Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра автоматизованих систем управління

КУРСОВИЙ ПРОЕКТ

з навчальної дисципліни

“Проектування інформаційних систем”

на тему

“Розробка інформаційної системи “Соціальна мережа подій” ”

Спеціальність **6.122.00.00 «Комп’ютерні науки»**

Студент гр. КН-312 Крохмалюк Богдан Володимирович

Керівник: Ковівчак Я.В.

Курсова робота захищена з оцінкою

“ ”

“ ” 20 р.

Члени комісії:

Львів – 2019

Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра автоматизованих систем управління

З А В Д А Н Н Я

на курсовий проект

Студенту Крохмалюку Богдану Володимировичу групи КН-312

1. Тема роботи Розробка інформаційної системи “Соціальна мережа подій”
2. Термін здачі студентом роботи “ ” 20 р.
3. Вихідні дані до курсового проекту
4. Перелік питань, які підлягають розробці в курсовому проекті
5. Перелік графічного матеріалу

Дата видачі завдання “ ” 20 р.

Керівник роботи Ковівчак Ярослав Васильович

Завдання прийняв до виконання студент Крохмалюк Богдан Володимирович

Зміст

[ПЕРЕЛІК СКОРОЧЕНЬ 4](#_Toc10602874)

[ВСТУП 5](#_Toc10602875)

[1. ПОСТАНОВКА ЗАДАЧІ 7](#_Toc10602876)

[2. ТЕХНІЧНЕ ЗАВДАННЯ 9](#_Toc10602877)

[2.1 Загальний опис системи 9](#_Toc10602878)

[2.2 Опис предметної області з переліком вимог, що накладаються 9](#_Toc10602879)

[2.3 Перелік учасників системи 9](#_Toc10602880)

[2.4 Перелік компонентів системи 10](#_Toc10602881)

[3. ОГЛЯД ЛІТЕРАТУРИ 11](#_Toc10602882)

[3.1 Методології 11](#_Toc10602883)

[3.2 Case-методологія 12](#_Toc10602884)

[3.3 Спіральна модель життєвого циклу 14](#_Toc10602885)

[4. МОДЕЛЮВАННЯ ПРОЦЕСІВ 17](#_Toc10602886)

[4.1 Опис середовища моделювання 17](#_Toc10602887)

[4.2 Діаграми IDEF0 18](#_Toc10602888)

[4.3 Діаграма DFD 22](#_Toc10602889)

[4.4 Діаграма IDEF3 23](#_Toc10602890)

[5. КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ 24](#_Toc10602891)

[5.1 Діаграма IDEF1x 24](#_Toc10602892)

[5.2 UML Діаграми 25](#_Toc10602893)

[5.2.1 Діаграма прецедентів 25](#_Toc10602894)

[5.2.2 Діаграма послідовностей 30](#_Toc10602895)

[5.2.3 Діаграма класів 31](#_Toc10602896)

[5.3 Вартісний аналіз 32](#_Toc10602897)

[6. ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКА БАЗИ ДАНИХ 33](#_Toc10602898)

[ВИСНОВОК 36](#_Toc10602899)

[СПИСОК ЛІТЕРАТУРИ 37](#_Toc10602900)

# ПЕРЕЛІК СКОРОЧЕНЬ

* ІС – інформаційна система;
* БД – база даних;
* ЖЦ – життєвий цикл;
* СУБД – система управління базою даних;
* ООП – об’єктно-орієнтований підхід.

# ВСТУП

Тенденції розвитку сучасних інформаційних технологій спричиняють постійне зростання складності інформаційних систем, які створюються у різноманітних областях людської діяльності. Сучасні великі проекти ІС характеризуються, як правило наступними особливостями:

* складність опису (велика кількість функцій, процесів, елементів даних і складні взаємозв'язки між ними), що вимагає ретельного моделювання й аналізу даних і процесів;
* наявність сукупності компонентів (підсистем), що знаходяться у тісній взаємодії, виконують певні локальні задачі і цілі функціонування (наприклад додатків, пов'язаних з обробкою трансакцій і рішенням регламентних задач, додатків аналітичної обробки (підтримки прийняття рішень), які використовують нерегламентовані запити до даних великого обсягу);
* відсутність прямих аналогів, що обмежує можливість використання типових проектних рішень і прикладних систем;
* необхідність узгодження існуючих додатків з новими розробками;
* функціонування в неоднорідному середовищі на декількох апаратних платформах;
* різнорідність рівня кваліфікації і сформованих традицій використання певних наборів інструментальних засобів у групах розробників;
* істотна тривалість проекту - обумовлена, з одного боку, обмеженими можливостями колективу розробників; з іншого боку, масштабами організації-замовника і різним ступенем готовності окремих її підрозділів до впровадження ІС.

Для успішної реалізації проекту об'єкт проектування повинен бути насамперед адекватно описаний, побудовані повні і несуперечливі функціональні та інформаційні моделі ІС. Накопичений на даний час досвід проектування ІС показує, що це складна, трудомістка і тривала за часом робота, що вимагає високої кваліфікації фахівців, які беруть участь у ній. Однак донедавна проектування ІС виконувалося в основному на інтуїтивному рівні з застосуванням неформалізованих методів, які базуються на мистецтві, практичному досвіді, експертних оцінках і дорогих експериментальних перевірках якості функціонування ІС. Крім того, у процесі створення і функціонування ІС інформаційні потреби користувачів можуть змінюватися або уточнюватися, що ще більше ускладнює розробку і супровід таких систем.

# 1. ПОСТАНОВКА ЗАДАЧІ

Сьогодні існує досить багато інформаційних систем для пошуку та перегляду подій (Concert.ua, Karabas.com, Facebook events та інші), проте у кожній ІС ми можемо побачити велику кількість недоліків:

* Пошук подій – при пошуку подій у цих системах реалізовано знаходження подій за подібними словами, категорією, та популярністю. Великим недоліком є те що досить складно побачити всі події які відбуваються в тому чи іншому місті за часовою шкалою та місцевістю. Вирішенням цієї проблеми може стати пошук подій на певній місцевості, та за проміжком часу. Уявіть ви відкрили свій смартфон і думаєте яку ж подію відвідати, проте у вас список подій в якому вигубитесь. Альтернативне рішення – у вас події відображені на вибраній вами території у вибраний вами час і ви можете зразу побачити що і де відбувається саме тоді коли вам зручно а не постфактум.
* Кількість подій – при пошуку подій на тій чи іншій площадці, ви завжди можете побачити різні події, тобто кожна площадка просуває свої події, на яких вона може заробити. З точки зору користувача не зручно шукати якісь події на кількох середовищах. Тому можна додати всі події з найпопулярніших площадок в одну базу даних і при пошуку події користувач зможе бачити на якій площадці вона створена і де він може купити квиток, при цьому різні події від різних площадок відображатимуться в єдиній програмі.
* Можливість створювати події – така можливість реалізована тільки в Facebook events і в неї теж є обмежений функціонал. Причиною цієї проблеми, як і попередньої є те що кожний представник хоче заробити як найбільше – тому якщо ви не відома людина, вам важко створити подію на популярній площадці та тим більше заробити на ній. Вирішенням цієї проблеми може стати введення можливості створення події і дозволити робити як комерційні так і повністю безкоштовні події, як приватні так і публічні.
* Розповсюдження інформації про подію – в даних інформаційних системах практикується виведення популярних подій за стратегією “все на кучу”, тобто користувач бачить всі події які популярні не залежно від його вподобань. Користувач повинен бачити >75% саме тих подій в яких він зацікавлений, тобто програма має аналізувати смаки користувача, переглянуті та відвідані події і передбачати на які події він чи вона хотіли б відвідати, саме ці події і виставляти в розділі рекомендовані.

