Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ ТА РОЗРОБКА ЗАСОБУ АВТОМАТИЗАЦІЇ ПРОЕКТНОЇ ПРОЦЕДУРИ АНАЛІЗУ ВПЛИВУ ПОДІЙ НА СТАН БД ІУС
(тема)

Виконав:
студент 2 курсу, групи ІТПм-22-2
Прокопенко М.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма ІТП
(повна назва спеціалізації)
Керівник проф. Чайніков С.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Гребеннік І.В.
(підпис) (прізвище, ініціали)

2024 р.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено.

Керівник кваліфікаційної роботи проф. Чайніков С.І.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Системотехніки
(повна назва)
Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма ІТП
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Прокопенко Максиму Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів та розробка засобу автоматизації проектної процедури аналізу впливу подій на стан БД ІУС
затверджена наказом університету від 06 жовтня 20 23 р. № 1157Ст
2. Термін подання студентом роботи до екзаменаційної комісії _____ січня 20 24 р.
3. Вихідні дані до роботи Проаналізувати методи аналізу та проектування ІУС, проаналізувати методи проектування ІУС за допомогою впливу подій, розробити CASE засіб для автоматизації аналізу впливу подій на стан БД ІУС. Система повинна представляти собою проект web-сайту. Перелік використовуваних програмних засобів: ОС MacOS, Azure Data Studio, Docker. Інтегроване середовище розробки Visual Studio Code.
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ. 4.2 Аналіз предметної області 4.3 Загальні концепції та визначення баз даних 4.4 Сучасні тенденції у сфері баз даних 4.5 Життєвий цикл інформаційної системи 4.6 Принципи структурного аналізу та аналіз методології SSADM 4.7 Case засоби для роботи із SSADM методологією 4.8 Огляд методів та технологій, які застосовуються в предметній області 4.9 Опис ERD методології як базовий концепт у SSADM 4.10 Опис та роль DFD діаграм у SSADM 4.11 CASE засоби для роботи з DFD та ERD 4.12 Постановка задачі 4.13 EEM: опис та термінологія 4.14 Події 4.15 Матриця подій-об'єктів 4.15 Ефекти: Опис та властивості 4.16 Entity Life History: Опис діаграми та правила користування 4.17 Доцільність реалізації CASE засобу 4.18 Бізнес вимоги 4.19 Функціональні вимоги 4.20 Діаграма використання системи 4.21 Нефункціональні вимоги до CASE застосунку 4.22 Обгрунтування вибору технологій для побудови WEB-застосунку 4.23 Обгрунтування вибору для написання API та БД 4.24

Опис структури розробленого CASE засобу та приклади використання 4.25 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Аналіз предметної області	проф. Чайніков С.І.		
Опис прийнятих проектних рішень	проф. Чайніков С.І.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	06.10.2023	
2	Аналіз досліджуваних методів в існуючих системах	15.10.2023	
3	Постановка задачі дослідження	25.10.2023	
4	Дослідження методів вирішення задачі	20.11.2023	
5	Огляд існуючих підходів технологій та методів у предметній області	05.12.2023	
6	Математичний опис методів рішення задачі	23.12.2023	
7	Розробка інформаційної технології	23.12.2023 – 10.01.2024	
8	Представлення на рецензування	за 3 дні	
9	Представлення кваліфікаційної роботи в ДЕК	за 2 дні	

Дата видачі завдання 06 жовтня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Чайніков С.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаціна робота: 72 стор., 14 рис., 5 табл., 1 додатків, 10 джерел.

АНАЛІЗ, SSADM, БАЗИ ДАНИХ, ПРОЕКТУВАННЯ СИСТЕМ, ІНФОРМАЦІЙНА СИСТЕМА, ERD ДІАГРАМИ, DFD ДІАГРАМИ, WEB-ЗАСТОСУНОК, API, ІНТЕРФЕЙС, NOSQL, JS, NODEJS, REACTJS

Об’єкт дослідження – методи проектування систем та баз даних.

Предмет дослідження – методології проектування систем за допомогою створення та наліза подій.

Мета роботи – дослідження методу проектування систем за допомогою подій та створення CASE засобу для інтеграції EEM методології у процес проектування.

Методи дослідження – аналіз існуючих CASE засобів для проектування систем та баз даних, аналіз методу проектування EEM та виявлення доцільності створення CASE засобу, який інтегрує у собі можливість використання EEM методології.

Результати роботи – проанализовано методологію EEM, проаналізовано існуючі CASE засоби для проектування систем, виявлено, що є необхідність у створенні CASE засобу з інтегрованою можливістю використання EEM методології, створено CASE засіб для використання EEM методології для проектування баз даних.

Область застосування – проектування систем та баз даних, програмування систем, тестування систем.

ABSTRACT

Work qualification: 72 pages, 14 figures, 5 tables, 1 appendices, 10 sources.

ANALYSIS, SSADM, DATABASES, SYSTEM DESIGN, INFORMATION SYSTEM, ERD DIAGRAMS, DFD DIAGRAMS, WEB APPLICATION, API, INTERFACE, NOSQL, JS, NODEJS, REACTJS

The object of research - the methods of designing systems and databases.

The subject of research - the methodology of designing systems using the creation and analysis of events.

The purpose of the work - to study the method of system design using events and create a CASE tool for integrating the EEM methodology into the design process.

Research methods – analysis of existing CASE tools for designing systems and databases, analysis of the EEM design method and identification of the feasibility of creating a CASE tool that integrates the possibility of using the EEM methodology.

The results of the work – the EEM methodology was analyzed, the existing CASE tools for designing systems were analyzed, it was found that there is a need to create a CASE tool with the integrated possibility of using the EEM methodology, a CASE tool was created for using the EEM methodology for database design.

Scope – system and database design, system programming, system testing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧУВАНЬ ТА ТЕРМІНІВ

SSADM – Structured Systems Analysis and Design Methodology;

DFD – Data Flow Diagrams;

EEM – Entity Event Modeling;

ELH – Entity Life History;

ІУС – інформаційно-управляючих систем;

БД – база даних;

СУБД – система управління базами даних;

SQL – Structured Query Language (мова структурованих запитів);

RDBMS – Relational Database Management System;

ERD – Entity Relationship Diagram (модель даних, яка дозволяє описувати концептуальні схеми предметної області);

HTML – HyperText Markup Language (мова гіпертекстової розмітки);

HTTP – HyperText Transfer Prorocol (протокол передачі гіпертекстових файлів);

JS – JavaScript (скриптова мова програмування);

ReactJS – JS фреімверк для створення візуальної частини Web-застосунку;

NodeJS – JS framework для створення серверної частини Web-застосунку;

D3 – безкоштовна бібліотека JavaScript із відкритим кодом для візуалізації даних.

ВСТУП

У сучасному цифровому світі бази даних є фундаментальним елементом управління та обробки інформації. Вони є критично важливими для різноманітних аспектів економічної, соціальної та наукової діяльності, відіграючи ключову роль у зберіганні, обробці та аналізі величезних обсягів даних. У зв'язку з цим, ефективне та оптимізоване проектування баз даних стає вирішальним для забезпечення продуктивності, безпеки та масштабованості сучасних інформаційних систем.

Основною метою цієї дипломної роботи є дослідження методологій проектування та моделювання баз даних, з акцентом на методологію Entity Event Modelling. У контексті стрімкого розвитку технологій та зростаючих вимог до обробки даних, традиційні підходи до проектування баз даних часто не можуть забезпечити необхідну гнучкість та ефективність. Тому пошук інноваційних підходів, які можуть впоратися з сучасними викликами у сфері управління даними, є актуальним та необхідним.

Ця робота також включає детальний огляд інших існуючих методологій проектування баз даних, з метою зробити комплексне порівняння та визначити унікальні переваги та потенційні обмеження кожної методології. Важливість такого аналізу полягає у визначенні найбільш ефективних підходів, що можуть бути використані для розв'язання конкретних проблем, пов'язаних з базами даних, у різних галузях. Особлива увага буде приділена аналізу таких популярних підходів, як ERD (Entity-Relationship Diagram) та DFD (Data Flow Diagram), які вже довгий час є стандартом у галузі.

Для досягнення цих цілей були визначені наступні основні завдання: проведення комплексного огляду сучасних методологій проектування баз даних, глибокий аналіз методології EEM, розробка та тестування CASE-засобу на основі цієї методології, а також оцінка його ефективності та можливостей інтеграції з іншими системами.

У дипломній роботі будуть застосовані різноманітні методи дослідження, включаючи аналітичний, емпіричний, та порівняльний аналіз. Це дозволить всебічно оцінити кожну методологію та визначити її придатність для вирішення специфічних задач у сфері проектування баз даних.

Робота складається з кількох розділів, кожен з яких висвітлює окремий аспект дослідження: від теоретичних основ до практичної реалізації та аналізу результатів. Така структура сприятиме глибокому розумінню теми та дозволить виявити потенціал застосування нових методологій у сучасних умовах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі ми здійснимо глибокий аналіз предметної області баз даних, яка є критично важливою для розробки та ефективного функціонування сучасних інформаційних систем. Бази даних не тільки служать фундаментом для зберігання та організації великих обсягів інформації, але й відіграють ключову роль у забезпеченні інтеграції, аналізу та доступності даних у різноманітних застосуваннях. Вони є невід'ємною частиною більшості систем, від простих веб-сайтів до складних корпоративних систем управління.

З розвитком цифрових технологій та збільшенням обсягів даних, які необхідно обробляти та аналізувати, бази даних набувають ще більшої важливості. Вони є центральним елементом в архітектурі будь-якої інформаційної системи, впливаючи на її продуктивність, масштабованість та безпеку. Це пояснює, чому проектування баз даних та вибір правильної методології моделювання є ключовими факторами успіху в інформаційних технологіях.

У рамках цього розділу ми зосередимо увагу на основних концепціях та визначеннях баз даних, аналізуючи їх роль у загальній структурі інформаційних систем. Ми дослідимо різні типи баз даних, їх архітектури та моделі даних, оцінюючи, як кожен з них може бути застосований для вирішення конкретних бізнес-завдань та технічних вимог. Також буде зроблено акцент на важливості ефективного проектування баз даних для оптимізації процесів обробки та використання даних в різних доменах застосування.

1.1 Загальні концепції та визначення баз даних

Розуміння баз даних та їхніх основних концепцій є критично важливим для ефективного проектування та управління інформаційними системами. Бази даних займають ключове місце в архітектурі сучасних інформаційних систем,

відіграючи вирішальну роль у зберіганні, обробці та аналізі даних. Вони бувають різних типів, включаючи реляційні бази даних (RDBMS), які ідеально підходять для систем, що потребують строгих гарантій цілісності даних, нереляційні бази даних (NoSQL) для великих масивів неструктурованих даних та об'єктно-орієнтовані бази даних, які використовуються в спеціалізованих застосуваннях [8].

Сучасні тенденції у сфері баз даних значно впливають на підходи до їхнього проектування та використання. Зі зростанням обсягів даних (Big Data) та потребою в їх аналізі, технології баз даних еволюціонують у напрямку підтримки швидкої обробки великих масивів даних та аналітики у реальному часі. Хмарні рішення також стають дедалі більш популярними, надаючи гнучкість, масштабованість та знижені витрати на обслуговування. Розвиток архітектур мікросервісів та розподілених систем сприяє використанню легких та гнучких баз даних, що відповідають вимогам швидкої ітерації та незалежного розгортання.

У контексті проектування баз даних, ці тенденції накладають вимоги на вибір архітектури, врахування потреб у масштабованості, швидкості обробки даних та гнучкості систем. Важливість ефективного проектування полягає не лише в створенні функціональної бази даних, а й у забезпеченні її здатності адаптуватися до змінюваних умов та вимог. Це включає вибір правильного типу бази даних, розробку оптимальної схеми даних та імплементацію відповідних методологій проектування, таких як нормалізація в реляційних базах даних або денормалізація в NoSQL. Кожен з цих виборів має свої переваги та обмеження, які слід уважно враховувати при проектуванні систем, що базуються на базах даних.

Цей розділ надає фундамент для глибшого розуміння ролі баз даних у сучасних інформаційних системах та важливості їхнього правильного проектування. Це є ключовим для створення ефективних, надійних та

масштабованих систем, здатних задовольнити різноманітні потреби користувачів і бізнесу.

1.2 Сучасні тенденції у сфері баз даних

Сфера баз даних постійно змінюється та розвивається, а сучасні тенденції визначають напрямок цього розвитку. Ось деякі ключові аспекти, які сьогодні впливають на проектування та використання баз даних:

Сучасні організації стикаються з великими обсягами даних, які надходять з різних джерел, таких як соціальні мережі, сенсори, додатки та інше. Ця тенденція, відома як Big Data, ставить виклик перед базами даних, оскільки вони повинні забезпечувати швидку обробку та аналіз цих великих обсягів даних у реальному часі. Аналітичні рішення, які дозволяють отримувати цінні інсайти з даних, стають дедалі більш важливими.

Хмарні рішення стають дедалі популярнішими завдяки їхній гнучкості, масштабованості та зниженню витрат на обслуговування. Вони дозволяють організаціям зберігати та обробляти дані в хмарному середовищі, що спрощує управління інфраструктурою баз даних та забезпечує доступність даних з будь-якого місця.

Розвиток архітектур мікросервісів стимулює використання легких, гнучких та розподілених баз даних, які відповідають вимогам швидкої ітерації та незалежного розгортання сервісів. Розподілені системи дозволяють обробляти великі обсяги даних та забезпечувати їхню доступність та надійність [9].

З огляду на зростаючу кількість кібератак та порушень безпеки, захист даних стає пріоритетом. Сучасні бази даних повинні мати вбудовані механізми шифрування, аутентифікації та авторизації, а також забезпечувати можливість аудиту дій з даними.

Зростає популярність використання відкритих джерел даних, таких як відкриті API та відкриті дані уряду. Ця тенденція вимагає інтеграції та аналізу різноманітних джерел даних у реальному часі.

Ці сучасні тенденції у сфері баз даних підкреслюють важливість не лише розуміння базових концепцій, а й здатності адаптуватися до постійних змін у світі даних та інформаційних технологій. Наявність правильної архітектури бази даних та відповідних інструментів дозволяє організаціям ефективно використовувати свої дані для прийняття інформованих рішень та досягнення успіху у сучасному світі [8].

Після докладного розгляду баз даних та їхнього застосування в проектуванні та аналізі, важливо розуміти, що бази даних є ключовим елементом у життєвому циклі Інформаційної Системи (ІУС). Бази даних грають критичну роль у всіх етапах створення та функціонування ІУС і мають величезний вплив на їхню ефективність та функціональність.

1.3 Життєвий цикл інформаційної системи

Життєвий цикл ІУС є процесом, що охоплює всі етапи від початкової концепції до виведення системи з експлуатації. Цей цикл розглядається як невід'ємна частина управління та розвитку інформаційних систем. Основні етапи життєвого циклу ІУС включають [10]:

- Початкова фаза (концепція): Визначення бізнес-потреб та формування концепції ІУС. Встановлення основної мети та обсягу проекту.
- Аналіз вимог: Збір, аналіз та формалізація вимог до ІУС. Визначення функціональності та характеристик системи.
- Дизайн: Розробка архітектури ІУС, включаючи структуру даних, інтерфейс користувача та інші компоненти системи.
- Реалізація (розробка): Створення та втілення програмного забезпечення, тестування та виправлення помилок.

- Впровадження: Введення ІУС в експлуатацію, перенесення даних, тренування персоналу та забезпечення коректного функціонування системи.
- Експлуатація та підтримка: Постійний моніторинг та управління роботою ІУС, вирішення проблем, оновлення та підтримка.
- Оновлення та розширення: Зміни та оновлення ІУС відповідно до нових вимог бізнесу, а також можливість розширення функціональності.
- Зняття з експлуатації: Виведення ІУС з експлуатації, архівування даних та завершення циклу.

