментів для реалізації веб-застосунку для визначення критичності елементів орг.с. Описано паттерни, які було використано під час розробки Також були спроектовані сторінки веб-додатку, їх наповнення та логічний зв’язок між ними. Створено базу даних за попередньо розробленими схемами.

Проведено демонстрацію повного шляху користувача від створення нового проекту та введення даних до отримання кінцевого результату. Проведено демонстрацію оцінки експертами ОргС на наявність КЕ.

ВИСНОВОК

У ході виконання випускної кваліфікаційної роботи за темою «Інтелектуальна система визначення рівня критичності елементів організаційної системи» було окреслено такі три розділи основної частини:

1. Аналітичний огляд орг.с. та задачі визначення критичності їх об’єктів.
2. Проектування системи визначення критичності елементів орг.с.
3. Програмна реалізація інтелектуальної системи з визначення рівня критичності елементів орг.с.

Таким чином сформовано та виконано повноцінний проект за обраною темою. Доведено актуальність задачі для розгляду та дослідження сучасним науковим суспільством.

Зібрано та досліджено та проаналізовано інформацію стосовно специфіки структури орг.с. у сфері FMCG. Проведено аналіз сучасних підходів до задачі, а також чітко сформульовано задачу визначення критичності елементів орг.с., та різних підходів до її вирішення.

Проведено послідовне проектування усіх етапів подальшої розробки інформаційного та програмного забезпечень ІС у вигляді веб-застосунку.

Проведено якісну реалізацію інформаційного та програмного забезпечень ІС у вигляді веб-застосунку. Дотримано усіх поставлених функціональних та нефункціональних вимог. Програмна частина написана із використанням якнайбільш підходящих сучасних програмних продуктів та дотриманням обраних підходів до написання програмного коду.

Продемонстровано можливості веб-додатку як зі сторони звичайного користувача, так і зі сторони експерта.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Джемилева А. Что такое FMCG: типы товаров, особенности рынка и тенденции [Електронний ресурс] / Джемилева А. // 2021. – Режим доступу до ресурсу: https://timeweb.com/ru/community/articles/chto-takoe-fmcg
2. Бобро Д.Г. Методологія оцінки рівня критичності об’єктів інфраструктури / Бобро Д.Г. // Стратегічні пріоритети. – 2016. – № 3 (40). – С. 77–86.
3. Гнатієнко Г.М. Модель визначення критично важливих елементів складної слабкоструктурованої системи / Гнатієнко Г.М. // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: Збірник матеріалів доповідей та тез. – Київ, 2021. – С.147-148.
4. Babenko, T. Modeling of critical nodes in complex poorly structured organizational systems / Babenko, T., Hnatiienko, H., Ignisca, V. та ін. // Proceedings of the 26th International Conference on Information Society and University Studies. – Kaunas, Lithuania, 2021. – pp. 92–101.
5. Мясников А.В. Study Guide по изучению дисциплины «Теория организации» / Мясников А.В // Методическое пособие. – Москва, 2007. – 143 с.
6. Java Documentation [Електронний ресурс] // Reference Documentation. – Version 18.0.1.1, 2022. – Режим доступу до ресурсу: https://docs.oracle.com/en/java/
7. Руководство по Java EE/Jakarta EE [Електронний ресурс] // Справочная документация. – 2018. – Режим доступу до ресурсу: https://metanit.com/java/javaee/
8. Spring Framework Documentation [Електронний ресурс] / Johnson R., Hoeller J. Donald K. et al // Reference Documentation. – Version 5.3.21, 2022. – Режим доступу до ресурсу: https://docs.spring.io/spring-framework/docs/current/reference/html/
9. Spring Security Documentation [Електронний ресурс] / Alex B., Taylor L. // Reference Documentation. – Version 3.0.8.RELEASE. – Режим доступу до

ресурсу: https://docs.spring.io/spring-security/site/docs/3.0.x/reference/springsecurity.html