**Мета курсової роботи:** Створити проектування універсального, зручного та багатофункціонального додатку для можливості пошуку та створення подій.

# 2. ТЕХНІЧНЕ ЗАВДАННЯ

## 2.1 Загальний опис системи

Система перегляду та створення подій, реалізована на платформі мобільного додатку і/або веб сервісу. Сервіс повинен давати можливість користувачам шукати події, та створювати їх та відображати в публічному/приватному доступі. Система повинна уникнути проблем які існують у сучасних платформах та сервісах подій.

## 2.2 Опис предметної області з переліком вимог, що накладаються

Можуть існувати такі вимоги та обмеження щодо предметної області:

* Не повинно бути двох однакових подій;
* Ідентифікатором користувача слугує електронна адреса;
* Користувач може створювати безліч подій;
* В БД архівуються застарівші події часом на один рік, після чого видаляються;
* Користувач має право вносити правки в його подію.

Таких умов може бути і більше, вони можуть бути іншими або їх взагалі може не бути, в залежності від глибини аналізу даної області, відтвореної в базі даних

## 2.3 Перелік учасників системи

Працювати з даною інформацєю будуть наступні учасники:

* Користувач;
* Модератор;

Користувач – особа що може створювати, редагувати, видаляти власні події. Та шукати, записуватись, купувати квитки на чужі події. Користувач може бачити інформацію про локацію, час, опис, та графічні матеріали будь-якої публічної подї, та будь якої приватної події, до якої він має доступ.

Модератор – особа що відповідає за видалення подій неприйнятного характеру, та блокування підозрілих користувачів.

## 2.4 Перелік компонентів системи

**Програмне забезпечення** — інформація, що визначає поведінку інформаційної системи.

**Апаратне забезпечення** — комплекс технічних засобів, необхідних для функціонування інформаційної системи.

Таким чином, **інформаційна система має дві складові** — програмне та апаратне забезпечення.

Інформаційна система взаємодіє з іншими системами, надсилаючи та отримуючи інформацію. Вона спрямовує запити до джерел інформації й отримує у відповідь необхідні дані, натомість споживачі самі надсилають до інформаційної системи запити. Система обробляє їх і надає споживачам відповіді.

Інформаційна система, як система управління, тісно пов’язується, як з системами збереження та видачі інформації, так і з іншої - з системами, що забезпечують обмін інформацією в процесі управління. Вона охоплює сукупність засобів та методів, що дозволяють користувачу збирати, зберігати, передавати і обробляти відібрану інформацію.

# 3. ОГЛЯД ЛІТЕРАТУРИ

## 3.1 Методології

Будемо порівнювати найпопулярніші методології проектування Case-технологію та об’єктно-орієнтований підходу(ООП).

Перша відмінність цих підходів один від одного полягає в принципах декомпозиції та структурної організації елементів (компонентів, модулів) системи. Згідно з цими принципами система являє собою структуру, що складається з чітко виражених модулів, пов’язаних між собою певними відносинами. Вид декомпозиції ООП – об’єктно-орієнтована. В рамках цього підходу система розбивається на набір об’єктів, відповідних об’єктам реального світу, взаємодіючих між собою шляхом посилки повідомлень.

Другою відмінністю є об’єднання в об’єкті як атрибутивних даних (характеристики, властивості), так і поведінки (функції, методи). У функціонально-орієнтованих системах функції і дані зберігаються (існують) окремо.

Третя відмінність двох підходів полягає в структурній організації всередині модулів системи. У структурному підході модуль складається з функцій, ієрархічно пов’язаних між собою відношенням композиції, тобто функція складається з підфункцій, підфункція з підпідфункцій і т.д. В об’єктно-орієнтованому підході ієрархія вибудовується з використанням двох типів відношень: композиції і спадкування (англ. IS A – це є). При цьому в об’єктно-орієнтованому підході «об’єкт-частина» може включатися відразу в кілька «об’єктів-ціле». Таким чином, модуль в структурному підході представляється у вигляді дерева, а в об’єктно-орієнтованому підході – у вигляді орієнтованого графа, тобто за допомогою більш загальної структури.

Найбільш популярними методологіями, що підтримують об’єктно-орієнтований підхід, зараз є: уніфікований процес (Unified Process, UP); екстремальне програмування (eXtreme Programming, XP); гнучке моделювання (Agile Modeling, AM). Базовим засобом фіксації (документування) результатів проектування систем за допомогою цих методологій є уніфікована мова моделювання (Unified Modeling Language, UML).

## 3.2 Case-методологія

CASE-технологія являє собою методологію проектування інформаційних систем, набір методів, нотацій та інструментальних засобів, що дозволяють у наочній формі моделювати предметну область, аналізувати модель системи на всіх етапах розробки та супроводу системи і розробляти додатки відповідно до інформаційних потреб користувачів.

Під нотацією будемо розуміти встановлені способи відображення елементів системи, тобто графи, таблиці, блок-схеми, формальні і природні мови. Парадигма – вихідна концептуальна схема (модель) постановки проблеми та її рішення. Моніторінг – комплексна система спостереження, контролю, оцінки і прогнозу явищ, процесів, об’єктів і т. п. Як інструментарій реалізації технології використовуються CASE-засоби, основними функціями яких є:

1. Централізоване зберігання в єдиній базі даних проекту (репозиторії) інформації про інформаційну систему протягом всього життєвого циклу. Репозиторій може зберігати об’єкти різних типів: діаграми, визначення екранів і меню, проекти звітів, опис даних, логіку їх обробки, вихідні коди програм і т.п.
2. Пряме проектування програмного забезпечення та баз даних. При цьому порядок використання розробниками CASE-засоби такий: створюється логічна модель системи; вибирається конкретна мова програмування або СУБД для побудови фізичної моделі, після чого CASE-засіб автоматично створює фізичну модель системи; допрацьовується фізична модель; виконується автоматична генерація тексту програми або структури бази даних на диску;
3. Зворотне проектування (реінжиніринг). У цьому випадку порядок використання CASE-засоби зворотний – від тексту програми або бази даних на диску до логічної моделі. Крім побудови, CASE-засоби дозволяють швидко інтегрувати отримані таким чином моделі в проект, а також з меншими втратами переходити від однієї фізичної реалізації до іншої.
4. Синхронізація моделей системи з її фізичною реалізацією. У разі зміни моделі системи можуть бути автоматично внесені необхідні зміни в фізичну реалізацію або навпаки.
5. Автоматичне забезпечення якості і тестування моделей на наявність помилок (наприклад, помилок нормалізації БД), повноту і несуперечність.
6. Автоматична генерація документації. Вся документація по проекту генерується автоматично на базі репозиторія (як правило, відповідно до вимог діючих стандартів).

Безсумнівна перевага CASE-технології полягає в тому, що документація завжди відповідає поточному стану справ, оскільки будь-які зміни в проекті автоматично відображаються в репозиторії.