Перший крок у будь-якому проекті — це визначення життєвого циклу. Це як путівник, який допомагає команді розробників та учасникам проекту управляти та координувати всі аспекти роботи над програмним забезпеченням. Життєвий цикл визначає порядок виконання етапів, починаючи з аналізу вимог і завершуючи впровадженням та підтримкою.

Проте, в процесі розвитку програмного забезпечення виникає безліч викликів, які потребують ретельного вирішення. Саме тут набуває значення використання моделей розробки програмного забезпечення. Моделі стають інструментом, який допомагає вирішувати проблеми та оптимізувати процеси на різних етапах життєвого циклу [10]. Вони допомагають визначити та уточнити вимоги, прогнозувати та управляти ризиками, моделювати архітектуру та забезпечувати ефективне прийняття рішень.

Тож, перейдемо до більш детального розгляду різних моделей розробки програмного забезпечення, розглянемо їх особливості та вплив на процес розробки. Розуміння цих моделей стане ключовим кроком у вдосконаленні та оптимізації процесу розробки програмного забезпечення. Моделі допомагають вирішувати численні питання та завдання на різних етапах життєвого циклу проекту.

Існує кілька способів ведення та реалізації життєвого циклу проекту. Кожна модель має свої переваги та недоліки, і вибір залежить від конкретних умов проекту.

- Каскадна модель: Проект розбивається на етапи, і кожен етап починається після завершення попереднього. Це простий і зрозумілий підхід, але менше гнучкий при змінах.
- Спиральна модель: Комбінує елементи ітераційного та каскадного підходів. Проект розглядається як спіраль, яку можна пройти кілька разів. Це надає гнучкість та поступове розширення можливостей проекту.
- Ітераційна та інкрементна модель: Проект розбивається на ітерації, кожна з яких представляє собою повний цикл розробки. Це дозволяє вводити зміни на будь-якому етапі та швидко впроваджувати частковий функціонал.
- RAD (Rapid Application Development): Зосереджена на швидкому розробленні та випробуванні прототипів. Дозволяє швидко впроваджувати функціонал та змінювати напрямок розробки.
- V-модель: Схожа на каскадну модель, але кожен етап розробки має відповідний етап тестування. Забезпечує велику увагу тестуванню та валідації.

Розглянувши детально життєвий цикл інформаційних управлінських систем (ІУС), який включає етапи планування, аналізу, проектування, реалізації та підтримки, ми можемо бачити, що кожен етап відіграє критичну роль у розвитку ефективних систем. Ці етапи надають структуру, яка допомагає забезпечити, що ІУС відповідає бізнес-вимогам та забезпечує необхідну функціональність. Проте, для досягнення цих цілей необхідний глибокий структурний аналіз. Тут допоможе методологія SSADM, яка забезпечує чіткий процес структурного аналізу та проектування. SSADM допомагає систематизувати процес розробки ІУС, використовуючи набір принципів та

технік, які забезпечують організований підхід до аналізу та проектування систем. Таким чином, переходячи від загального опису життєвого циклу ІУС до більш специфічних принципів SSADM, ми зосереджуємося на тому, як структурований підхід може покращити якість та ефективність розробки інформаційних систем.

1.4 Принципи структурного аналізу та аналіз методології SSADM

Методологія SSADM, відома як Structured Systems Analysis and Design Method, є стандартною практикою для аналізу та проектування інформаційних систем, особливо популярною в урядових структурах та великих корпораціях. Основна мета SSADM полягає в тому, щоб надати структурований механізм для організації та керування процесом розробки системи. Ця методологія базується на принципах водоспадної моделі розвитку, де кожен етап розробки має чітко визначені результати та залежності.

SSADM акцентує на важливості фаз та модулів. Цей підхід розділяє процес на кілька ключових фаз: аналіз вимог, логічне проектування та фізичне проектування, кожна з яких складається з низки модулів, які детально описують необхідні кроки в процесі розробки. Одним із ключових елементів SSADM є моделювання даних, яке вимагає створення докладних моделей для точного відображення інформації у системі [2]. Крім того, методологія зосереджується на моделюванні вимог, включаючи ідентифікацію, аналіз та документування вимог користувачів, а також на моделюванні процесів, яке визначає, як бізнес-операції будуть виконуватися в межах системи.

Особливої уваги заслуговує ітеративний розвиток, який дозволяє повертатися до попередніх етапів для перегляду та уточнення. Цей підхід є важливим для забезпечення гнучкості процесу та адаптації до змінних вимог проекту. У рамках SSADM використовуються різні типи аналізу та проектування, включаючи логічний аналіз даних, який зосереджений на

структурі даних без урахування їх фізичного зберігання, та логічний аналіз процесів, що визначає процеси та їх взаємодії, не залучаючи конкретні технічні рішення. Також існує фізичне проектування, яке враховує обмеження апаратного та програмного забезпечення.

Хоча SSADM пропонує структурований підхід, який забезпечує чіткість та послідовність у процесі розробки, та генерує об'ємну та корисну документацію, цей підхід може бути часозатратним через детальний аналіз та документацію [1]. Також він виявляється менш гнучким у порівнянні з аджайл-методологіями та може бути занадто складним для малих проектів.

SSADM найкраще підходить для великих та складних проектів, де необхідна детальна документація та чітке розуміння вимог, особливо коли ці вимоги відомі заздалегідь і малоймовірно зміняться. Це ідеально підходить для урядових та корпоративних систем, де потрібен високий рівень формалізації та структури.

Після ретельного розгляду методології SSADM, її принципів та особливостей, наступним логічним кроком у нашому дослідженні є розгляд CASE-засобів (Computer-Aided Software Engineering), які спеціально розроблені для підтримки та оптимізації процесів, властивих SSADM. Важливо розуміти, для чого вони потрібні та як вони можуть підсилити ефективність розробки інформаційних систем, використовуючи цю методологію.

1.5 CASE засоби для роботи із SSADM методологією

CASE-засоби для SSADM є ключовими інструментами, які спрощують та автоматизують багато аспектів проектування системи. Вони забезпечують візуалізацію, моделювання, аналіз вимог, та документування процесів, що дозволяє розробникам зосередитися на більш творчих та аналітичних аспектах роботи, замість рутинного ведення документації та планування. Ці засоби стають незамінними при реалізації складних проектів, оскільки вони

допомагають вирішувати завдання структурування та управління великими масивами даних та процесів, характерних для SSADM.

На сьогоднішній день існує ряд CASE-інструментів, які розроблені спеціально для використання в рамках SSADM. До таких засобів можна віднести, наприклад, Rational Rose, Enterprise Architect, та інші. Кожен з цих інструментів має свої особливості та функціонал, проте всі вони спрямовані на полегшення процесу проектування систем за допомогою автоматизації, візуалізації та підтримки документації. Використання таких засобів не тільки покращує якість кінцевого продукту, але й значно скорочує час, необхідний для розробки, що робить їх незамінними в арсеналі сучасного розробника систем.

2 ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ, ЯКІ ЗАСТОСОВУЮТЬСЯ В ПРЕДМЕТНІЙ ОБЛАСТІ

SSADM, будучи однією з найбільш комплексних та структурованих методологій для розробки інформаційних систем, включає ряд специфічних методів та технік, які сприяють ефективному аналізу, проектуванню та реалізації систем [3].

Серед цих методів особливе місце займають такі інструменти, як ERD (Entity-Relationship Diagram) та DFD (Data Flow Diagram). ERD є фундаментальним інструментом для візуалізації та структурування зв'язків між різними елементами системи, що дозволяє глибше зрозуміти її архітектуру та внутрішню логіку. З іншого боку, DFD забезпечує зрозуміле представлення потоків даних всередині системи, відображаючи, як інформація переміщується та обробляється між різними компонентами [11].

Цей розділ надасть нам можливість краще зрозуміти, як саме ці та інші методи впливають на процес розробки систем за допомогою SSADM, та яку роль вони відіграють у досягненні високої якості та ефективності кінцевих продуктів. Ми розглянемо, як ці інструменти інтегруються у загальну структуру SSADM та як вони допомагають розробникам у вирішенні конкретних задач проектування та аналізу.

2.1 Опис ERD методології як базовий концепт у SSADM

ERD є фундаментальним інструментом в області аналізу та проектування баз даних, який використовується для візуалізації та структурування даних у системах. Ця діаграма забезпечує інтуїтивно зрозуміле графічне представлення структури бази даних, що включає сутності, їх атрибути та зв'язки між ними. Ключовою особливістю ERD є її здатність забезпечити чітке розуміння

взаємодії та залежностей між різними компонентами системи, що значно спрощує процес проектування та розвитку.

ERD вирішує цілий ряд проблем, з якими стикаються розробники та аналітики систем. Насамперед, вона допомагає у виявленні та структуруванні сутностей, що є частиною системи, та відносин між ними [4]. Це може включати, наприклад, клієнтів, замовлення, продукти в контексті бізнес-системи. Кожна сутність представлена на діаграмі зі своїми відповідними атрибутами, що дає повне уявлення про її характеристики та властивості.

Крім того, ERD ефективно візуалізує зв'язки між сутностями, що є критичним для розуміння, як дані взаємодіють у системі. Ці зв'язки можуть бути різних типів, таких як "один до одного", "один до багатьох" чи "багато до багатьох", та кожен тип має свої особливості та вплив на структуру бази даних. Правильне розуміння та відображення цих зв'язків є ключовим для створення ефективних та оптимізованих баз даних.

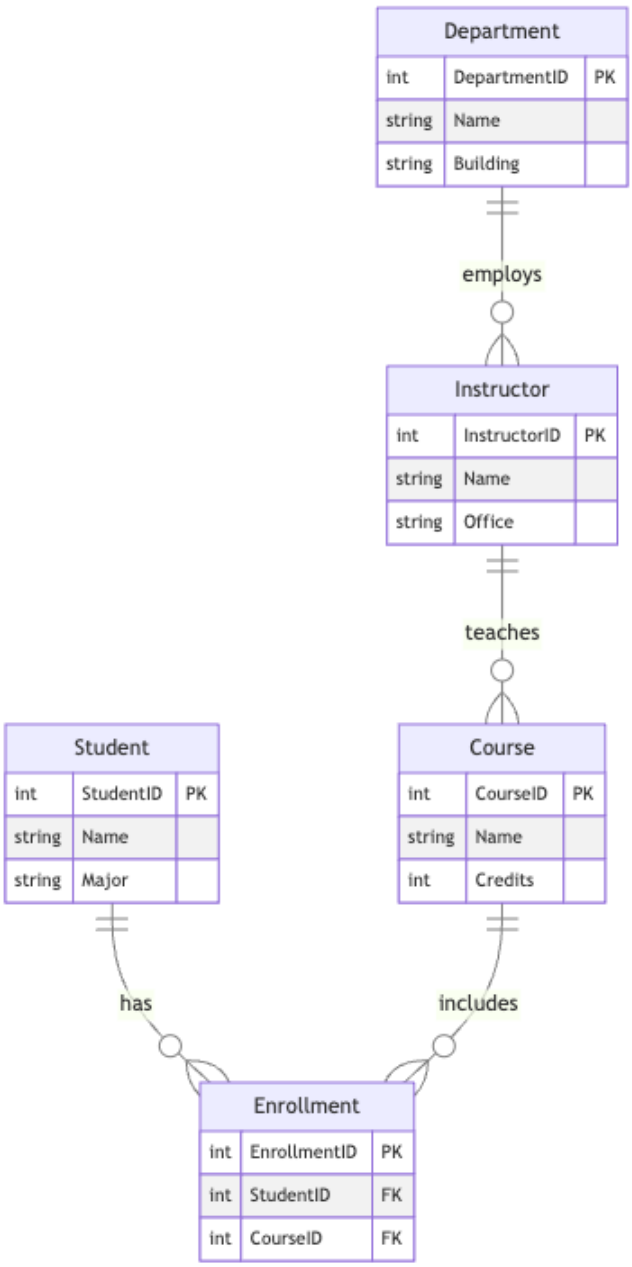

Рис. 2.1 – Приклад ERD діаграми

Ще однією важливою перевагою ERD є її роль у спілкуванні та документації. Діаграми ER забезпечують зрозумілий та стандартизований спосіб представлення даних, який може бути легко зрозумілим як технічними, так і не технічними сторонами проекту. Це робить ERD незамінною для

обговорення структури бази даних з клієнтами, менеджерами та іншими зацікавленими сторонами, які можуть не мати глибоких технічних знань.

Загалом, ERD є важливим інструментом, який дозволяє ефективно планувати, аналізувати та проектувати бази даних, забезпечуючи чітке та послідовне розуміння структури даних та їх взаємозв'язків. Використання ERD є фундаментальним для будь-якого проекту, що вимагає ретельного аналізу та організації даних, та є ключовим елементом в процесі проектування ефективних інформаційних систем.

Переходячи від концепції ERD, яка зосереджена на структурі та взаємозв'язках між даними в системі, ми тепер підходимо до розгляду ще одного критичного елемента в методології SSADM - Data Flow Diagrams (DFD). DFD відкриває новий вимір у процесі аналізу та проектування систем, доповнюючи статичну структуру даних, представлену в ERD, динамічним візуалізуванням потоків даних [11].

2.2 Опис та роль DFD діаграм у SSADM

Data Flow Diagrams є центральним елементом в методології SSADM та відіграють важливу роль у візуалізації та аналізі потоків даних в інформаційних системах. DFD дозволяють аналітикам та розробникам систем зрозуміти, як дані переміщаються всередині системи, включаючи вхідні та вихідні точки, шляхи обробки даних та взаємодію між різними компонентами системи. Ця можливість забезпечує глибоке розуміння бізнес-процесів, що є необхідним для ефективного проектування та реалізації системи.

У контексті SSADM, DFD використовуються на різних етапах розробки для забезпечення, що всі вимоги та функціональності системи ретельно враховані [4]. Наприклад, на етапі аналізу вимог DFD використовуються для визначення та відображення основних процесів, які має виконувати система, та способів обробки даних. Це допомагає уточнити та узгодити вимоги до системи

з зацікавленими сторонами, забезпечуючи, що кінцева система відповідатиме очікуванням користувачів.

DFD відрізняються від ERD, які зосереджені на структурі даних та їх взаємозв'язках. Тоді як ERD ефективно демонструють зв'язки між сутностями та атрибутами в базі даних, DFD надають більш динамічне розуміння системи, показуючи, як дані перетікають між процесами та як ці процеси взаємодіють. Отже, DFD забезпечують ширший погляд на бізнес-логіку та потоки даних, що є важливим для розуміння загальної архітектури та функціонування системи.

DFD також є невід'ємною частиною процесу документації в SSADM. Вони допомагають візуально представити складні бізнес-процеси та потоки даних, роблячи інформацію доступною та зрозумілою не тільки для технічних спеціалістів, але й для менеджерів проектів, аналітиків бізнесу та інших зацікавлених сторін, які можуть не мати глибоких технічних знань. Це сприяє кращому розумінню та узгодженню вимог до системи на всіх рівнях управління проектом [11].

DFD у SSADM є ключовим інструментом для ефективного візуального представлення та аналізу потоків даних в системі. Вони допомагають уточнити вимоги, забезпечити чітке розуміння бізнес-процесів та сприяють ефективному спілкуванню між усіма учасниками проекту, підвищуючи якість та ефективність розробки системи.