10. Spring JDBC Documentation [Електронний ресурс] / J.Schauder, J.Bryant, M.Paluch, B.Wilhelm // Reference Documentation. – Version 2.4.0, 2022. – Режим доступу до ресурсу: https://docs.spring.io/ spring-data/jdbc/docs/current/reference/html/
11. Python 3 Documentation [Електронний ресурс] // Reference Documentation. – Version 3.10.5. – Режим доступу до ресурсу: https://docs.python.org/3/
12. Hagberg A.A. Exploring network structure, dynamics, and function using NetworkX / A.A.Hagberg, D.A.Schult, P.J.Swart // in Proceedings of the 7th Python in Science Conference (SciPy2008). – Pasadena, CA USA, 2008. – pp. 11–15
13. Networkx Documentation [Електронний ресурс] // Reference Documentation. – Version 2.8.4, 2022 – Режим доступу до ресурсу: https://networkx.org/documentation/stable/index.html
14. Шабанов П.А. Научная графика в Python [Електронний ресурс] / Шабанов П.А. // Методическое пособие. – 2015. – Режим доступу до ресурсу: https://github.com/whitehorn/Scientific_graphics_in_python
15. Apache Tomcat Documentation [Електронний ресурс] // Reference Documentation. – Version 8.0.53, 2018 – Режим доступу до ресурсу: https://tomcat.apache.org/tomcat-8.0-doc/
16. Thymeleaf Documentation [Електронний ресурс] // Reference Documentation. – Version 3.0.11.RELEASE, 2018 – Режим доступу до ресурсу: https://www.thymeleaf.org/documentation.html
17. Справочник по HTML [Електронний ресурс] // Справочная документация. – Режим доступу до ресурсу: http://htmlbook.ru/html
18. Руководство по CSS [Електронний ресурс] // Справочная документация. – Режим доступу до ресурсу: https://developer .mozilla.org/ru/docs/Web/CSS/Reference

ДОДАТОК А

Лістинг Front-end частини

calculations.html

```
<!DOCTYPE html>
<html xmlns:th="http://thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link type="text/css" rel="stylesheet" th:href="@{/css/main.css}">
   <link type="text/css" rel="stylesheet" th:href="@{/css/calculations.css}">
   <title>Document</title>
</head>
<body>
   <div th:replace="/header"></div>
   <div class="main">
      <div class="controls">
            <a th:href="@{/new_project}" class="controls_link">Нова схема</a>
            <div class="pause"></div>
            <a th:if="${session.project}" th:href="@{/download}" class="controls_link">Скачати</a>
         </div>
      <div class="output" th:each="element : ${results}">
         <div class="element">
            <div class="info" th:text="${element.getElement() + ' - ' + elements.get(element.getElement())}">1 -
Генеральний директор</div>
            <table class="table">
               <tr>
                  <td th:text="${element.getFinalGrate()}">1</td>
                  <td th:text="${element.getFirst()}">1</td>
                  <td th:text="${element.getSecond()}">1</td>
                  <td th:text="${element.getThird()}">1</td>
               </tr>
               <tr>
                  <th>Експерт</th>
                  <th>Вплив на ресурси</th>
                  <th>Контроль потоків</th>
                  <th>Прийняття рішень</th>
               </tr>
               <tr th:each="row : ${secondTour.get(element.getElement())}">
                  <td th:text="${experts.get(row.getExpert())}">Джон</td>
                  <td th:text="${row.getFirst()}">1</td>
                  <td th:text="${row.getSecond()}">2</td>
                  <td th:text="${row.getThird()}">3</td>
               </tr>
            </table>
         </div>
      </div>
   </div>
</body>
</html>
```