Основна мета використання Case-технологій полягає в максимальній автоматизації стадій аналізу і проектування системи з метою побудови формальних і несуперечливих моделей системи. Інша, не менш важлива, мета використання CASE-технологій – винесення частині діяльності зі стадії кодування в стадію проектування. Більшість сучасних CASE-засобів підтримує методології структурного та / або об’єктно- орієнтованого аналізу і проектування інформаційних систем. Вибір того чи іншого підходу (парадігми) має на увазі дотримання його і на стадії кодування (згідно з принципом концептуальної спільності). Їх відмінність один від одного полягає у виборі способу декомпозиції системи (завдання). Якщо за основу приймається функціональна (алгоритмічна) декомпозиція, то мова йде про структурний підхід, якщо об’єктна – про об’єктно-орієнтований. Вибір того чи іншого підходу залежить від специфіки розв’язуваної задачі. Як правило, структурний підхід застосовується для автоматизації завдань, що оперують великими обсягами "пасивних" даних і орієнтованих на використання реляційних баз даних (наприклад, облік, збір статистики, математичні та інженерні розрахунки, аналіз даних). Об’єктно-орієнтований підхід в основному орієнтований на вирішення завдань, в яких чітко простежується поділ системи на взаємодіючі між собою сутності (наприклад, імітаційне моделювання, управління технічними об’єктами або технологічними процесами, моніторинг). Найбільш характерна ця особливість для розподілених систем. Сучасні засоби програмування і управління БД в переважній більшості забезпечують можливість як структурного (процедурного, функціонального), так і об’єктно-орієнтованого програмування. Від розробників залежить, як використовувати ці можливості. Одне можна точно стверджувати, що при побудові інтерфейсу систем остаточно "переміг" об’єктно- орієнтований підхід. Його вже давно не програмує, а "малюють" за допомогою засобів візуальної розробки. При цьому кожен елемент інтерфейсу (поле введення, командна кнопка, перемикачі, таблиці і т. д.) є об’єктом з властивостями, методами та подіями. При програмуванні бізнес- логіки, зберігання і обробки великих обсягів даних методи і засоби структурного підходу ще довго будуть знаходити своє застосування.

## 3.3 Спіральна модель життєвого циклу

Найбільш відомим і поширеним варіантом еволюційної моделі є спіральна модель, що стала вжефактично самостійної моделлю, що має різні сценарії розвитку і деталізації.

Спіральна модель була вперше сформульована Баррі Боем в 1988 р Відмінною особливістю цієї моделі є спеціальна увага ризикам, що впливає на організацію життєвого циклу.

Головне досягнення спіральної моделі полягає в тому, що вона пропонує спектр можливостей адаптації вдалих аспектів існуючих моделей процесів життєвого циклу.

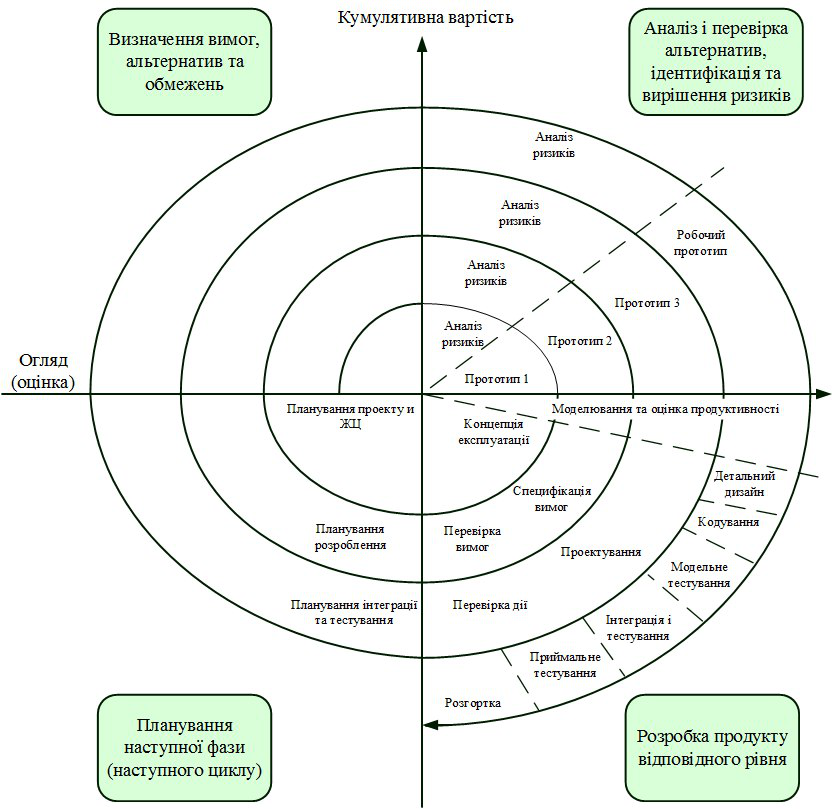
Дана модель життєвого циклу характерна при розробці новаторських (нетипових) систем. На початку роботи над проектом у замовника і розробника немає чіткого бачення підсумкового продукту (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості в успішній реалізації проекту (ризики дуже великі). В зв’язку з цим приймається рішення розробки системи по частинах з можливістю зміни вимог або відмови від її подальшого розвитку. Розвиток проекту може бути завершено не тільки після стадії впровадження, але і після стадії аналізу ризику.

Рис. 1 Спіральна модель ЖЦ

Переваги моделі:

* Дозволяє швидше показати користувачам системи працездатний продукт, тим самим, активізуючи процес уточнення і доповнення вимог;
* Допускає зміну вимог при розробці інформаційної системи, що характерно для більшості розробок, у тому числі і типових;
* Забезпечує більшу гнучкість в управлінні проектом;
* Дозволяє отримати більш надійну і стійку систему. По мірі розвитку системи помилки і слабкі місця виявляються і виправляються на кожній ітерації;
* Дозволяє удосконалювати процес розробки – аналіз, проведений в кожній ітерації, дозволяє проводити оцінку того, що має бути змінено в організації розробки, і поліпшити її на наступній ітерації;
* Зменшуються ризики замовника. Замовник може з мінімальними для себе фінансовими втратами завершити розвиток неперспективного проекту.

Недоліки моделі:

* Збільшується невизначеність у розробника в перспективах розвитку проекту. Цей недолік випливає з попереднього достоїнства моделі;
* Ускладнені операції тимчасового і ресурсного планування всього проекту в цілому. Для вирішення цієї проблеми необхідно ввести тимчасові обмеження на кожну із стадій життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота виконана. План складається на основі статистичних даних, отриманих у попередніх проектах та особистого досвіду розробників.

# 4. МОДЕЛЮВАННЯ ПРОЦЕСІВ

## 4.1 Опис середовища моделювання

Функціональні можливості інструментальних засобів структурного моделювання ділових процесів будуть розглянуті на прикладі case-засобу AllFusion Process Modeler (Bpwin). Bpwin підтримує три методології моделювання: функціональне моделювання (IDEF0); опис бізнес-процесів (IDEF3); діаграми потоків даних (DFD).

Bpwin має досить простий і інтуїтивно зрозумілий інтерфейс користувача. При запуску AllFusion Process Modeler за замовчуванням з'являється основна панель інструментів, палітра інструментів (вид якої залежить від обраної нотації) і, у лівій частині, навігатор моделі – Modelexplorer.

Під час створення нової моделі виникає діалог, у якому слід вказати, чи буде модель створена заново або вона буде відкрита з файлу або з репозиторію Modelmart, потім ввести ім'я моделі й вибрати методологію, у якій буде побудована модель.

Як було зазначено вище, AllFusion Process Modeler підтримує три методології IDEF0, IDEF3 і DFD, кожна з яких вирішує свої специфічні завдання. В AllFusion Process Modeler можлива побудова змішаних моделей, тобто модель може містити одночасно діаграми як IDEF0, так і IDEF3 і DFD. Вміст палітри інструментів змінюється автоматично, коли відбувається перемикання з однієї нотації на іншу.