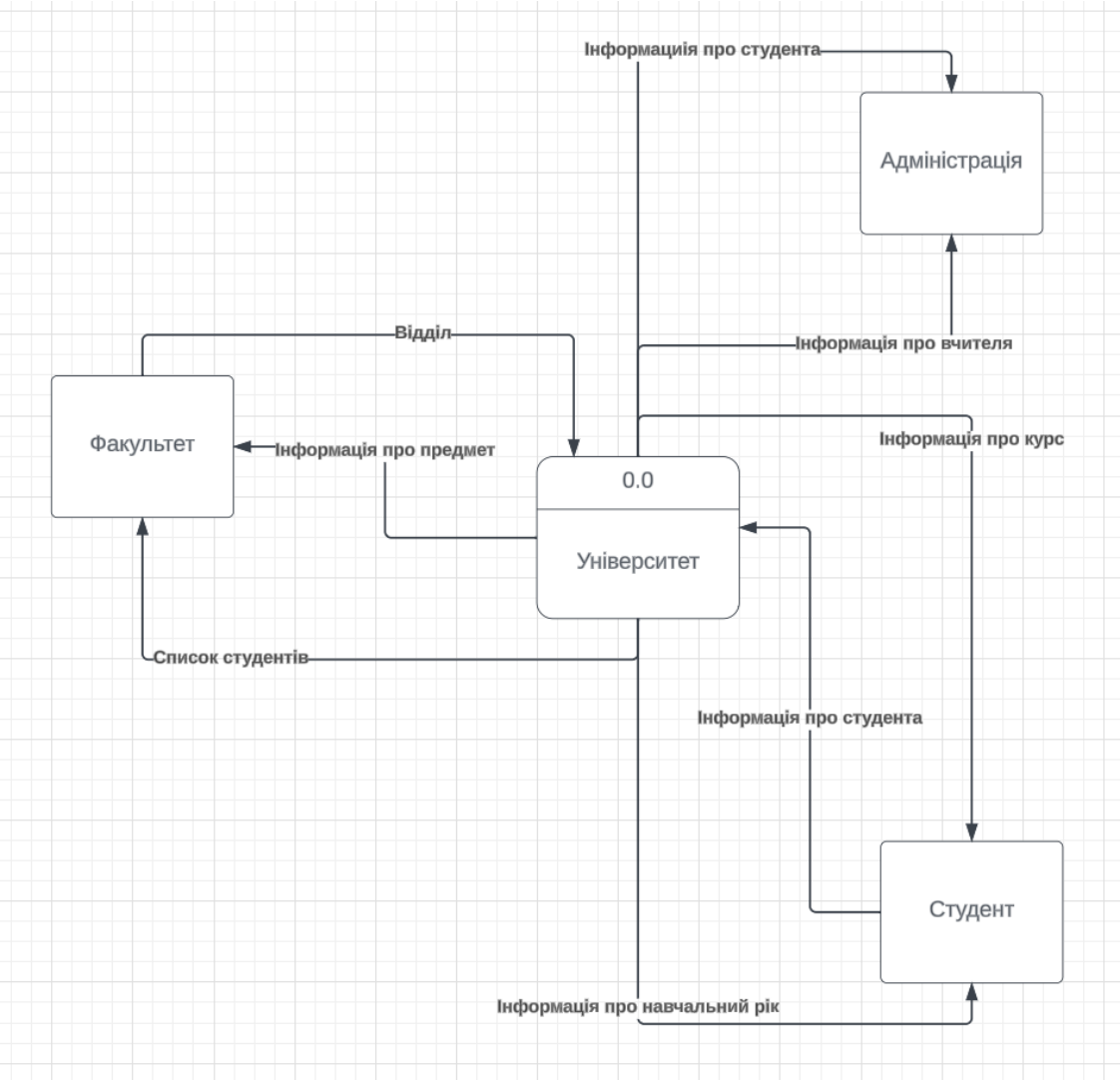

Рис. 2.2 – Приклад DFD діаграми

Використання рівнів в Data Flow Diagrams є ключовою частиною їх ефективності та універсальності, особливо в контексті SSADM. Рівні в DFD дозволяють розглядати систему з різних перспектив, розкладаючи її на більш деталізовані частини для кращого аналізу та розуміння. Ця властивість робить DFD незамінним інструментом для розробки складних систем, де необхідно ретельно вивчити кожен аспект функціонування. Розглянемо основні рівні, які можна виділити при побудові DFD:

- Контекстна діаграма (Рівень 0): Це найвищий рівень в DFD, який представляє систему в цілому як один процес. На цьому рівні показано, як система взаємодіє з зовнішнім середовищем, включаючи

користувачів, зовнішні системи та інші сутності. Контекстна діаграма дозволяє отримати загальне уявлення про систему та її зв'язки з зовнішнім світом.

- Рівень 1 (та вище): На цьому рівні DFD розкладає головний процес, представлений на контекстній діаграмі, на більш деталізовані підпроцеси. Кожен підпроцес представлений окремими діаграмами, які показують, як дані потокують всередині цих підпроцесів. Рівень 1 допомагає розробникам зрозуміти деталі бізнес-процесів та їх взаємодію всередині системи.
- Подальші рівні (2, 3, ...): Ці рівні вводять ще більшу деталізацію, розбиваючи підпроцеси рівня 1 на ще менші елементи. Це дозволяє розробникам детально аналізувати окремі частини системи та їх функціонування, виявляючи специфічні потреби та вимоги до кожного аспекту системи.

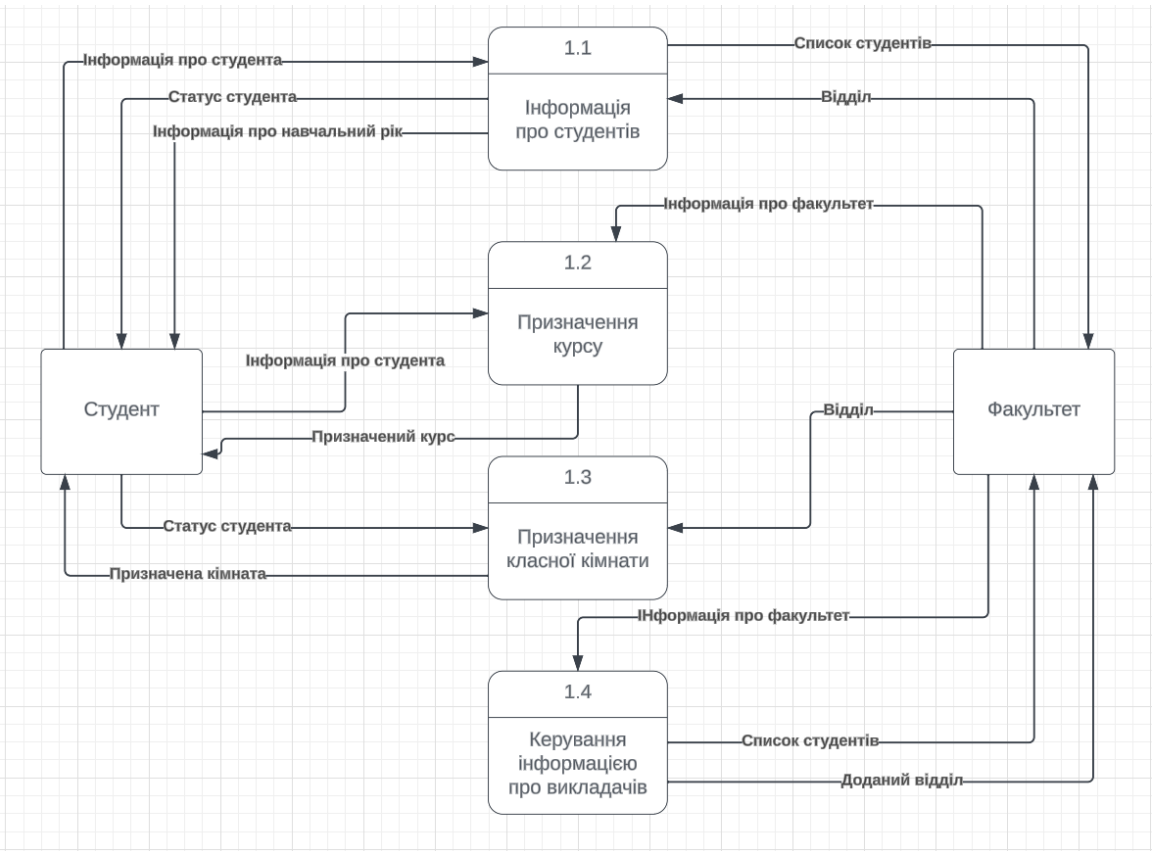

Рис. 2.3 – Приклад DFD діаграми 1го рівня

Створення рівнів у DFD є необхідним для кількох причин. Насамперед, це дозволяє розробникам поступово занурюватися в деталі системи, починаючи від загальної картини та поступово переходячи до більш специфічних аспектів. Такий підхід сприяє кращому розумінню складних систем та допомагає виявити потенційні проблеми або недоліки на ранніх етапах проектування.

Крім того, рівні в DFD забезпечують чітке та організоване представлення інформації. Замість того, щоб намагатися вмістити всю інформацію про систему на одній діаграмі, що може бути перевантаженою та незрозумілою, рівні дозволяють подавати інформацію у вигляді легкозрозумілих, керованих частин. Це не лише спрощує аналіз та розробку, але й полегшує спілкування між різними членами команди та зацікавленими сторонами проекту.

Використання рівнів у DFD є фундаментальним для ефективного аналізу, планування та проектування інформаційних систем в рамках методології SSADM. Цей підхід дозволяє не лише детально вивчити кожен аспект системи, але й забезпечує чітке та організоване представлення складних бізнес-процесів.

Переходячи від поглибленого аналізу методологій DFD та ERD які є критичними компонентами в процесі аналізу та проектування інформаційних систем в рамках методології SSADM, цілком доречним є розгляд ролі та значення CASE засобів. Ці інструменти надають суттєву підтримку в реалізації вищезазначених методологій, зокрема, шляхом автоматизації ключових процесів розробки, включаючи створення та управління DFD та ERD, що сприяє підвищенню продуктивності та точності проектування систем.

В наступному розділі цієї дипломної роботи зосереджується увага на впливі CASE засобів на процес проектування інформаційних систем. Буде проведено детальний аналіз взаємодії цих інструментів з методологіями DFD та ERD та оцінка їх внеску в оптимізацію проектного процесу. Особлива увага буде приділена різноманіттю доступних CASE засобів, їх ключовим

функціоналам та перевагам, які вони можуть надати в контексті використання методології SSADM.

2.3 CASE засоби для роботи з DFD та ERD

CASE засоби - це широкий клас інструментів, що використовуються для полегшення процесів проектування, аналізу, тестування та підтримки програмного забезпечення. Вони забезпечують автоматизацію багатьох аспектів розробки програмного забезпечення, від початкових етапів аналізу вимог до підтримки та оновлення готових систем. Основна мета CASE засобів - підвищення продуктивності розробників, покращення якості програмного продукту та забезпечення стандартизації процесів розробки.

Що стосується використання CASE засобів для створення та управління ERD, це включає широкий спектр функцій, які спрощують процеси візуалізації, моделювання та аналізу баз даних. Наприклад, CASE інструменти дозволяють автоматично генерувати ERD на основі визначених даних, що значно скорочує час, необхідний для їх створення вручну, та знижує ризик помилок.

Одним з популярних прикладів CASE засобів, які підтримують роботу з ERD, є Microsoft Visio. Visio надає розробникам набір зручних шаблонів та інструментів для створення ERD, дозволяючи ефективно візуалізувати структуру бази даних та зв'язки між її елементами. Також можна використовувати такі інструменти, як Lucidchart або ER/Studio, які також пропонують розширені можливості для створення та управління ERD.

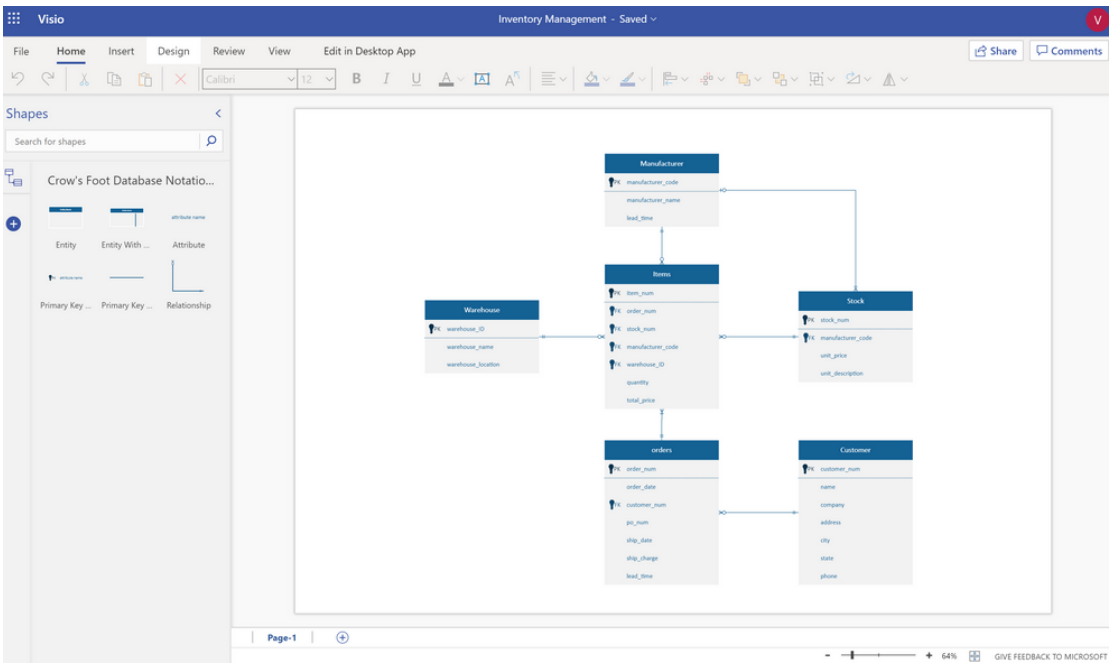

Рис. 2.4 – Приклад ERD діаграми у Microsoft Visio

Крім того, деякі більш комплексні CASE системи, такі як Oracle SQL Developer або IBM Rational Rose, включають в себе функціональність для роботи з ERD як частину більш широкого набору інструментів для проектування та управління базами даних. Вони не тільки дозволяють створювати та редагувати ERD, але й забезпечують інтеграцію з іншими аспектами баз даних, такими як SQL скрипти, схеми баз даних та інші пов'язані елементи.