ДОДАТОК Б

Лістинг Back-end частини

StartSecondTour.java

```
package project.controllers.commands;

import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Component;
import org.springframework.ui.Model;

import project.dao.ExpertDao;
import project.dao.FirstTourDao;
import project.dao.ProjectDao;
import project.entities.Expert;
import project.entities.FirstTourRes;
import project.entities.Project;

@Component
public class StartSecondTour implements Command {

	private final JavaMailSender mailSender;
	private final ExpertDao expertDao;
	private final ProjectDao projectDao;
	private final FirstTourDao firstTourDao;

	@Autowired
	public StartSecondTour(JavaMailSender mailSender, ExpertDao expertDao, ProjectDao projectDao,
FirstTourDao firstTourDao) {
```

```
                this.mailSender = mailSender;
                this.expertDao = expertDao;
                this.projectDao = projectDao;
                this.firstTourDao = firstTourDao;
        }

        @Override
        public void execute(HttpServletRequest request, HttpServletResponse response, Model model) throws Exception {
                int curExpertId = Integer.valueOf(request.getParameter("expert"));
                Expert curExpert = expertDao.get(curExpertId);
                boolean isOver = projectDao.isFirstTourOver(curExpert.getProject());
                Project project = projectDao.get(curExpert.getProject());

                if (!isOver)
                        return;

                HttpSession session = request.getSession();
                String path = session.getServletContext().getRealPath("/");
                String fileName = path + "data\\output\\" + project.getUser() + "\\" + project.getName() + "\\" + project.getName() + "_legend.txt";
                String text = new String(Files.readAllBytes(Paths.get(fileName)), StandardCharsets.UTF_8);
                int n = (int) Math.round(text.split("\n").length * 0.1);

                List<FirstTourRes> fRes = firstTourDao.getRes(project.getId());
                List<Integer> counts = new ArrayList<>();
                for (FirstTourRes i : fRes) {
                        counts.add(i.getPoints());
                }

                int threshold = Collections.max(counts) + 1;
        int currentNum = 0;
        while (currentNum < n && currentNum <= counts.size()) {
            threshold -= 1;
            int k = 0;
            for (int i : counts)
                if (i >= threshold)
                    k += 1;
                else
                        break;
            currentNum = k;
        }
```

```
        for (FirstTourRes i : fRes) {
            if (i.getPoints() >= threshold)
                    firstTourDao.writeRes(project.getId(), i.getElement());
        }

        List<Expert> experts = expertDao.getProjects(project.getId());
        SimpleMailMessage message = new SimpleMailMessage();
        for (Expert expert : experts) {
                                message.setTo(expert.getEmail());
                                message.setSubject("Ви були запрошені на опитування в якості есперта");
                                message.setText("http://localhost:8080/diploma/second_tour?id=" + expert.getId());

                                mailSender.send(message);
                    }
            }
}
```

CalcResults.java

```
package project.controllers.commands;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.ui.Model;

import project.dao.ExpertDao;
import project.dao.ProjectDao;
import project.dao.SecondTourDao;
import project.entities.Expert;
import project.entities.Result;
import project.entities.SecondTour;

@Component
```

```
public class CalcResults implements Command {

	private final ProjectDao projectDao;
	private final ExpertDao expertDao;
	private final SecondTourDao secondTourDao;

	@Autowired
	public CalcResults(ProjectDao projectDao, ExpertDao expertDao, SecondTourDao secondTourDao) {
		this.projectDao = projectDao;
		this.expertDao = expertDao;
		this.secondTourDao = secondTourDao;
	}

	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response, Model model) throws Exception {
		int expertId = Integer.valueOf(request.getParameter("id"));
		Expert expert = expertDao.get(expertId);
		int projectId = expert.getProject();

		if (!projectDao.isSecondTourOver(projectId))
			return;

		List<SecondTour> ratings = secondTourDao.getProjects(projectId);
		Map<Integer, Result> res = new HashMap<>();

		for (SecondTour rating : ratings) {
			if (!res.containsKey(rating.getElement())) {
				Result newResult = new Result();
				newResult.setElement(rating.getElement());
				newResult.setProject(projectId);
				newResult.setFirst(0);
				newResult.setSecond(0);
				newResult.setThird(0);
				res.put(rating.getElement(), newResult);
			}

			Result curentResult = res.get(rating.getElement());
			curentResult.setFirst(curentResult.getFirst() + rating.getFirst());
			curentResult.setSecond(curentResult.getSecond() + rating.getSecond());
			curentResult.setThird(curentResult.getThird() + rating.getThird());
			res.put(rating.getElement(), curentResult);
```

```
                }

                int expertsNum = expertDao.getProjects(projectId).size();
                for (Entry<Integer, Result> entry : res.entrySet()) {
                        Result currentRes = entry.getValue();
                        currentRes.setFirst(Math.round(currentRes.getFirst() / expertsNum));
                        currentRes.setSecond(Math.round(currentRes.getSecond() / expertsNum));
                        currentRes.setThird(Math.round(currentRes.getThird() / expertsNum));

                        currentRes.setFinalGrate(Math.round((currentRes.getFirst() + currentRes.getSecond() +
currentRes.getThird()) / 3));

                        projectDao.createRes(currentRes);
                }
        }
}
```