Модель в AllFusion Process Modeler розглядається як сукупність робіт, кожна з яких оперує з деяким набором даних. Робота зображається у вигляді прямокутників, дані - у вигляді стрілок. Якщо клацнути по будь-якому об'єкту моделі лівою кнопкою миші, з'являється контекстне меню, кожний пункт якого відповідає редактору деякої властивості об'єкта.

## 4.2 Діаграми IDEF0

Контекстна діаграма є вершиною деревоподібної структури діаграм і являє собою найбільш загальний опис системи і її взаємодії із зовнішнім середовищем. Для створення діаграми пошуку оптимальної вартості рецепту, використав стрілки входу (База даних квитків з різних сторонніх платформ, Заявки про замовлення квитків, Аккаунти учасників систем), виходу ( Квитки, Події), керування (Алгоритм пошуку квитків, Обмеження програмних засобів, Алгоритми обробки транзакцій, Законодавство, Правила створення подій) та механізму (Учасник ІС, Модератор ІС) та за допомогою властивостей стрілок виконав відповідні підписи.

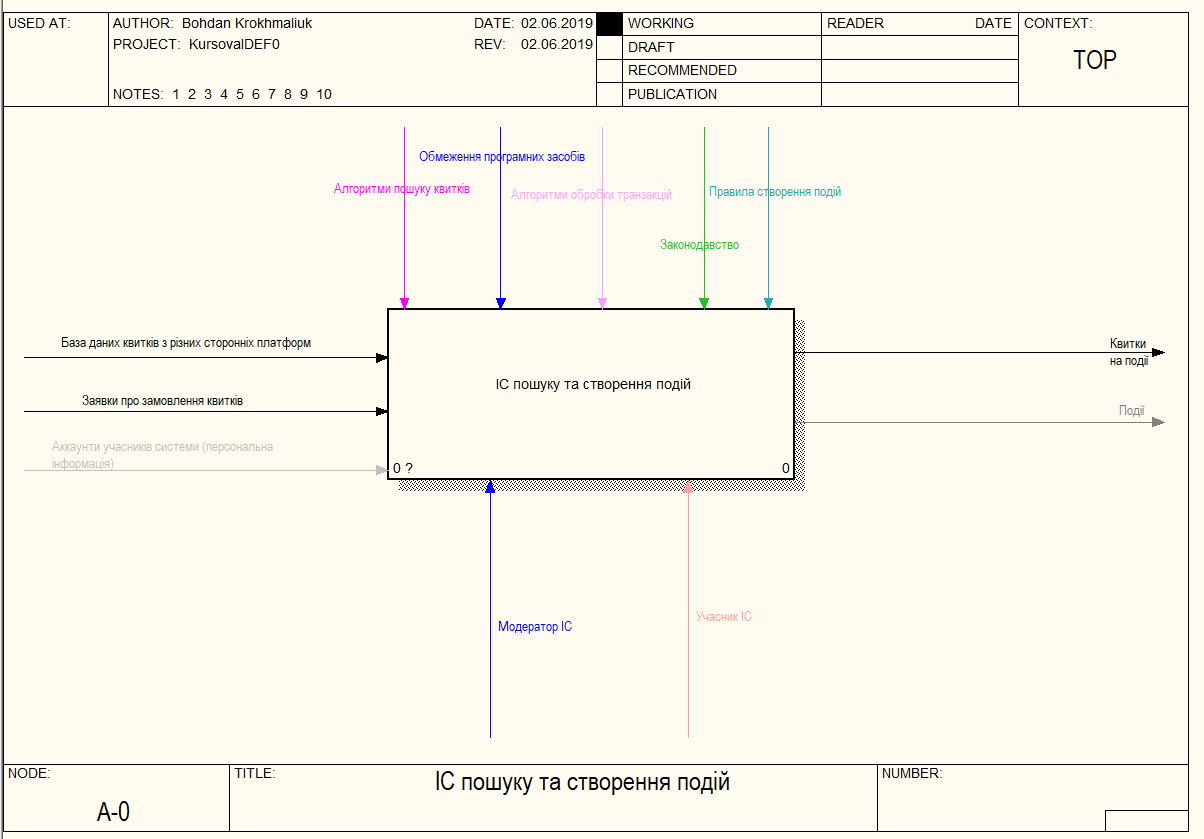


Рис. 2 IDEF0 Контекстна діаграма

Декомпозиція контекстної діаграми дає можливість розбити її на менші процеси, декомпозиція кожного наступного рівня розбиває кожен етап на підетапи які мають свої відповідні стрілки керування. На рисунках 3-6 відображені діаграми декомпозицій кожного з етапів, до 3 рівня декомпозиції. На рисунку 7 (дерево вузлів) – можна наочно побачити на які вузли розбита система.