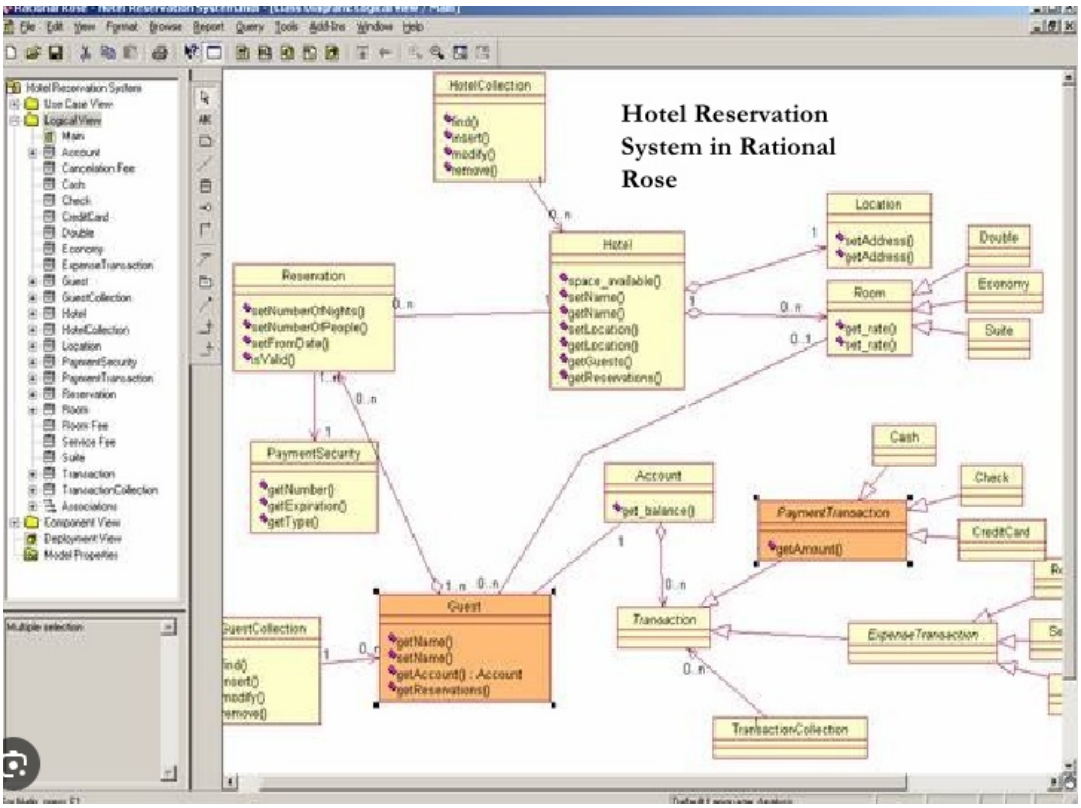

Рис. 2.5 – Приклад ERD діаграми у IBM Rational Rose

CASE засоби для DFD надають можливість розробникам швидко та ефективно створювати діаграми, які відображають як дані пересуваються в системі, де вони обробляються, та як вони зберігаються. Це значно знижує час, необхідний для ручного проектування цих складних діаграм, і допомагає уникнути помилок, які можуть виникнути при ручному введенні.

Один з популярних інструментів для побудови DFD є Lucidchart, який дозволяє користувачам онлайн створювати та спільно використовувати DFD, що сприяє співпраці в команді та забезпечує легкий доступ до діаграм з будь-якого місця. IBM Rational Rose також є важливим інструментом у цій області, який забезпечує більш комплексний підхід до аналізу та проектування систем, включаючи підтримку DFD.

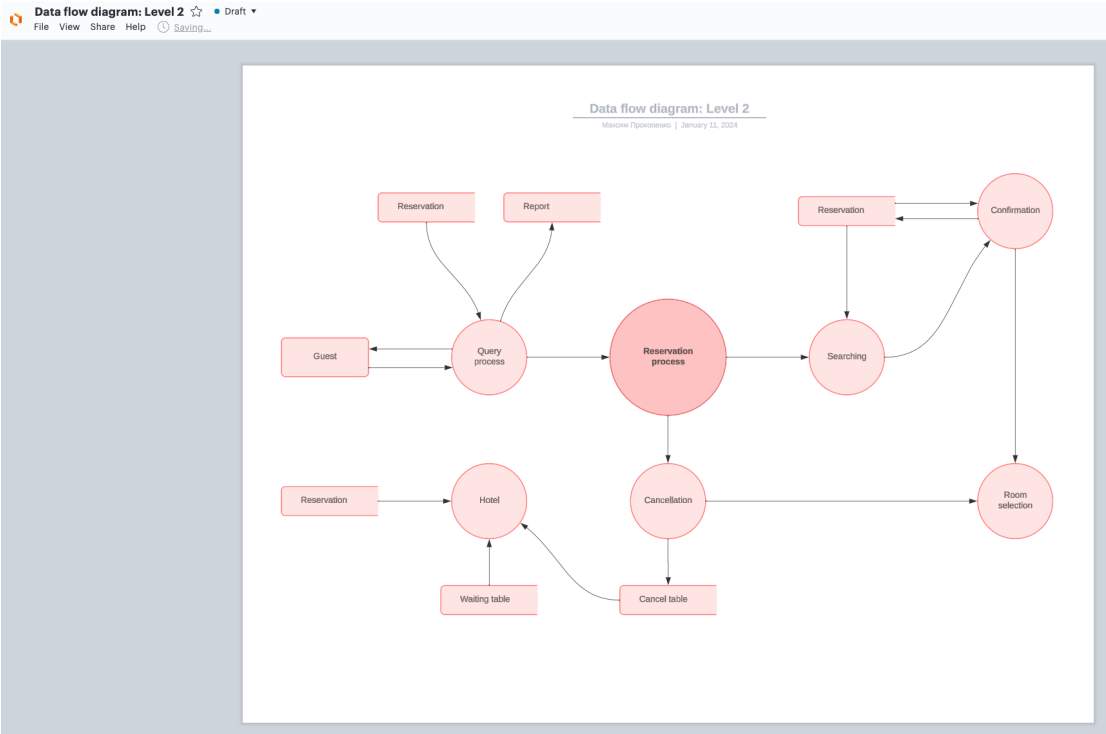

Рис. 2.6 – Lucidchart CASE засіб для побудови DFD діаграм

CASE засоби, які підтримують ERD, дозволяють глибоко аналізувати та візуалізувати структуру бази даних, забезпечуючи чітке розуміння взаємозв'язків та атрибутів сутностей. З іншого боку, інструменти для DFD спрощують процес зображення потоків даних всередині системи, підвищуючи розуміння взаємодій та процесів обробки інформації. Це дозволяє розробникам і аналітикам ефективно виявляти потенційні проблеми, оптимізувати процеси та гарантувати, що кінцева система буде відповідати всім вимогам.

Використання CASE засобів для ERD та DFD в рамках методології SSADM або будь-якої іншої методології розробки значно підвищує якість та ефективність проектних процесів. Вони дозволяють краще розуміти складні системи, сприяють точному визначенню вимог та забезпечують ефективну комунікацію між усіма учасниками проекту. Таким чином, CASE засоби є невід'ємною частиною процесу розробки якісних та надійних інформаційних систем.

3 ПОСТАНОВКА ЗАДАЧІ

Постановка задачі включатиме чітке визначення проблем, які потребують розгляду, окреслення основних питань, які будуть досліджуватися, та формулювання конкретних завдань, що повинні бути виконані в рамках дипломної роботи. Це також передбачає визначення мети дослідження та його об'єктивів, які будуть слугувати основою для подальшого аналізу та вивчення вибраної тематики.

У цьому розділі ми також окреслимо методологічні підходи та інструменти, які будуть застосовані для вирішення поставлених задач, забезпечуючи цим обґрунтований та систематичний підхід до дослідження. Важливим аспектом буде також визначення обмежень та припущень, на яких базуватиметься дослідження, що допоможе забезпечити чіткість та об'єктивність отриманих результатів.

Загалом, цей розділ відіграє ключову роль у визначенні курсу всього дослідницького процесу та підготовці грунту для подальшої детальної роботи над дипломною роботою.

3.1 Доцільність розробки CASE засобу для EEM методології

У цьому розділі дипломної роботи ставиться завдання провести глибокий аналіз методології Entity Event Modelling. Основна мета аналізу полягає у визначенні доцільності розробки спеціалізованого CASE засобу, призначеного для підтримки та оптимізації процесів, які виконуються в рамках цієї методології.

Для досягнення цієї мети, задача буде включати наступні ключові аспекти:

- Детальний огляд методології Entity Event Modelling: Оцінка основних принципів та процедур, що використовуються в методології, з особливим акцентом на її особливості, сильні та слабкі сторони.
- Аналіз поточного стану CASE засобів у сфері проектування баз даних та системного аналізу: Огляд існуючих інструментів, їх можливостей та обмежень у контексті моделювання сутностей та подій.
- Оцінка потреб Entity Event Modelling у контексті CASE засобів: Аналіз, чи існуючі інструменти достатньо ефективні для підтримки цієї методології, або ж існує потреба у розробці спеціалізованого CASE засобу.
- Розробка критеріїв оцінки доцільності створення спеціалізованого CASE засобу для Entity Event Modelling: Визначення ключових параметрів та метрик, які будуть використані для оцінки переваг та вартості розробки нового інструменту.
- Аналіз можливих вигод та ризиків, пов'язаних з розробкою нового CASE засобу: Оцінка потенційного впливу на процес розробки, якість кінцевих систем, а також врахування часових та фінансових аспектів розробки.

Результатом цього аналізу має стати вмотивоване висновок щодо доцільності створення спеціалізованого CASE засобу для методології Entity Event Modelling. Це дозволить встановити, чи зможе такий інструмент значно покращити процеси проектування та розробки в рамках зазначеної методології, а також оцінити потенційний попит та ефективність його впровадження у практику розробки програмного забезпечення.

3.2 Створення CASE засобу для EEM методології

В рамках цього розділу дипломної роботи ставиться задача розробки інноваційного CASE засобу, спеціально призначеного для підтримки та

оптимізації процесів, які використовуються у методології Entity Event Modelling. Ця задача не тільки вимагає глибокого розуміння принципів EEM, але й зосереджується на технічних аспектах розробки програмного забезпечення, що може підтримувати складні процеси моделювання в рамках цієї методології. Ключовою метою є створення ефективного та інтуїтивно зрозумілого інструменту, який сприятиме аналітикам та розробникам у візуалізації, аналізі та управлінні структурою даних та процесами в рамках EEM.

Вхідні дані та задачі:

- Збір інформації про сутності та їх атрибути: Вхідні дані повинні включати детальну інформацію про кожну сутність, що аналізується в рамках EEM, включаючи їх атрибути та зв'язки.
- Інформація про життєві цикли сутностей: Дані про різні стани та події, які можуть відбуватися з кожною сутністю протягом її життєвого циклу.
- Інтеграція з існуючими даними та системами: Можливість інтегрувати CASE засіб з існуючими базами даних та системами, що вже використовуються в рамках EEM.

Вихідні результати та функціональність:

- Створення та управління ERD: CASE засіб повинен дозволяти користувачам створювати та управляти ERD, візуалізуючи структуру бази даних і зв'язки між сутностями.
- Внесення та відображення інформації про життєві цикли сутностей: Функціональність для запису та відображення детальної інформації про події, що впливають на сутності, з можливістю перегляду та аналізу цих подій.
- Перегляд та аналіз усіх подій: Забезпечити можливість переглядати список усіх подій, що впливають на сутності, та аналізувати їх вплив на загальну систему.

- Web інтерфейс: Розробка засобу у формі веб-додатку для забезпечення легкого доступу та використання, незалежно від місцезнаходження користувача.

Розробка такого засобу вимагає детального планування та виконання, включаючи вибір технологій, методологію розробки, тестування та впровадження. Кінцевим результатом має стати повнофункціональний, гнучкий та надійний CASE засіб, який не тільки полегшить процеси в рамках ЕЕМ, але й підвищить загальну якість та ефективність розробки інформаційних систем.

4 ОПИС ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВИРІШЕННЯ ЗАДАЧІ

У цьому розділі дипломної роботи ми зосереджуємося на детальному описі інформаційної технології, яка лежить в основі методології Entity Event Modelling (EEM). Ця методологія є ключовою для розуміння та аналізу даних та процесів у складних інформаційних системах, і, як така, заслуговує на всебічне вивчення.

EEM відрізняється своїм унікальним підходом до моделювання бізнес-процесів та даних, що забезпечує глибоке розуміння взаємодій між різними елементами системи. В цьому розділі ми детально розглянемо теоретичні основи EEM, включаючи її ключові концепції, принципи та техніки. Окрема увага буде приділена розгляду таких аспектів, як структура та атрибути сутностей, їх життєві цикли та події, що на них впливають, а також способи їх візуалізації та аналізу в рамках розроблюваного CASE засобу.

Також ми охопимо практичні аспекти застосування EEM, аналізуючи, як ця методологія може бути втілена в реальних проектах. Це включає розгляд потенційних викликів та переваг, які EEM може принести для процесу розробки інформаційних систем. Крім того, буде здійснено аналіз інструментів та технік, необхідних для ефективної реалізації EEM в рамках сучасного програмного забезпечення.

Завершення цього розділу дасть чітке уявлення про методологію EEM, її місце в сучасному світі розробки програмного забезпечення та перспективи її подальшого розвитку та застосування.

4.1 EEM: Опис та термінологія

Логічне моделювання даних описує статичну структуру стану системи; EEM описує, як такий стан системи розвивається протягом життя системи. EEM використовує статичну структуру станів системи як допомогу для

локалізації також опису динамічної поведінки системи. Тому він поєднує терміни зі статичного опису системи (сутності, зв’язки, атрибути) з термінами, що спеціалізуються на динамічних змінах системи, зокрема термін події.

У розробці системи з використанням SSADM методи EEM використовуються в момент, коли базові неофіційні вимоги вже узгоджені. EEM використовується для фіксації технічних деталей системи, яка відповідає цим неофіційним вимогам, і для перевірки вимог щодо узгодженості.

Техніка EEM містить велику кількість позначень, тому формалізація стає досить складною. Щоб впоратися з цією складністю, ми розглянемо EEM у декілька окремих кроків.

4.1.1 Події у EEM

У контексті EEM, концепція події займає центральне місце та відіграє ключову роль у розумінні та моделюванні бізнес-процесів та системних взаємодій. Подія в EEM можна визначити як значиму зміну або важливий момент в життєвому циклі сутності, який впливає на стан або атрибути цієї сутності. Це може бути, наприклад, створення нового запису, оновлення існуючого, або ж видалення. Події визначають динаміку системи та є основою для аналізу поведінки та взаємодій в системі.

Події в EEM є центральними для розуміння як інформація та дані перетікають та обробляються в рамках системи. Вони дозволяють відобразити не тільки статичні структури, як, наприклад, у ERD, але й динамічні процеси, що відбуваються в системі [6]. Важливість подій полягає у тому, що вони допомагають моделювати реальні сценарії, які відбуваються в бізнес-процесах, та відображати реальні взаємодії між сутностями.

Події в EEM мають ряд важливих характеристик. Насамперед, вони є дискретними, що означає, що вони відбуваються в певний момент часу та мають чітко визначені початок та кінець. Крім того, кожна подія має свої

причини та наслідки, що дозволяє створювати логічні зв'язки між різними подіями та сутностями в системі. Події можуть викликати зміни в станах сутностей або викликати інші події, утворюючи складну мережу взаємодій.

Ще однією важливою характеристикою подій в EEM є їхня специфікація. Для кожної події можна визначити ряд параметрів, таких як її тип, важливість, учасники, пов'язані дані та умови її виникнення. Ця деталізація дозволяє глибше аналізувати та розуміти бізнес-процеси, а також допомагає в розробці більш точних та ефективних інформаційних систем.

Враховуючи важливість подій в EEM, розробка інструментів та методик для їх моделювання та аналізу є критичною задачею. Це включає створення відповідних засобів для запису, відображення та аналізу подій, що відбуваються в системі, а також розробку методик для оцінки їх впливу на бізнес-процеси та рішення. В цьому контексті, CASE засоби, що підтримують EEM, повинні надавати можливості для ефективного та інтуїтивного моделювання подій, дозволяючи аналітикам та розробникам точно відображати та аналізувати реальні бізнес-процеси.

4.1.2 Матриця подій-об’єктів у EEM

Матриця подій-об'єктів в методології Entity Event Modelling є важливим інструментом для аналізу та візуалізації взаємодій між сутностями (об'єктами) та подіями в системі. Цей інструмент дозволяє детально вивчити, як різні події впливають на об'єкти та як ці взаємодії формують загальну поведінку та функціональність системи.

Матриця подій-об'єктів структурована у вигляді таблиці, де рядки зазвичай представляють сутності (об'єкти) системи, а стовпці – різні події [7]. Кожна комірка матриці вказує на взаємодію між конкретною сутністю та подією. Це може включати інформацію про те, чи впливає подія на сутність,

яким чином вона це робить, та які зміни в сутності або її станах відбуваються в результаті цієї події.