ProjectDao.java

```
package project.dao;

import java.sql.SQLException;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

import project.dao.mappers.IntegerMapper;
import project.entities.Project;
import project.entities.Result;

@Component
public class ProjectDao {

        private final JdbcTemplate jdbcTemplate;

        @Autowired
        public ProjectDao(JdbcTemplate jdbcTemplate) {
                this.jdbcTemplate = jdbcTemplate;
        }
```

```
	public Project get(int id) throws SQLException {
		String sql =
				"SELECT * "
				+ "FROM projects "
				+ "WHERE id=?";
		List<Project> projects = jdbcTemplate.query(sql, new Object[] {id}, new BeanPropertyRowMapper<>(Project.class));
		if (projects.size() == 0)
			return null;
		return projects.get(0);
	}

	public Project get(String name, String user) throws SQLException {
		String sql =
				"SELECT * "
				+ "FROM projects "
				+ "WHERE name=? AND user=?";
		List<Project> projects = jdbcTemplate.query(sql, new Object[] {name, user}, new BeanPropertyRowMapper<>(Project.class));
		if (projects.size() == 0)
			return null;
		return projects.get(0);
	}

	public List<Project> getUsersProjects(String user) throws SQLException {
		String sql =
				"SELECT * "
				+ "FROM projects "
				+ "WHERE user=?";
		List<Project> projects = jdbcTemplate.query(sql, new Object[] {user}, new BeanPropertyRowMapper<>(Project.class));
		if (projects.size() == 0)
			return null;
		return projects;
	}

	public int countUsersProjects(String login) {
		String sql =
				"SELECT COUNT(*) "
				+ "FROM projects "
				+ "WHERE user=?";
```

```
		List<Integer> projects = jdbcTemplate.query(sql, new Object[] {login}, new IntegerMapper());
		return projects.get(0);
	}

	public boolean isFirstTourOver(int id) {
		String sql =
				"SELECT * "
				+ "FROM experts "
				+ "LEFT JOIN first_tour ON first_tour.expert=experts.id "
				+ "WHERE project=? AND element IS NULL";
		List<Integer> experts = jdbcTemplate.query(sql, new Object[] {id}, new IntegerMapper());
		return experts.size() == 0;
	}

	public boolean isSecondTourOver(int projectId) {
		String sql =
				"SELECT first_tour_res.element "
				+ "FROM first_tour_res "
				+ "LEFT JOIN experts ON first_tour_res.project=experts.project "
				+ "LEFT JOIN second_tour ON first_tour_res.element=second_tour.element
AND second_tour.expert=experts.id "
				+ "WHERE first_tour_res.project=? AND second_tour.id is NULL";
		List<Integer> elements = jdbcTemplate.query(sql, new Object[] {projectId}, new
IntegerMapper());
		return elements.isEmpty();
	}

	public boolean create(Project proj) throws SQLException {
		String sql =
				"INSERT INTO projects (name, user) "
				+ "VALUES (?, ?)";
		int num = jdbcTemplate.update(sql, proj.getName(), proj.getUser());
		return num > 0;
	}

	public boolean delete(String user, String project) {
		String sql =
				"DELETE FROM projects "
				+ "WHERE user=? AND name=?";
		int num = jdbcTemplate.update(sql, user, project);
		return num > 0;
	}
```

```
	public boolean createRes(Result res) throws SQLException {
		String sql =
				"INSERT INTO results (project, element, final, first_param, second_param, third_param) "
				+ "VALUES (?, ?, ?, ?, ?, ?)";
		int num = jdbcTemplate.update(sql, res.getProject(), res.getElement(), res.getFinalGrate(), res.getFirst(), res.getSecond(), res.getThird());
		return num > 0;
	}

	public List<Result> getRes(int projectId) {
		String sql =
				"SELECT project, element, final AS finalGrate, first_param AS first, second_param AS second, third_param AS third "
				+ "FROM results "
				+ "WHERE project=?";
		List<Result> res = jdbcTemplate.query(sql, new Object[] {projectId}, new BeanPropertyRowMapper<>(Result.class));
		return res;
	}
}
```