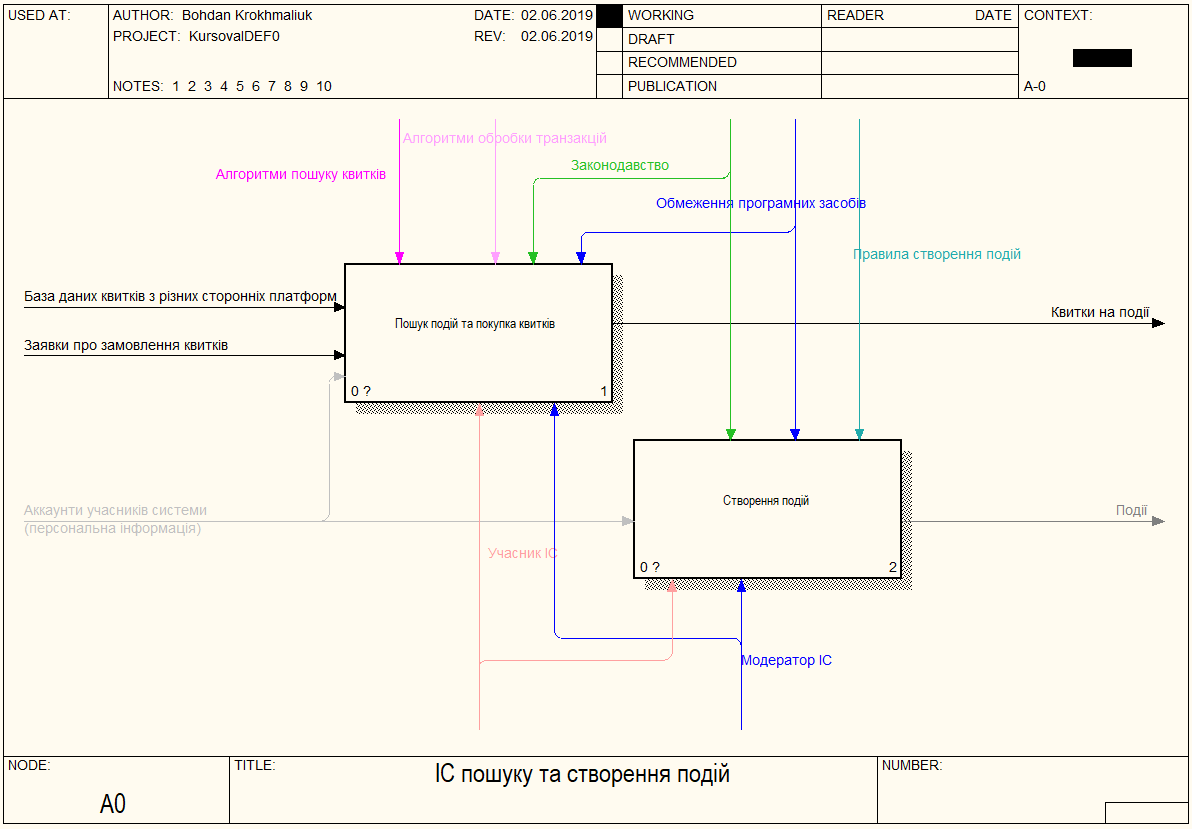
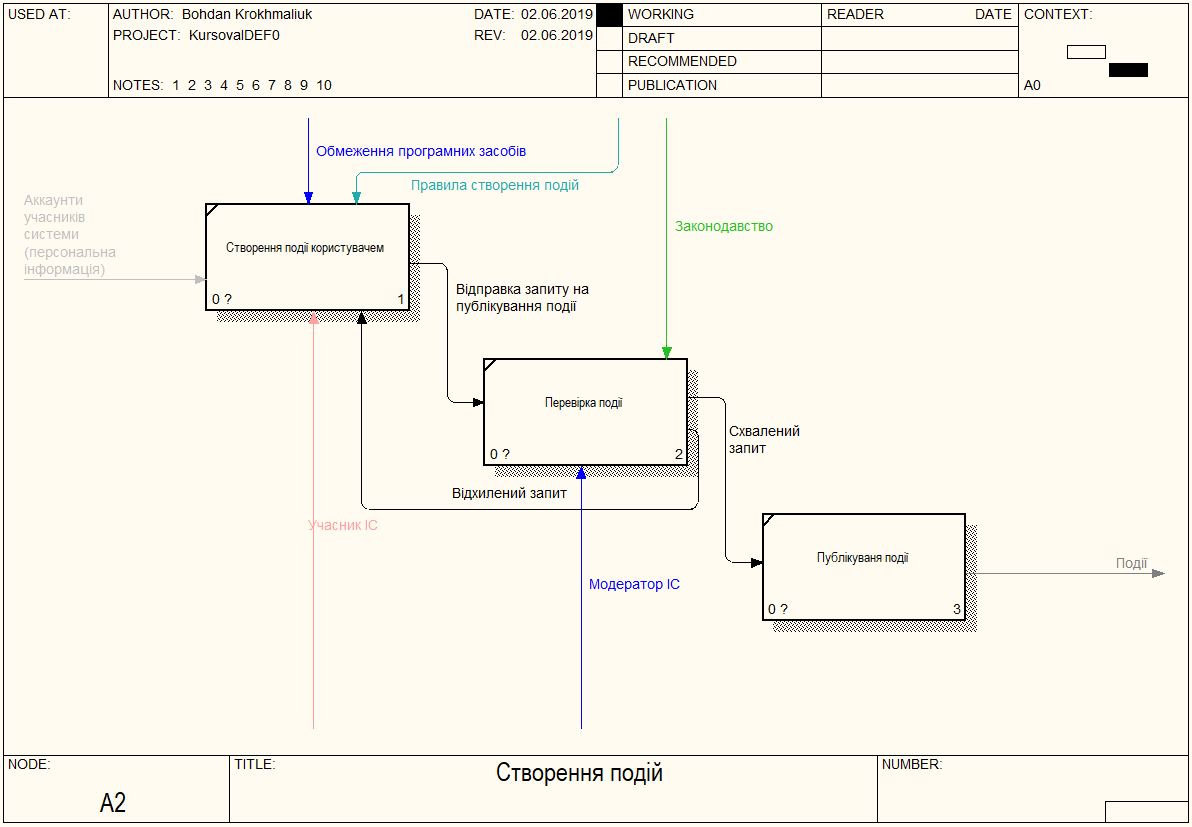
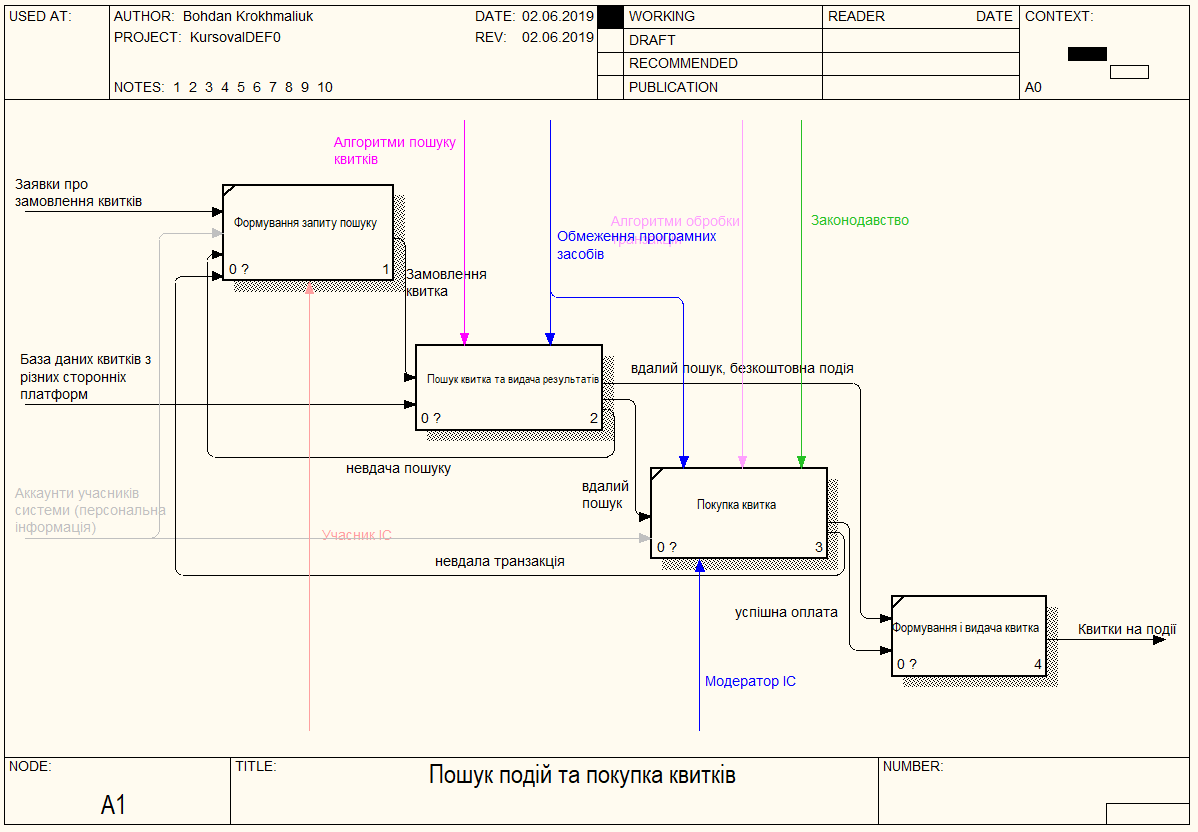
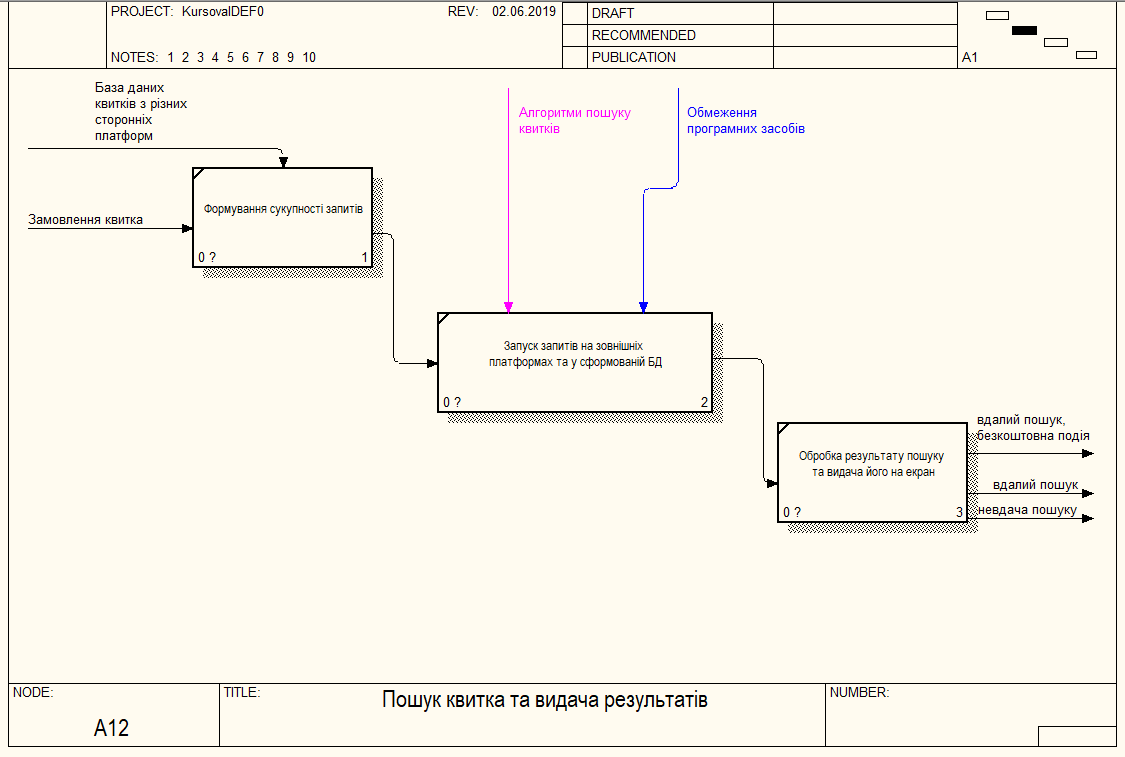
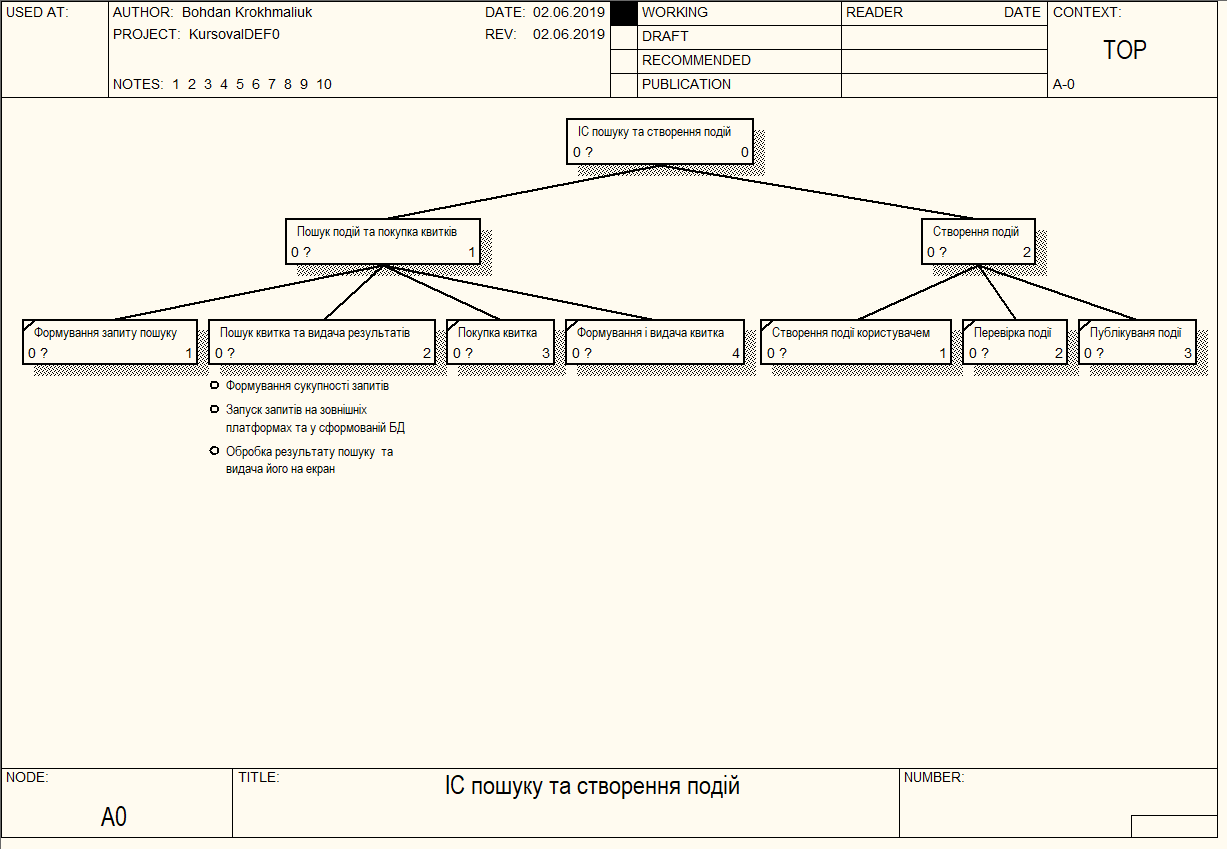