Розглянемо процес, який застосовується для створення матриці:

- Ідентифікація об'єктів та подій: Першим кроком у створенні матриці є визначення основних об'єктів (сутностей) та подій, що мають значення для системи. Це може бути засноване на аналізі бізнес-процесів та вимог до системи.
- Структурування матриці: Після ідентифікації основних елементів створюється структура матриці, де об'єкти розміщуються по рядках, а події – по стовпцях.
- Заповнення матриці: У кожну комірку матриці вноситься інформація про взаємодію між об'єктом та подією. Це може включати відмітки про те, що подія ініціює зміни в об'єкті, або впливає на його стан.
- Аналіз матриці: Після заповнення матриці проводиться аналіз для виявлення ключових залежностей та взаємодій між об'єктами та подіями. Це може включати визначення критичних подій, які впливають на багато об'єктів, або об'єктів, які є чутливими до великої кількості подій.

Таблиця 4.1 – Приклад матриці подія-сутність

Сутність / Подія	Зарахування	Призначення викладача	Оновлення курсу	Зміна режиму роботи
Студент	С	-	М	-
Вчитель	-	М	М	-
Бібліотека	-	-	-	М
Кафедра	М	М	М	-

На прикладі видно, яким чином кожна сутність змінюється під час виконання кожної події. С- це створення сутності, М – модифікація. Таким чином можна побачити, що подія “зарахування” створює сутність “студент” та модифікує сутність “кафедра”.

Матриця подій-об'єктів є цінним інструментом для архітекторів та розробників систем. Вона дозволяє глибше розуміти взаємодії та залежності в системі, що є критично важливим для проектування ефективних та надійних систем. Через свою здатність відображати складні взаємозв'язки між об'єктами та подіями, матриця допомагає виявити потенційні проблеми та розробити стратегії для їх вирішення. Також вона сприяє кращому розумінню процесів та може бути використана для оптимізації та підвищення ефективності системи.

4.1.3 Ефекти: Опис та властивості

Ефекти в методології EEM представляють собою важливу концепцію, яка описує зміни або вплив, який події чинять на сутності в рамках системи. Ефект в EEM - це не просто реакція сутності на подію, а скоріше детальний опис того, як подія впливає на атрибути або стан сутності, змінюючи її поведінку або властивості [7].

Ефекти в EEM часто описуються як частина специфікації події. Вони мають включати точне визначення того, що відбувається з сутністю або її атрибутами під впливом події. Наприклад, якщо ми розглядаємо сутність "Замовлення" у системі управління замовленнями, ефект події "Замовлення Підтверджено" може включати зміну статусу замовлення з "В обробці" на "Підтверджено".

При створенні ефектів є декілька характеристик, якими треба користуватись для біль повного опису:

- Чітке визначення: Кожен ефект повинен мати чітке та однозначне визначення. Важливо точно описати, які аспекти сутності змінюються та яким чином вони змінюються під впливом події.
- Відповідність з логікою системи: Ефекти повинні бути логічно сумісні з загальними правилами та принципами системи. Вони не повинні вводити протиріччя або непередбачені взаємодії, які можуть порушити цілісність системи.
- Узгодженість з визначеннями сутностей: Ефекти повинні відповідати визначенням та атрибутам сутностей, на які вони впливають. Наприклад, ефект не може змінити атрибут, який не існує у даній сутності.
- Документація та специфікації: Ефекти повинні бути належно задокументовані та включені в специфікації системи. Це забезпечує, що вони будуть враховані при проектуванні та реалізації системи.
- Перевірка на консистентність: Після визначення ефектів важливо перевірити їх на консистентність та відсутність суперечностей у контексті всієї системи. Це може включати аналіз впливу ефектів на поведінку системи та її відповідність очікуваному функціоналу.

Використання ефектів у EEM дозволяє аналітикам та розробникам системи створювати більш точні та реалістичні моделі, які відображають реальні взаємодії та поведінку сутностей в різних сценаріях. Це, в свою чергу, сприяє створенню ефективних, надійних та легко адаптованих систем, які точно відповідають бізнес-вимогам та потребам користувачів.

4.1.4 Entity Life History: Опис діаграми та правила користування

Entity Life History діаграми в методології Entity Event Modelling (EEM) є важливим інструментом, який дозволяє візуалізувати та аналізувати життєві цикли сутностей у межах системи. Ці діаграми детально показують, як різні

події впливають на сутності протягом часу, забезпечуючи глибоке розуміння динаміки та взаємодій, що відбуваються в системі.

ELH діаграми представляють собою послідовність станів сутності, крізь які вона проходить від моменту створення до завершення свого життєвого циклу. Кожен стан відображається графічно, зазвичай у формі вузлів або прямокутників, а переходи між станами показують, як сутність реагує на різні події або дії [6].

Для створення ELH діаграми, спочатку потрібно визначити ключові стани, через які проходить сутність протягом свого життєвого циклу. Наприклад, для сутності "замовлення" в системі управління замовленнями стани можуть включати "створено", "обробляється", "відправлено", "доставлено". Потім визначаються події, які викликають переходи між цими станами, наприклад, подія "підтвердження платежу" може перевести замовлення зі стану "створено" в "обробляється".

Коли ELH діаграма правильно побудована, вона має наступні властивості та кожна властивість має свою функцію у використанні:

- Візуальне відображення життєвого циклу: Діаграма надає візуальне уявлення про те, як сутність розвивається та змінюється відповідно до різних подій.
- Динамічне розуміння системи: Вона дозволяє зрозуміти не тільки статичну структуру даних, але й динамічні аспекти системи, відображаючи, як дані та сутності реагують на зовнішні подразники.
- Допомога у виявленні проблем та можливостей для оптимізації: Аналізування різних шляхів, якими може розвиватися життєвий цикл сутності, допомагає виявити потенційні проблеми або точки, де процеси можуть бути оптимізовані.
- Гнучкість у моделюванні: ELH діаграми можуть бути адаптовані до різних типів сутностей та складності систем, забезпечуючи універсальність у використанні.

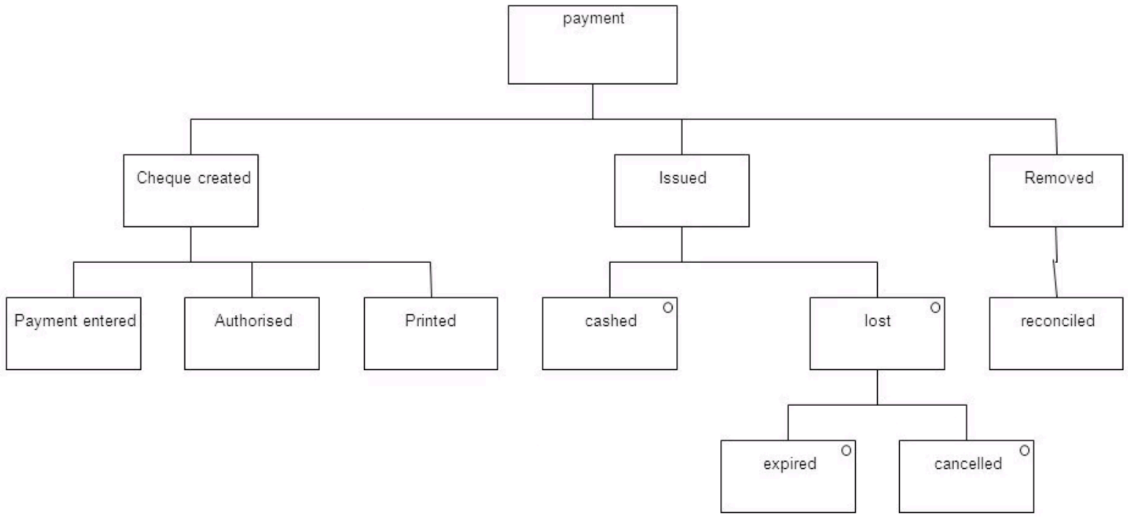

Рис. 4.1 – Приклад ELH діаграми

Як приклад розглянемо сутність "оплата" у системі. Стани можуть включати "проведена", "закінчена", "відмінена". Події, такі як "виданий", можуть переміщати оплату зі стану "створена" в "проведена", в той час як "встрата" може перемістити її в "відмінена". Ця діаграма допоможе зрозуміти, як взаємодії з оплатами впливають на їх статус у системі.

Загалом, ELH діаграми є потужним інструментом в EEM, що надає глибоке занурення у поведінку сутностей в системі та сприяє кращому розумінню та ефективній реалізації бізнес-процесів.

4.2 Доцільність реалізації CASE засобу

Розглядаючи доцільність реалізації CASE засобу, спеціально призначеного для підтримки методології Entity Event Modelling (EEM), стає зрозумілим, що такий підхід може відкрити значні можливості для проектування та аналізу систем баз даних. Важливість такого засобу полягає у його унікальній спроможності інтегрувати традиційне візуальне моделювання,

втілене у ERD діаграмах, з динамічним і глибоким аналізом, який пропонує EEM.

Об'єднання ERD і EEM у рамках єдиного CASE засобу створює потужний інструмент для розробників та аналітиків. ERD діаграми ефективно відображають структуру баз даних, представляючи сутності, їх атрибути та зв'язки. Коли ця статична структура поєднується з динамічним аналізом життєвих циклів сутностей, який надає EEM, розробники отримують глибше розуміння того, як дані взаємодіють та еволюціонують в реальному часі. Таке поєднання дозволяє не лише зобразити структуру даних, але й розуміти поведінку та логіку системи на більш глибокому рівні.

Більшість існуючих CASE засобів зосереджені на традиційному проектуванні баз даних, як-от створення ERD діаграм, але часто не враховують динамічні аспекти системи, такі як взаємодії подій і сутностей, що є ключовими в EEM. Це призводить до відсутності глибокого розуміння динаміки системи, що може бути критично важливим для розробки ефективних та надійних систем.

Реалізація CASE засобу, який включає підтримку EEM, відкриває шлях для розробки більш адаптивних та гнучких систем баз даних. Такий засіб дозволить розробникам:

- Ефективно моделювати динаміку даних: Шляхом інтеграції ERD та EEM розробники зможуть не лише створювати структурні моделі, але й аналізувати, як зміни в одній частині системи впливають на інші.
- Розширити аналітичні можливості: Новий інструмент дозволить аналітикам глибше занурюватися в аналіз бізнес-процесів, виявляючи потенційні проблеми або можливості для оптимізації.
- Підвищити якість та надійність систем: З розширеним розумінням взаємодій між подіями та сутностями, можна розробляти більш стабільні та надійні системи.

У підсумку, реалізація спеціалізованого CASE засобу для підтримки EEM є не тільки доцільною, але й необхідною для сучасних розробників і аналітиків систем. Такий інструмент забезпечить нові горизонти в проектуванні та аналізі систем баз даних, розкриваючи їх потенціал з нової, кращої сторони.

Розробка CASE засобу для підтримки методології EEM має велике значення для різних груп професіоналів у галузі IT, зокрема для архітекторів систем, розробників та тестувальників. Цей інструмент може надати кожній з цих груп унікальні переваги, що підвищують ефективність їхньої роботи та якість кінцевих продуктів.

Для архітекторів систем, CASE засіб, що інтегрує EEM, стане незамінним інструментом для проектування складних систем. Він дозволить:

- Глибше розуміти бізнес-процеси: Архітектори зможуть візуалізувати та аналізувати взаємодії між сутностями та подіями, що є критично важливим для створення ефективних архітектурних рішень.
- Проектувати більш надійні системи: Інструмент допоможе архітекторам прогнозувати можливі проблеми та конфлікти на ранніх етапах проектування, забезпечуючи створення більш стабільних та надійних систем.

Розробники програмного забезпечення виявлять, що CASE засіб для EEM значно полегшує процес кодування та реалізації систем. Він дозволить:

- Ефективніше моделювати та кодувати: Розробники зможуть швидко переходити від концептуальних моделей до реалізації коду, маючи чітке розуміння структури та поведінки системи.
- Зменшити час на розробку: Автоматизація та візуалізація складних аспектів системи може значно зменшити час, потрібний для розробки, та знизити ризик помилок.

Тестувальники знайдуть цей інструмент корисним для планування та виконання тестування систем. Він допоможе:

- Підготовити ефективні тестових сценарії: З розумінням взаємодій між подіями та сутностями, тестувальники можуть розробляти більш цілісні та всебічні тестові сценарії.
- Виявляти потенційні проблеми: Аналізуючи життєві цикли сутностей та їх взаємодії, тестувальники зможуть більш ефективно ідентифікувати потенційні вразливі точки та проблемні аспекти системи.

У підсумку, реалізація CASE засобу для ЕЕМ стане цінним вкладом у сферу розробки програмного забезпечення, пропонуючи інструмент, який розширює можливості аналізу, проектування та тестування складних інформаційних систем. Такий підхід не тільки збільшує ефективність робочих процесів, але й підвищує якість та надійність кінцевих продуктів.

5 РОЗРОБКА ВИМОГ ДО CASE ЗАСОБУ З ПРОЕКТУВАННЯ ERD ТА ДОДАТКОВИМИ МОЖЛИВОСТЯМИ EEM МЕТОДОЛОГІЇ

У цьому розділі дипломної роботи ми зосередимо увагу на визначенні та аналізі вимог до розробки CASE засобу, призначеного для підтримки методології EEM. Ретельне визначення вимог є ключовим етапом в процесі розробки будь-якого програмного продукту, а особливо – складного інструменту, який повинен відповідати потребам різних груп користувачів, включаючи архітекторів систем, розробників та тестувальників.

Важливість цього розділу полягає у створенні чіткого та структурованого списку вимог, який буде слугувати основою для подальшого процесу проектування та розробки CASE засобу. Вимоги повинні охоплювати різні аспекти, включаючи функціональність, користувацький інтерфейс, продуктивність, безпеку, інтеграцію з іншими системами, а також гнучкість та масштабованість системи.

Цей розділ також зосередиться на визначенні специфічних характеристик, які роблять CASE засіб ефективним для EEM. Це включає аналіз способів, якими інструмент може підтримувати моделювання ERD, візуалізацію життєвих циклів сутностей, обробку комплексних взаємодій між подіями та об'єктами, а також надання можливостей для глибокого аналізу системи.

Завершальним етапом цього розділу буде формулювання загальних та конкретних вимог, які допоможуть забезпечити, що розроблений CASE засіб буде не тільки функціональним та зручним у використанні, але й відповідатиме всім сучасним стандартам розробки програмного забезпечення, забезпечуючи високу якість та ефективність роботи з системами, базованими на EEM.

5.1 Бізнес вимоги

Бізнес вимоги до застосунку визначають основні очікування та необхідні характеристики програмного продукту з точки зору бізнесу. Ці вимоги є вирішальними для забезпечення того, що розроблений застосунок відповідає комерційним цілям організації, задовольняє потреби користувачів та сприяє досягненню бізнес-стратегій.

В нашому випадку ми можемо виділити ключову бізнес вимогу та охарактеризувати її наступним чином: покращення продуктивності процесу розробки.

Бізнес має потребу в оптимізації та покращенні продуктивності процесу розробки програмного забезпечення. Важливо, щоб CASE застосунок надавав інструменти, які дозволяють розробникам і архітекторам систем ефективно проектувати, аналізувати та модифікувати системи на основі EEM. Це має включати:

- Автоматизація завдань: Застосунок повинен надавати автоматизацію рутинних завдань, таких як генерація коду або документації, знижуючи тим самим часові витрати та ризик помилок.
- Інтерактивне моделювання: Інструмент має дозволяти швидко створювати та оновлювати моделі, включаючи ERD та діаграми життєвого циклу сутностей, що допоможе у візуалізації та кращому розумінні структури системи.
- Спільна робота та інтеграція: Застосунок має підтримувати інтеграцію з іншими інструментами та платформами, а також надавати можливості для спільної роботи команди, сприяючи кращій координації та обміну інформацією.

Ця вимога спрямована на зниження витрат та підвищення ефективності розробки, що, в свою чергу, призведе до скорочення часу виведення продукту на ринок і покращення якості кінцевого програмного продукту.

5.2 Функціональні вимоги

У розділі, присвяченому функціональним вимогам, ми зосередимо увагу на конкретних характеристиках та можливостях, які очікується від розроблюваного програмного продукту. Функціональні вимоги визначають ясно та точно, що саме програма повинна робити, які задачі вона має вирішувати та яким чином вона має взаємодіяти з користувачами чи іншими системами. Це основа для розробки будь-якого застосунку, оскільки саме функціональні вимоги формують основний функціонал та поведінку продукту.

Через те, що не усі функціональні вимоги будуть імплементовано в першій версії – є сенс створити таблицю функціональних вимог, та вказати на якій стадії проекту ця вимога буде виконана.

Таблиця 5.1 – Функціональні вимоги для CASE засобу

Ідентифікатор	Пріоритет	Версія	Функціональна вимога
FR-01	1	1.0	Дозвіл користувачам створювати облікові записи для доступа до платформи
FR-02	1	1.0	Можливість користувачам створювати та редагувати свій профіль
FR-03	1	1.0	Можливість створювати ERD діаграми
FR-04	1	1.0	Можливість додавати ELH для кожної сутності на діаграмі
FR-05	1	1.0	Можливість переглядати список усіх подій, які впливають на сутності

FR-06	1	1.0	Можливість переглядати усі події які впливають на одну сутність
FR-07	2	2.0	Можливість будувати матрицю сутність-подія
FR-08	2	2.0	Можливість будувати діаграму ELH для кожної сутності
FR-09	2	2.0	Можливість експортувати ERD та ELH діаграми

Після того, як усі ключову бізнес та функціональні вимоги було побудовано та приоритезовано – можемо побудувати Use Case діаграми, які дадуть нам візуальну картину того, хто та яким чином буде користуватись кожною функціональною частиною.

5.3 Діаграма використання системи

Діаграма використання є особливо важливою на ранніх етапах розробки, оскільки вона допомагає учасникам проекту (включаючи архітекторів систем, розробників, аналітиків бізнесу та кінцевих користувачів) отримати загальне уявлення про систему та її взаємодії [4]. Крім того, ця діаграма може слугувати основою для детальнішого аналізу та проектування, включаючи створення більш специфічних діаграм та моделей.

В нашому випадку системою будуть користуватись три типа користувачів:

- Адміністратор БД: Цей тип користувача має повний доступ до роботи системи. Він вміє створювати ERD діаграми, додавати сутності та зв’язки між сутностіми. Тобто цей користувач має можливість створювати, редагувати та видаляти усе, що пов’язано з

проектуванням баз даних. Також В нього є можливість додавати події до кожної сутності, переглядати усі події які вже є в системі.

- Розробник БД: Цей тип користувача має доступ до системи, але не має можливості редагувати ті описи БД та діаграми, які вже були створені. Цьому типу користувача необхідно тільки дивитись те, що архітектор БД створив, та переносити візуальне та схематичне відтворюванни БД на рівень коду.
- Тестувальник застосунку: Цей тип користувача використовує застосунок для кращого написання тест кейсів під час підготовки до тестування, або вже під час самого тестування. Вони не можуть нічого змінювати у створених діаграмах та описах БД, але мають необхідність переглядати усе, що пов’язано із сутностями, особливо з історією сутності.

Після того, як ми виділили основних користувачів та описали як саме кожен користувач може використовувати систему, можемо створити Use Case діаграми, для кращого розуміння.

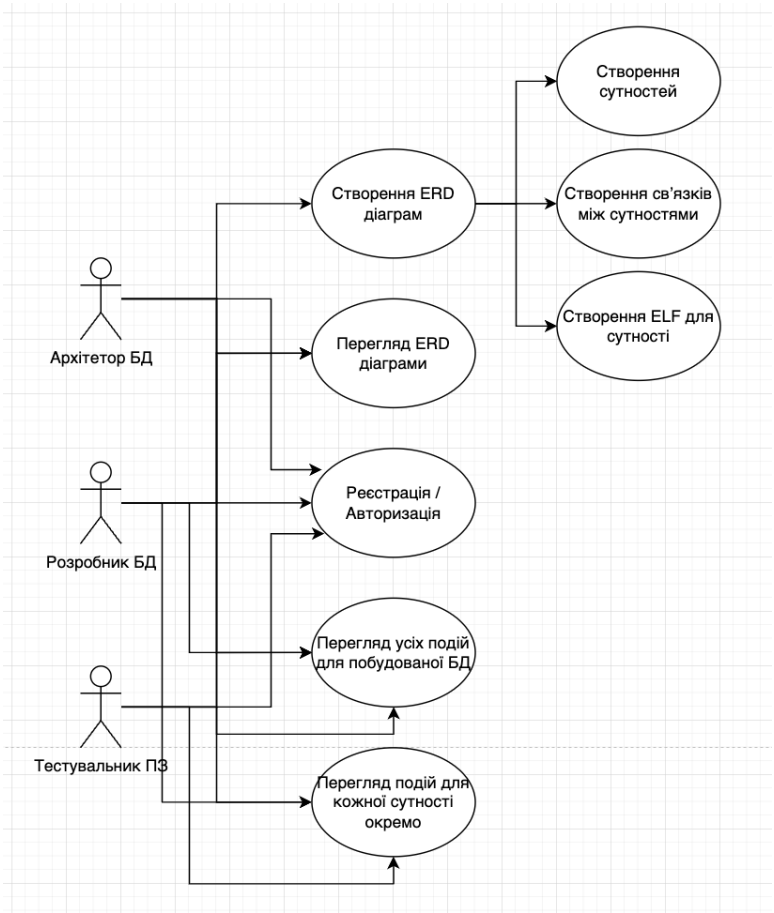

Рис. 5.1 – Use Case діаграма для різних типів користувачів

Після того, як ми побудували усе необхідне для побудови функціональної частина застосунку – залишилось описати нефункціональні вимоги до системі, адже вони є не менш важливими.

5.4 Нефункціональні вимоги до CASE застосунку

У цьому розділі, присвяченому нефункціональним вимогам, ми зосередимо увагу на аспектах розробки CASE застосунку, які перевищують безпосередні функціональні можливості, але є критично важливими для загальної ефективності, надійності та задоволення користувача. Нефункціональні вимоги включають в себе широкий спектр критеріїв, таких як

продуктивність, масштабованість, безпека, сумісність, надійність та зручність користування.

Нефункціональні вимоги відіграють ключову роль у забезпеченні, що система буде стабільною, легкою у підтримці, масштабованою та зручною для користувачів різного рівня. Вони також важливі для визначення, як система буде взаємодіяти з іншими системами, які міркування безпеки потрібно враховувати та які технічні обмеження необхідно врахувати.

Вимоги краще оформлювати у вигляді таблиці через те, що вони будуть приоритезовані та можуть бути виконані не в першій версії застосунку.

Таблиця 5.2 – Нефункціональні вимоги

Ідентифікатор	Пріоритет	Версія	Вимога
PR-01	1	1.0	Застосунок має працювати у браузері Chrome v.68 та вище
PR-02	1	1.0	Кількість ERD діаграм, система може зберігати має бути до 100 записів у БД
PR-03	1	1.0	Застосунок має використовувати захищений протокол спілкування з API HTTPS
PR-04	1	1.0	Система має зберігати від одного до 10 унікальних користувачів

PR-05	1	2.0	Застосунок має бути локалізованим на українську та англійську мови

Після того, як усі вимоги до застосунку було оброблено та описано ми можемо переходити до наступного розділу, де опишемо технології які було обрано для написання застосунку, та опис самого застосунку з прикладами використання.

6 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

6.1 Обгрунтування вибору технологій для побудови web-застосунку

Вибір розробки CASE застосунку як веб-додатку на мові JavaScript базується на низці стратегічних переваг, які цей підхід надає. По-перше, це підходить для сучасних тенденцій у розробці програмного забезпечення, де універсальний доступ і платформо-незалежність стають критично важливими. Веб-додатки, доступні через браузер, забезпечують легкий доступ до інструментів для широкого кола користувачів незалежно від їхнього обладнання та операційних систем, що є важливим для роботи в гетерогенних IT-середовищах.

По-друге, JavaScript як мова програмування надає значну гнучкість та ефективність у розробці веб-додатків. Завдяки своїй високій популярності та широкому використанню, JavaScript підтримується більшістю сучасних веб-браузерів, що робить його ідеальним вибором для створення крос-платформних рішень. Крім того, сучасні JavaScript-фреймворки та бібліотеки, такі як React, Angular чи Vue.js, пропонують потужні інструменти для розробки інтуїтивно зрозумілих та візуально привабливих користувацьких інтерфейсів.

Також важливим є той факт, що веб-додатки зазвичай легші у підтримці та оновленні. Оновлення можуть бути випущені і впроваджені безпосередньо на сервері, що прибирає потребу в ручному оновленні кожного клієнтського додатку. Це не тільки спрощує процес розгортання оновлень, але й забезпечує, що всі користувачі мають доступ до найновіших функцій і виправлень.

Враховуючи ці фактори, розробка CASE застосунку як веб-додатку на JavaScript видається логічним вибором. Це забезпечує широку доступність, гнучкість та ефективність, які є необхідними для створення сучасного, адаптивного та легко доступного інструменту для розробки та аналізу в рамках методології ЕЕМ.

Для спрощення написання візуальної частини Web-додатку ми використали бібліотеку ReactJS.

ReactJS, відомий також як React, є однією з найпопулярніших та широко використовуваних JavaScript бібліотек для розробки веб-інтерфейсів, особливо у сфері створення односторінкових застосунків (SPA). Розроблений Facebook, ReactJS вперше був представлений у 2013 році та з тих пір широко використовується розробниками по всьому світу завдяки своїй гнучкості, ефективності та зручності у використанні.

ReactJS пропонує декларативний підхід до розробки веб-інтерфейсів, що дозволяє розробникам легко створювати інтерактивні та динамічні веб-сторінки. Основна концепція React полягає у використанні компонентів - ізольованих, перевикористовуваних блоків коду, які управляють окремими частинами інтерфейсу. Кожен компонент містить свій власний стан (state) та логіку, що дозволяє створювати складні інтерфейси з простих та управлінних компонентів.

Однією з ключових переваг ReactJS є його виртуальний DOM (Document Object Model), який дозволяє оптимізувати взаємодію з браузером. Замість того, щоб безпосередньо модифікувати DOM при кожній зміні в інтерфейсі, React створює виртуальну копію DOM та обновляє тільки ті частини, які зазнали змін. Це призводить до значної ефективності, особливо у складних застосунках з великою кількістю динамічних змін.

React також підтримує JSX, розширення JavaScript, що дозволяє розробникам використовувати HTML-подібний синтаксис безпосередньо у своєму JavaScript-коді. Це робить компоненти React більш читабельними та зрозумілими, оскільки макет та логіка компоненту знаходяться поруч.

Іншою важливою особливістю React є можливість швидкої інтеграції з іншими бібліотеками та фреймворками, що робить його гнучким вибором для широкого спектра застосунків. React може бути легко інтегрований з такими

бібліотеками, як Redux для управління станами додатків або з React Router для навігації.

Використання ReactJS у розробці веб-застосунків не тільки сприяє створенню високоефективних та динамічних інтерфейсів, але й дозволяє розробникам сконцентруватися на бізнес-логіці та функціональності, замість витрачання часу на складні маніпуляції з DOM або управління станом компонентів. Отже, ReactJS вважається однією з найкращих технологій для розробки сучасних веб-застосунків.

6.2 Обгрунтування вибору для написання API та БД

Через те, що візуальна частина в нас написана на JS, є сенс обрати ту саму мову програмування для написання серверної частини.

Вибір Node.js для розробки серверної частини веб-застосунків можна вважати ключовим рішенням, яке обумовлене рядом переваг, що відповідають сучасним вимогам розробки програмного забезпечення. Node.js, який є крос-платформним JavaScript середовищем виконання, дозволяє використовувати JavaScript для написання серверного коду, що робить його вельми привабливим для проектів, де JavaScript вже використовується на фронтенді.

Однією з основних переваг Node.js є його асинхронна, неблокуюча модель вводу-виводу, яка дозволяє оптимально обробляти велику кількість одночасних запитів без необхідності створення додаткових потоків. Ця особливість робить Node.js ідеальним для розробки масштабованих мережевих застосунків, особливо у випадках, коли необхідно обробляти велику кількість одночасних запитів, таких як у веб-сервісах, API або реальному часі комунікаційних застосунках.

Використання однієї мови програмування на обох сторонах клієнта і сервера також спрощує розробку. Розробники можуть перевикористовувати код та розробляти більш однорідні та інтегровані системи. Це також сприяє

кращому спілкуванню між членами команди, оскільки всі використовують один і той же стек технологій.

Крім того, Node.js має велику та активну спільноту розробників, що означає наявність багатого набору модулів та бібліотек, доступних через npm (Node.js Package Manager). Це надає розробникам широкі можливості для швидкого та ефективного використання готових рішень, що може значно скоротити час розробки та зменшити витрати на створення функціональності "з нуля".

Загалом, використання Node.js для серверної частини надає значні переваги у вигляді підвищеної продуктивності, ефективності та гнучкості. Ці аспекти роблять Node.js відмінним вибором для розробки сучасних веб-застосунків, особливо в тих випадках, коли важливі швидкість відгуку системи та здатність ефективно обробляти велику кількість одночасних запитів.

Через те, що для написання серверної частини було обрано NodeJS, є сенс для БД обрати NoSQL СУБД через те, що цей тип СУБД широко використовується для подібного роду серверних застосунків.

Вибір MongoDB як бази даних для API, розробленого на Node.js, є логічним і продуктивним рішенням, оскільки обидва ці технологічні рішення добре взаємодоповнюють одне одного. MongoDB, як провідна NoSQL база даних, пропонує гнучкість у роботі з даними, яка особливо корисна у сучасних динамічних веб-додатках. Її нереляційний характер дозволяє легко зберігати та обробляти великі обсяги неструктурованих або напівструктурованих даних, які часто зустрічаються в інтернет-додатках.

Однією з ключових переваг MongoDB є її сумісність з JavaScript та Node.js. Вона використовує формат даних BSON, що є розширенням JSON. Це робить MongoDB ідеальною для роботи з Node.js, де JSON широко використовується для обміну даними між сервером та клієнтом. Розробники, які вже працюють з JavaScript, відчувають природний перехід до використання

MongoDB, оскільки вони можуть використовувати схожий синтаксис і підходи при роботі з даними.

Крім того, MongoDB ефективно підтримує асинхронні операції, які є основою Node.js. Це забезпечує гармонійну інтеграцію між сервером, що працює на Node.js, та базою даних, дозволяючи розробникам використовувати неблокуючі запити та асинхронний код для підвищення продуктивності застосунків.