Рис. 3 IDEF0 діаграма декомпозиції 1-ого рівня

 Рис. 4 IDEF0 діаграма декомпозиції 2-ого рівня “Створення подій”

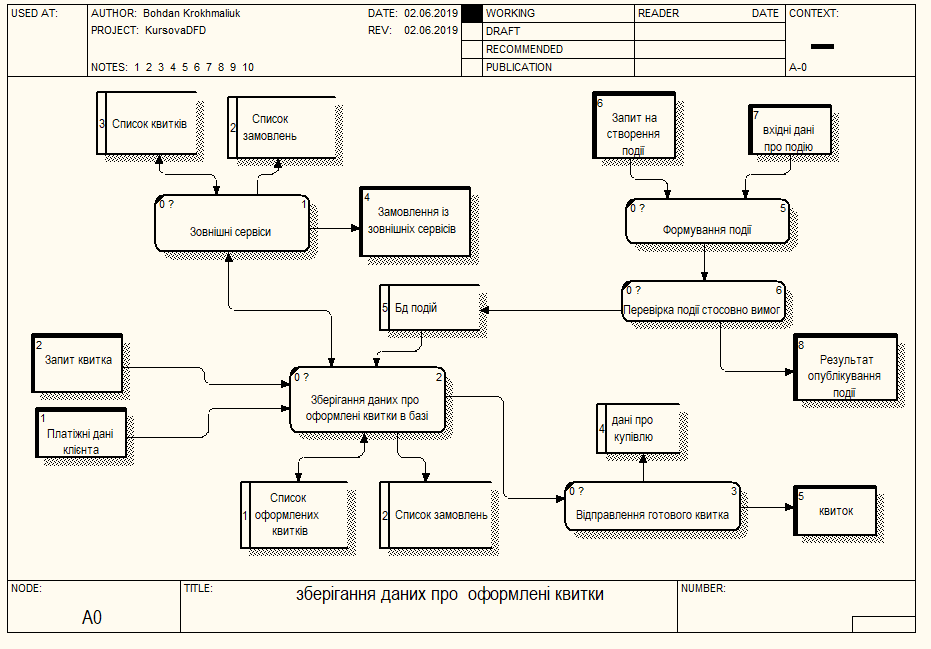
Рис. 5 IDEF0 діаграма декомпозиції 2-ого рівня “Пошук подій та покупка квитків”

Рис. 6 IDEF0 діаграма декомпозиції 3-ого рівня “Пошук подій та покупка квитків”

Рис. 7 IDEF0 діаграма дерева вузлів

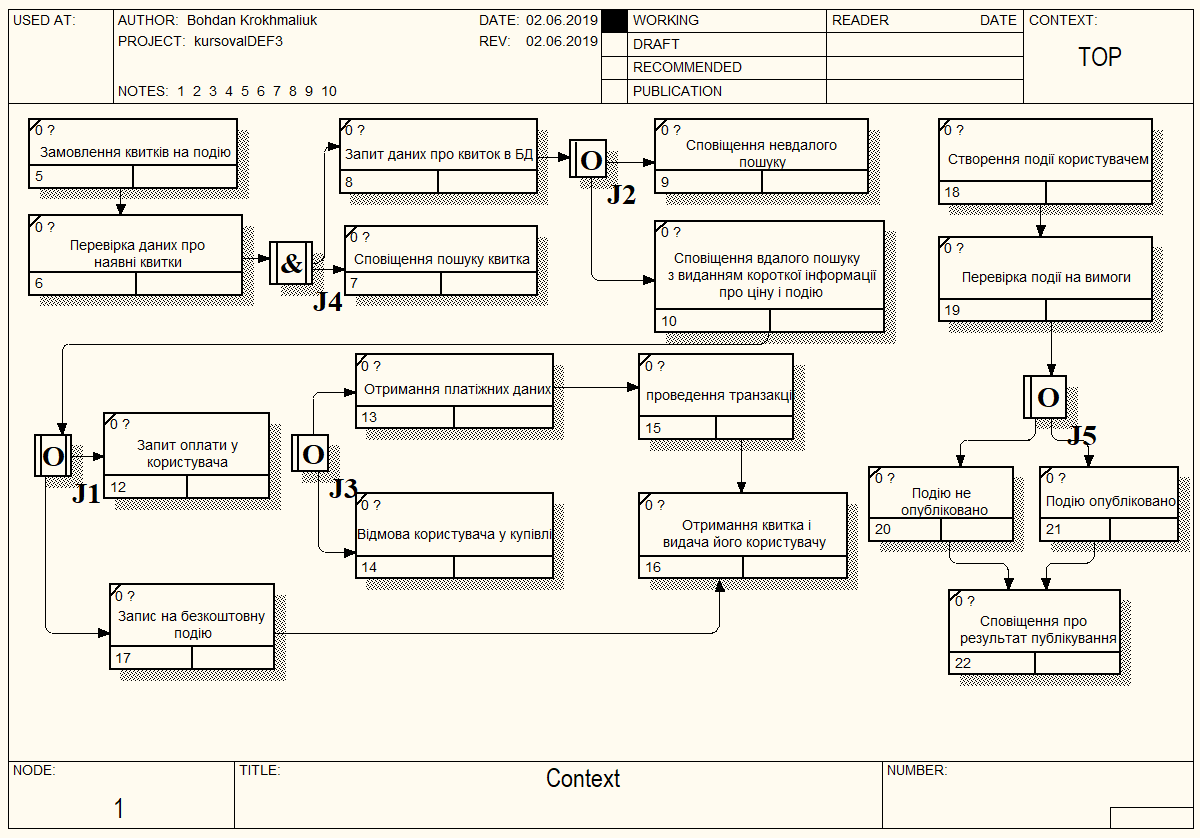
## 4.3 Діаграма DFD

Діаграми потоків даіних (Data Flow Diagramming) є основним засобом моделювання функціональних вимог до проектованої системи. Вимоги представляються у вигляді ієрархії процесів, зв'язаних потоками даних. Діаграми потоків даних показують, як кожен процес перетворює свої вхідні дані у вихідні, і виявляють відношення між цими процесами. DFD-діаграми успішно використовуються як доповнення до моделі IDEF0 для опису документообігу й обробки інформації. Подібно IDEF0, DFD представляє систему, що моделюється, як мережу зв'язаних робіт. Основні компоненти DFD (як було сказано вище) – процеси або роботи, зовнішні сутності, потоки даних, сховища даних.

Рис. 8 DFD діаграма

## 4.4 Діаграма IDEF3

IDEF3 — це метод, що має основною метою дати можливість аналітикам описати ситуацію, коли процеси виконуються в певній послідовності, а також описати об'єкти, що беруть участь спільно в одному процесі. Техніка опису набору даних IDEF3 є частиною структурного аналізу. На відміну від деяких методик описів процесів IDEF3 не обмежує аналітика надмірно твердими рамками синтаксису, що може привести до створення неповних або суперечливих моделей. IDEF3 може бути також використаний як метод створення процесів. IDEF3 доповнює IDEF0 і містить усе необхідне для побудови моделей, які надалі можуть бути використані для імітаційного аналізу. Кожна робота в IDEF3 описує деякий сценарій бізнес-процесу й може бути складовою іншої роботи. Оскільки сценарій описує мету й рамки моделі, важливо, щоб роботи йменувалися дієслівним іменником, що позначають процес дії, або фразою, що містить такий іменник.

Рис. 9 IDEF3 діаграма

# 5. КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ

## 5.1 Діаграма IDEF1x

IDEF1X є методом для розробки реляційних баз даних і використовує умовний синтаксис, спеціально розроблений для зручного побудови концептуальної схеми. Концептуальною схемою ми називаємо універсальне уявлення структури даних в рамках комерційного підприємства, незалежне від кінцевої реалізації бази даних і апаратної платформи. Будучи статичним методом розробки, IDEF1X спочатку не призначений для динамічного аналізу за принципом "AS IS", тим не менш, він іноді застосовується в цій якості, як альтернатива методу IDEF1. Використання методу IDEF1X найбільш доцільно для побудови логічної структури бази даних після того, як всі інформаційні ресурси досліджені (скажімо з допомогою методу IDEF1) і рішення про впровадження реляційної бази даних, як частини корпоративної інформаційної системи, було прийнято.