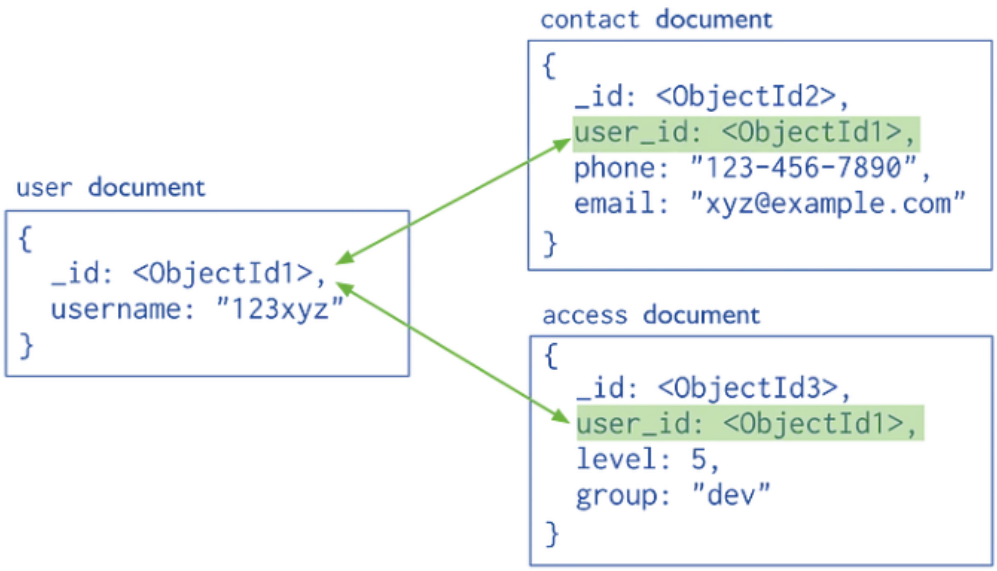

Рис. 6.1 – Приклад NoSQL схеми даних

Також, MongoDB пропонує потужні можливості для масштабування та розподілу даних, що є важливим для великих та швидко зростаючих веб-додатків. Її гнучка схема даних дозволяє легко адаптуватися до змін у структурі даних, що є частим явищем у динамічному світі веб-розробки.

У підсумку, інтеграція MongoDB з Node.js є сильним поєднанням для створення ефективних, швидких та масштабованих веб-додатків. Такий підхід не тільки спрощує розробку та забезпечує гнучкість у роботі з даними, але й відкриває двері до створення більш потужних та адаптивних рішень для сучасних веб-застосунків.

6.3 Опис структури розробленого CASE засобу та приклади використання

Важливість цієї частини полягає в створенні чіткої, логічної та користувацьки орієнтованої архітектури сторінок, яка впливає на загальне сприйняття та зручність користування застосунком. Акуратно спланована структура сторінок є ключовою для забезпечення інтуїтивно зрозумілого навігаційного потоку, ефективної взаємодії з користувачем та оптимального представлення контенту.

Для того, щоб застосунком було зручно користуватись, треба описати як сторінки будуть організовані, які елементи інтерфейсу будуть використані, та як вони взаємодіятимуть для створення зручного та ефективного користувацького досвіду. Також буде розглянуто, як ця структура сприятиме досягненню загальних цілей застосунку та як вона впливатиме на сприйняття та взаємодію користувачів з застосунком.

Проаналізувавши бізнес вимоги та функціональні вимоги, можемо намалювати наступну структурну схему CASE засобу:

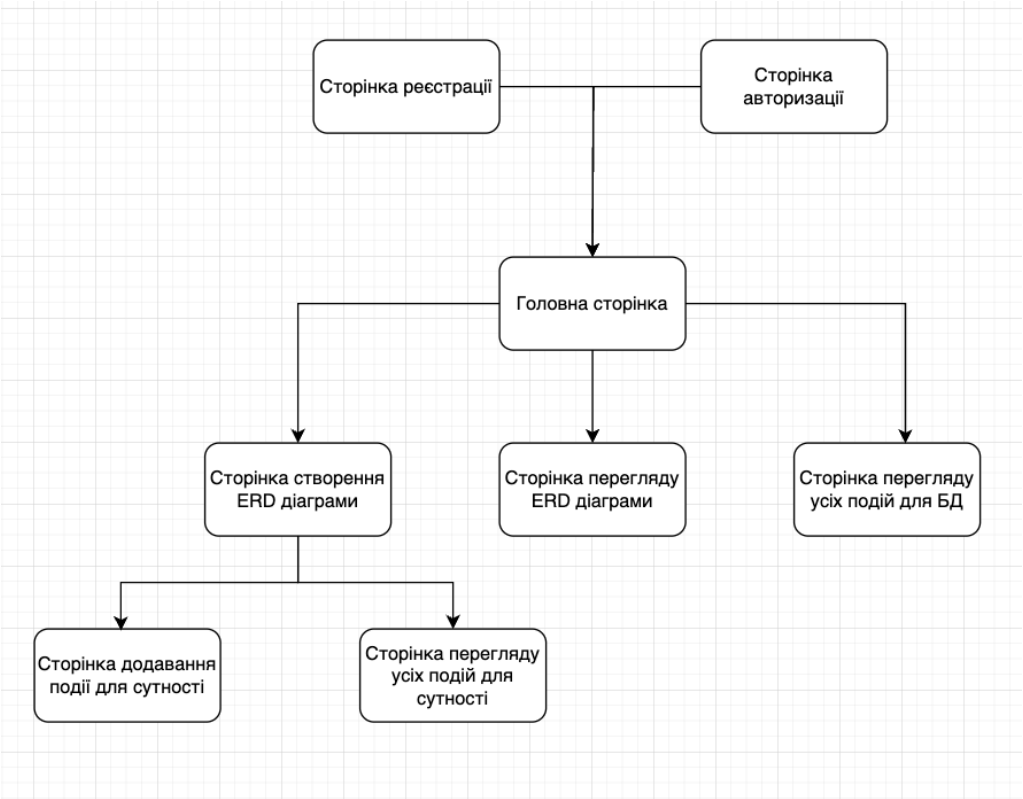

Рис. 6.2 – Структурна схема CASE засобу для формування ERD діаграми з додаванням EEM методології

Потрапляючи на головну сторінку застосунку у користувача є можливість обрати вже існуючу ERD діаграму, та продовжити роботу з нею, або почати створювати нову.

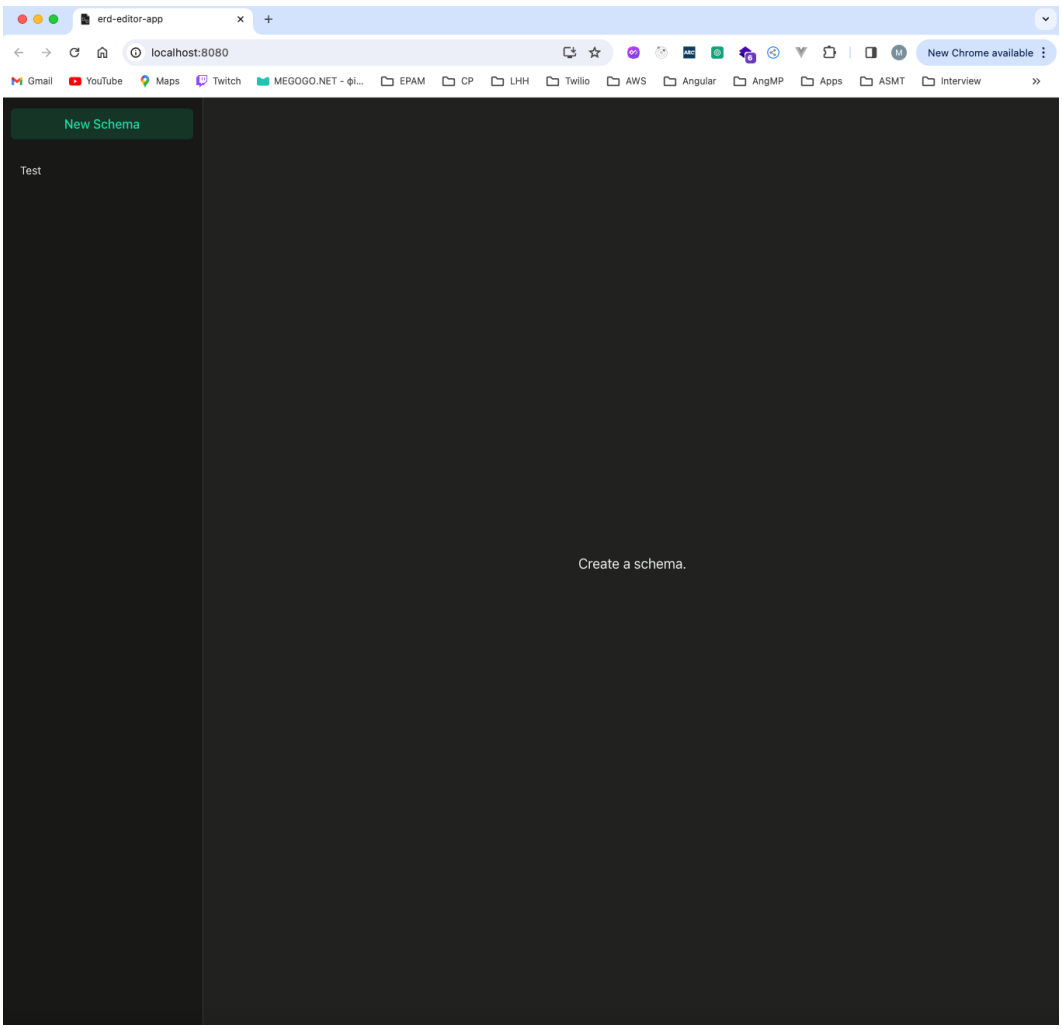

Рис. 6.3 – Головна сторінка застосунку для побудови ERD діаграми

Якщо обрати вже інсуючу діаграму, користувач потрапляє на сторінку ERD опису пази даних та може продовжити працювати з нею. Йде запит на сервер, який віддає повністю структуру збереженої діаграми, та застосунок вимальовує її на останньому етапі збереження.

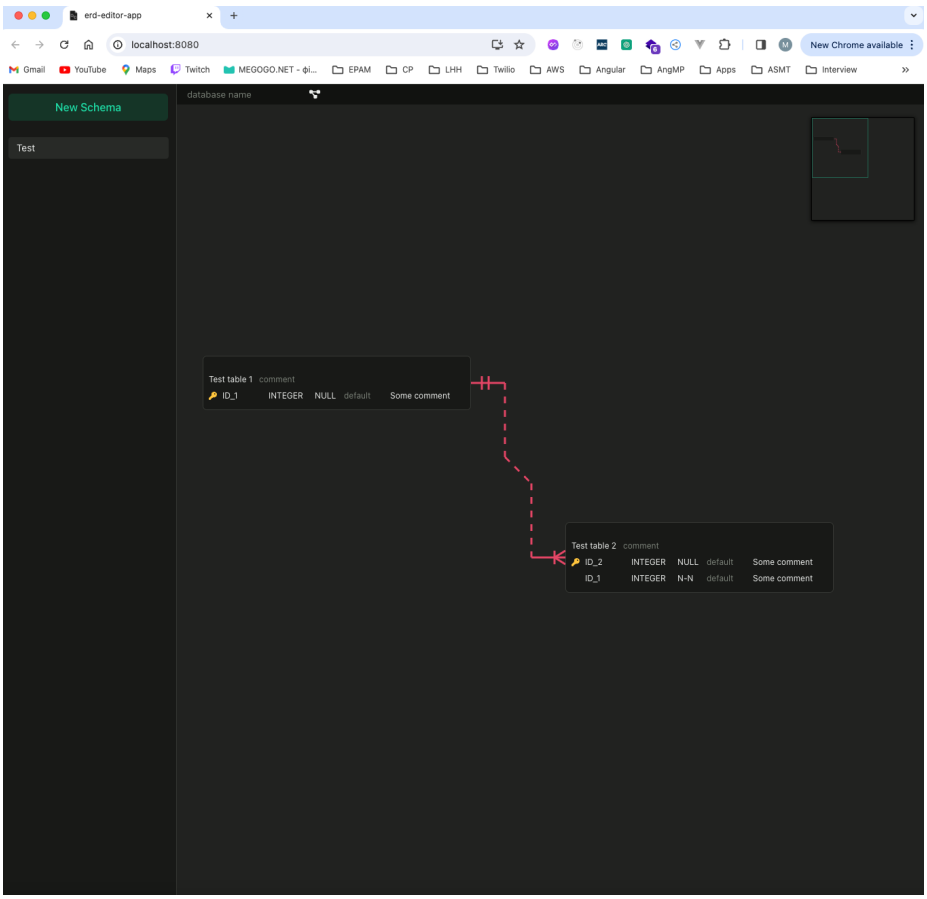

Рис. 6.4 – Вибір вже існуючої діаграми та продовження роботи з нею

Якщо користувач обирає створення нової діаграми, він потрапляє на пусту сторінку побудови діаграми, де може обрати назву БД та почати створювати першу сутність.

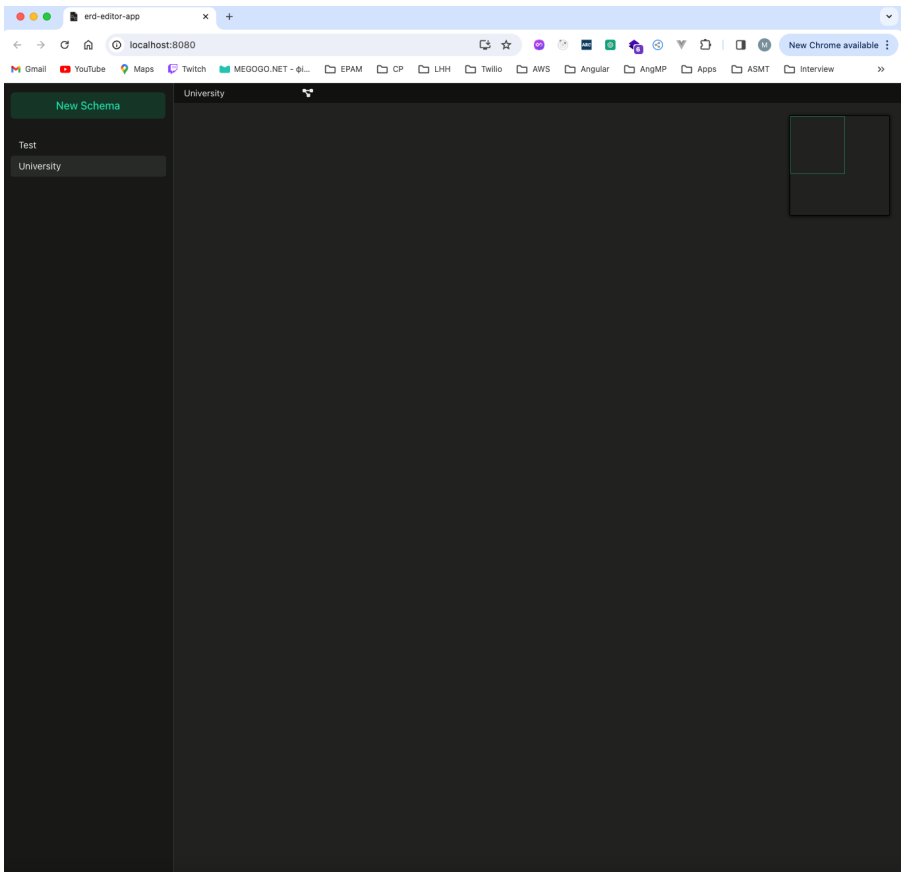

Рис. 6.5 – Сторінка щойно створеної БД та початок роботи з діаграмою

Наступним кроком буде створення самої діаграми. Це нам потрібно для того, щоб в нас з’явились сутності, для яких ми будемо додавати події. ERD діаграма з сутностями буде створена на основі бази даних університету. Оберемо декілька ключових сутностей:

- Студент;
- Курс, який студент буде проходити на початку року;
- Предмет з курсу, яких може бути декілька протягом навчального року;
- Тип оцінювання предмету, який може використовуватись для різних предметів.

Наш застосунок спочатку дає можливість описати ERD цієї структури для того, щоб була можливість працювати із сутносятми.

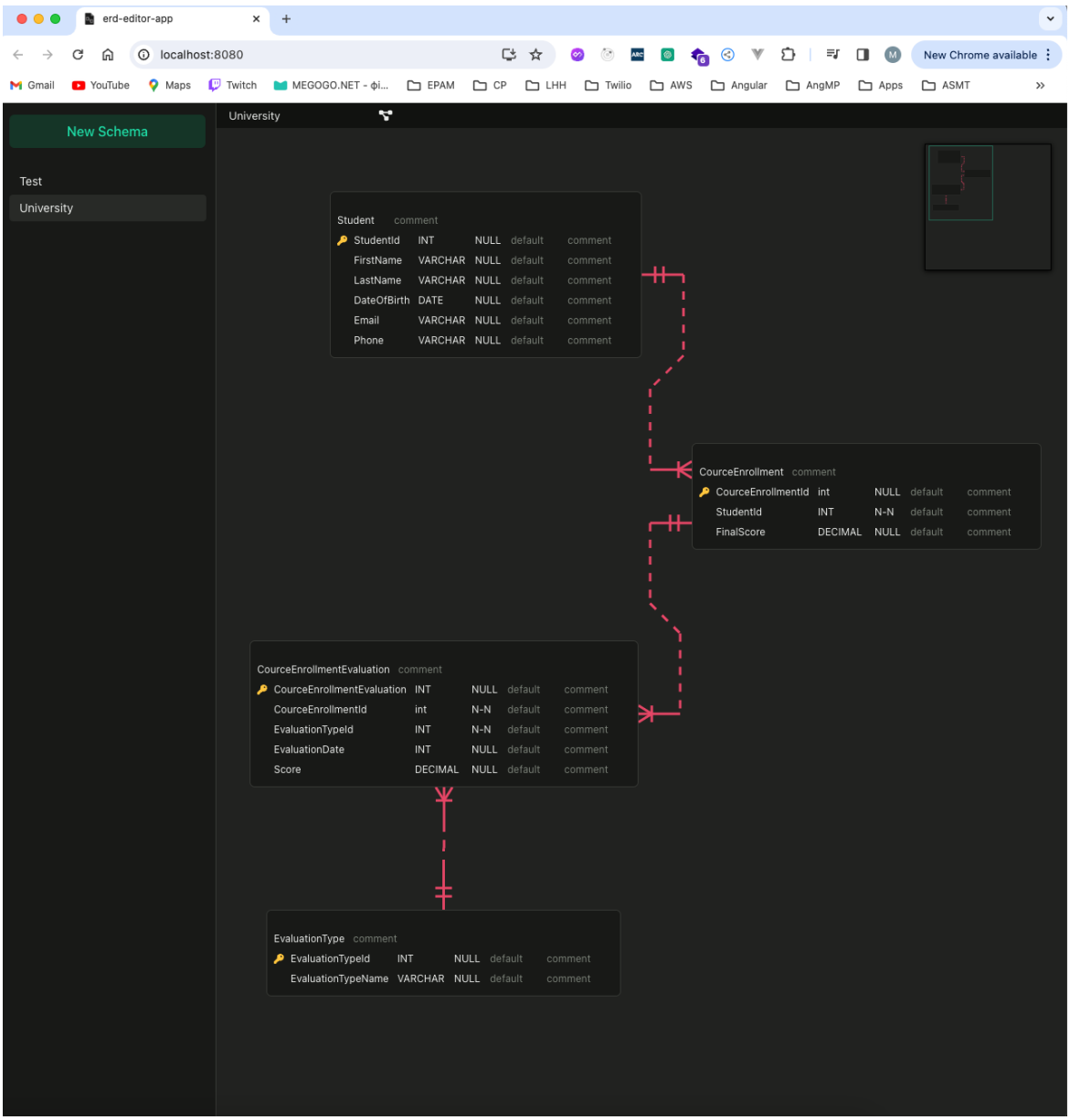

Рис. 6.6 – Створення ERD діаграми для роботи із сутностями

Після того, як сутності представлені у діаграмі у користувача з’являється можливість додавати події для кожної сутності вказуючи що саме ця подія робить із сутністю. Наприклад додамо подію – “Зарахування студента до університету”. Серез усіх цих сутностей ця подія впливає тільки на сутність – “Студент”, а саме створює її.

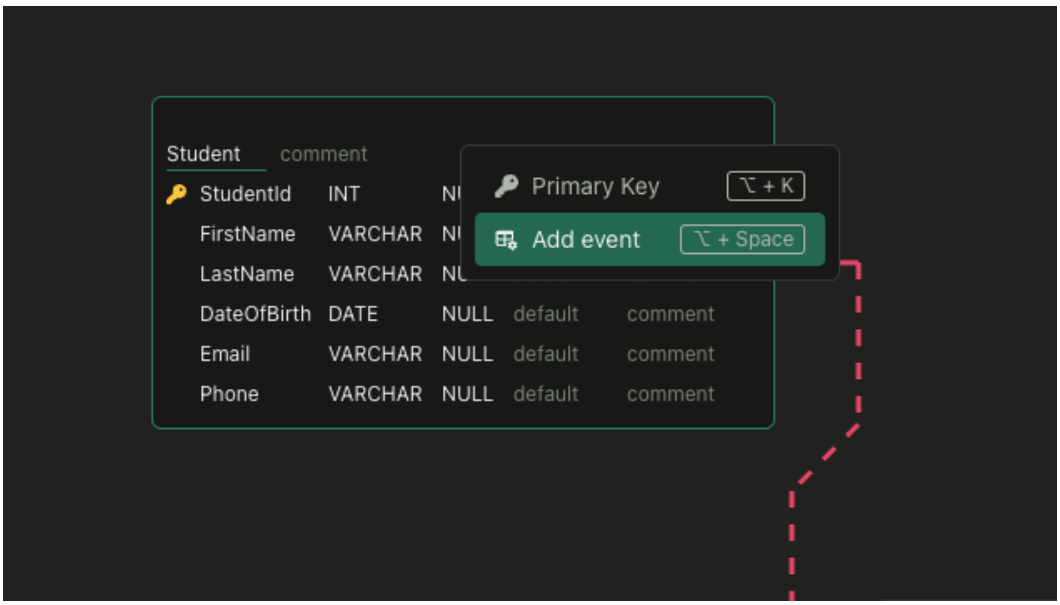

Рис. 6.7 - Контекстне меню для створення подій для сутності

Це контекстне меню викликає вікно, в якому є можливість описати саму подію. Подія складається з назви самої події, її опис та яким чином подія впливає на сутність. Опис події є корисною можливістю, для того щоб усім типам користувачів краще зрозуміти що саме ця подія робить та в який час вона викликається.

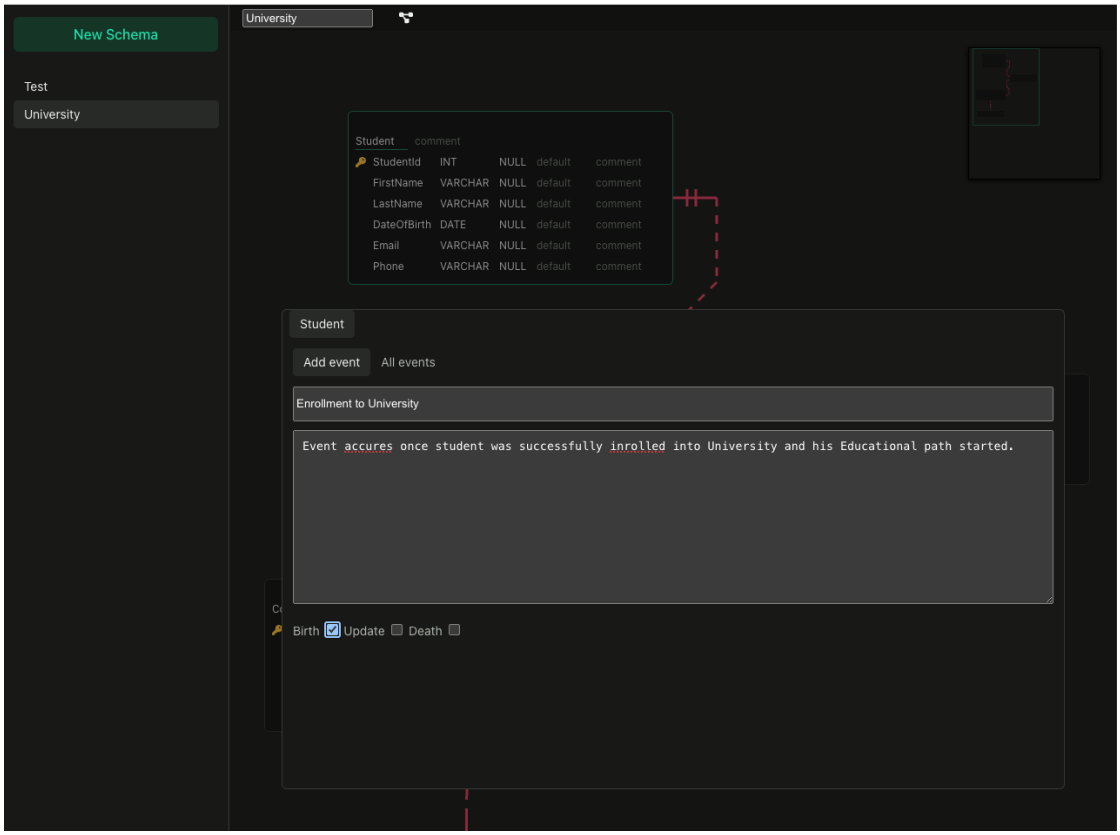

Рис. 6.8 – Створення події для сутності

Після того, як подію було створено, у користувача є можливість подивитись на список усіх подій, які було додано для цієї сутності. Розглянемо по одній події для кожного типу зміни сутності. Додамо ще подію – “Перехід на наступний курс”, яка робить зміну в сутності студента. Також додамо подію – “Закінчення університету”, яка не просто змінює сутність студента, вона викликає смерть цієї сутності системі, яка проектується.

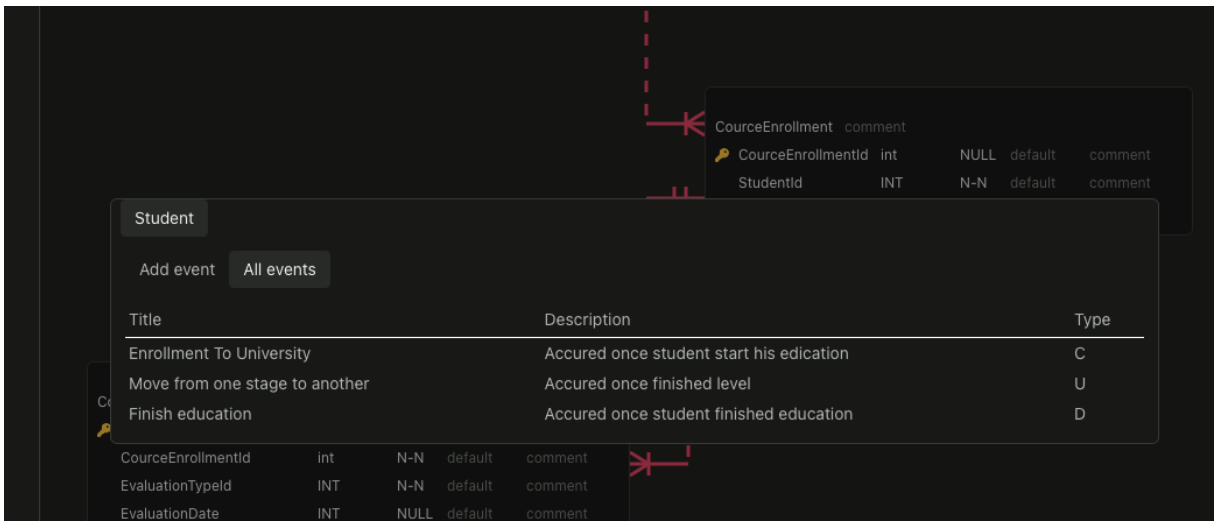

Рис. 6.9 – Список усій подій сутності

Крім цього у нашому застосунку є можливість подивитись список усіх подій, які було створено для усіх сутностей. Далі цей застосунок буде розвиватись, та будуть створюватись нові можливості. Наприклад можливість створювати матрицю Подія-Сутність, яку можна буде експортувати у вигляді таблиці для кращого аналізу інженерами, хто розробляє систему.

ВИСНОВКИ

Як результат цієї дипломної роботи підсумовується широкий спектр досліджень та розробок, виконаних у галузі роботи з базами даних, проектування систем за методологією Structured Systems Analysis and Design Method (SSADM) та застосування підходу Entity Event Modelling (EEM). Глибоке занурення в предметну область і ретельний аналіз існуючих методологій та підходів дозволили нам створити фундамент для розробки інноваційного CASE застосунку, який ефективно поєднує в собі роботу з ERD та EEM.

Ця робота виявила ряд ключових викликів та можливостей, пов'язаних з проектуванням та аналізом складних систем баз даних. Вивчення SSADM показало важливість структурованого підходу до аналізу та проектування систем, в той час як розгляд EEM надав цінних інсайтів щодо моделювання життєвих циклів сутностей та їхніх взаємодій. Інтеграція цих методологій у розробці CASE застосунку відкрила нові перспективи для аналітиків та розробників, надаючи їм потужний інструмент для ефективного проектування та аналізу.

Основним досягненням стала розробка CASE застосунку, який не тільки полегшує роботу з ERD, але й інтегрує методологію EEM, забезпечуючи глибше розуміння взаємозв'язків і динаміки в системах баз даних. Цей інструмент стає мостом між традиційними методами візуалізації структурних зв'язків у базах даних та більш динамічним та всебічним підходом до моделювання життєвих циклів сутностей.

Завдяки цьому застосунку аналітики та розробники тепер можуть більш ефективно ідентифікувати, аналізувати та управляти складними взаємодіями та процесами в системах. Вони мають змогу краще зрозуміти вплив різних подій на сутності та їхні стани, що є критично важливим для створення надійних, ефективних та адаптивних інформаційних систем.

В цілому, результати цієї роботи відкривають нові горизонти у сфері проектування та аналізу систем баз даних, пропонуючи інноваційні інструменти та підходи, які можуть значно покращити якість та ефективність розробки складних інформаційних систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ed Downs, Peter Clare, Ian Coe. Structured Systems Analysis and Design Method: Application and Context - Subsequent, 1992.
2. McGraw-Hill. SSADM Version 4: A Practical Approach – Version 3, 1995.
3. The Stationery Office. SSADM Foundation - Subsequent, 2000.
4. UML Class Diagram Relationships [Електронний ресурс] – Режим доступу до ресурсу: https://creately.com/blog/diagrams/class-diagram-relationships/.
5. Douglas Crockford. JavaScript: The Good Parts / O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2008. – 147p.
6. Entity Life Histories (ELH) [Електронний ресурс]. 21.11.2023. – Режим доступа: http://www.jacksonworkbench.co.uk/stevefergspages/papers/entity_event_modelling/index.html.
7. Modeling entities and events [Електронний ресурс]. 21.11.2023. – Режим доступа: https://www.ibm.com/docs/en/odm/8.9.0?topic=solutions-modeling-entities-events.
8. Alex Petrov: A Deep Dive into How Distributed Data Systems Work – Edition 1, 2019.
9. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems – Edition 1, 2017.
10. Neal Ford, Mark Richards. Fundamentals of Software Architecture: A Comprehensive Guide to Patterns, Characteristics, and Best Practices – Edition 1, 2020.
11. Технологія структурного системного анализу та автоматизованого проектування інформаційно-управляючих систем. Навч. пособник / Упоряд.: Е.Г. Петров, С.І. Чайніков, В.М. Левикін та інш. – Харків: ХНУРЕ,2004. - 73 с.