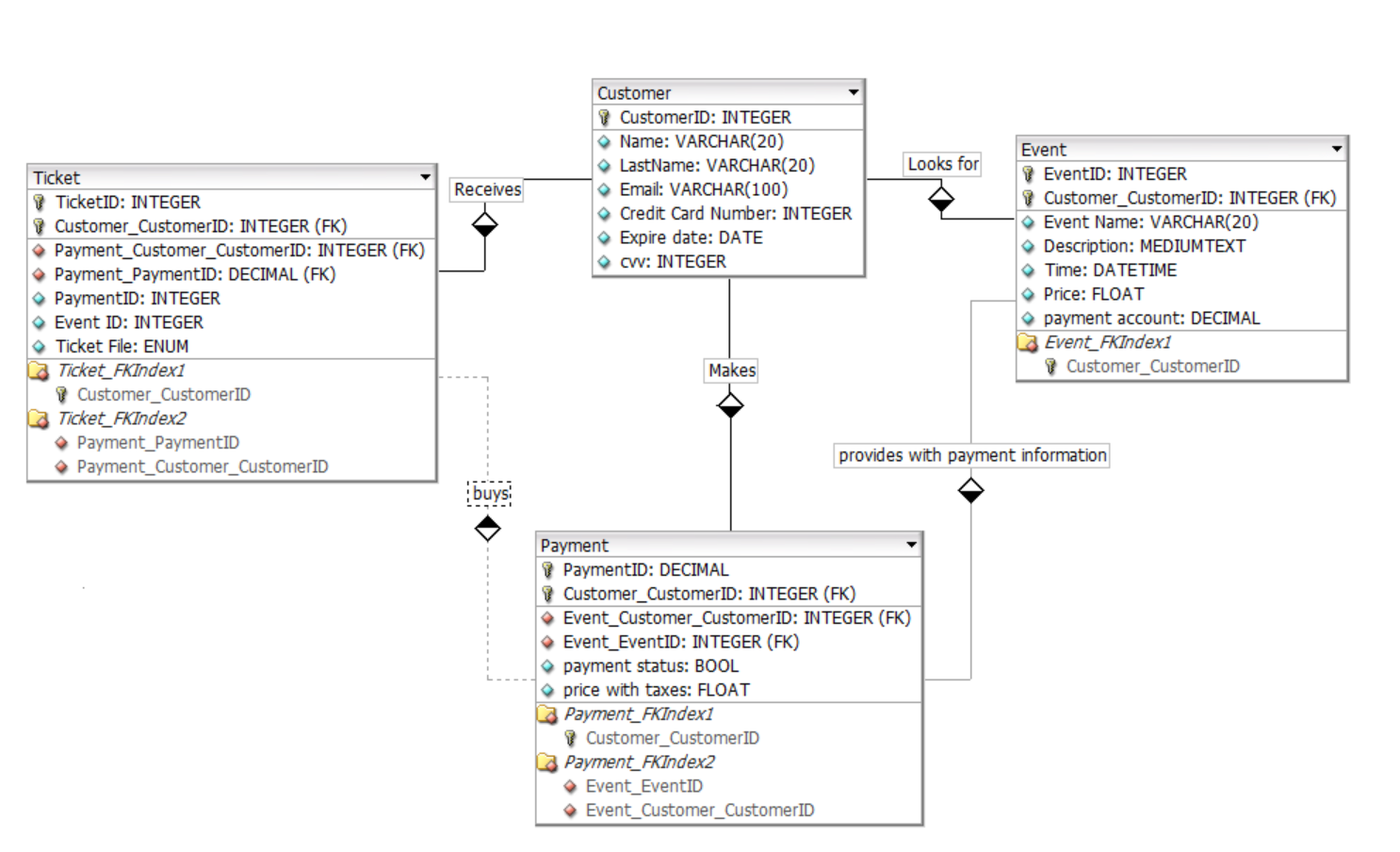


Рис. 10 IDEF1x діаграма

## 5.2 UML Діаграми

UML (англ. Unified Modeling Language) — уніфікована мова моделювання. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

### 5.2.1 Діаграма прецедентів

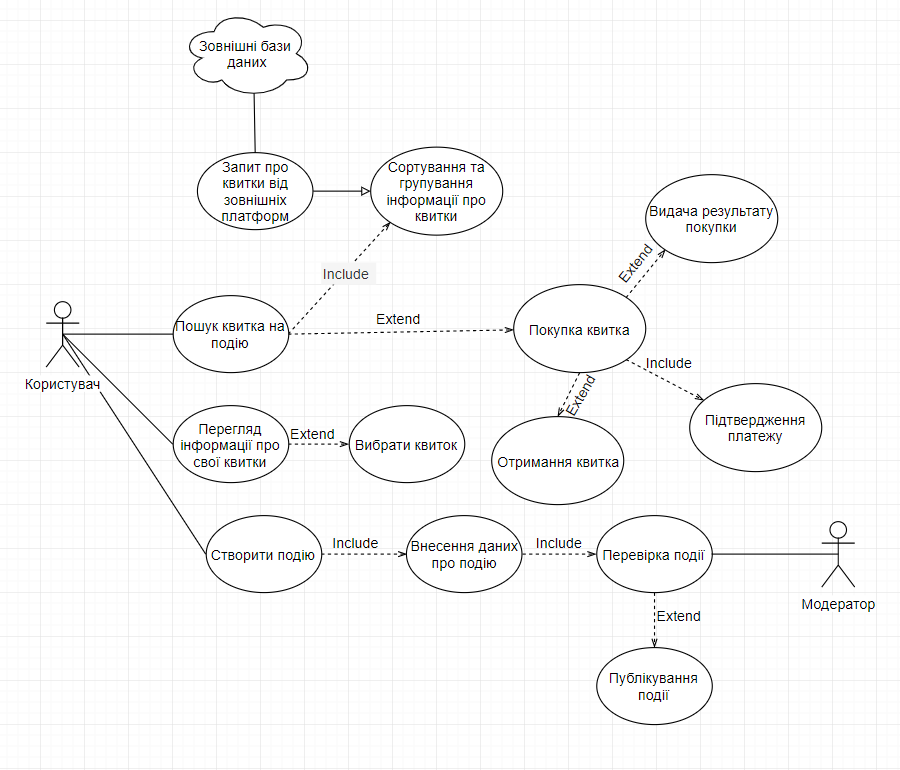


Рис. 11 Діаграма прецедетнів

Опишемо головні прецеденти, подані на діаграмі.

|  |  |
| --- | --- |
| Прецедент | Пошук квитка на подію |
| Короткий опис | Користувач переглядає упорядковані результати пошуку квитка |
| Суб’єкти | Користувач, Система. |
| Передумови | Існує інформація про заданий квиток |
| Основний потік | Прецедент починається, коли Користувач зробив запит в систему. Система надає Користувачу список результатів. |
| Альтернативний  потік | Користувач повертається на основну сторінку пошуку квитка. |
| Постумови | Клієнт отримує інформацію. |

Табл. 1 Опис прецедента “Пошук квитка на подію”

|  |  |
| --- | --- |
| Прецедент | Сортування і групування інформації про квитки |
| Короткий опис | Відсортувати інформацію про квитки отриману від зовнішніх платформ |
| Суб’єкти | Система |
| Передумови | Зовнішні платформи повернули результати запиту |
| Основний потік | Прецедент починається, коли зовнішні платформи повертають результат запиту. Якщо дані введені коректно, то здійснюється сортування та групування інформації про квитки. |
| Альтернативний  потік | Зовнішні платформи не змогли виконати запит(некоректні дані). |
| Постумови | Користувач отримує логічно-структуровану інформацію. |

Табл. 2 Опис прецедента “Сортування і групування інформації про квитки”

|  |  |
| --- | --- |
| Прецедент | Видача результату від зовнішніх платформ |
| Короткий опис | Зовнішні платформи видають результат запиту |
| Суб’єкти | Зовнішні платформи , Система. |
| Передумови | Користувач ввів коректні дані |
| Основний потік | Прецедент починається, коли Зовнішні платформи знайшли результати в своїх базах даних. Якщо дані введені коректно, то Система дає базу даних квитків на запитану подію. |
| Альтернативний  потік | Не знайдено результатів |
| Постумови | Користувач отримує дані про квитки. |

Табл. 3 Опис прецедента “Видача результату від зовнішніх платформ”

|  |  |
| --- | --- |
| Прецедент | Покупка квитка |
| Короткий опис | Користувач купує вибраний квиток. |
| Суб’єкти | Користувач, Система. |
| Передумови | Наявний квиток. |
| Основний потік | Прецедент починається, коли Користувач здійснив пошук квитка на подію. Система запитує Користувача про платіж. |
| Альтернативний  потік | Користувач повертається на основну сторінку вибору функцій. |
| Постумови | Користувач отримує результат покупки. |

Табл. 4 Опис прецедента “Покупка квитка”

|  |  |
| --- | --- |
| Прецедент | Підтвердження платежу |
| Короткий опис | Користувач підтверджує платіж. |
| Суб’єкти | Користувач, Система. |
| Передумови | Користувач вибрав квиток. У Користувача є достатня кількість грошей на рахунку. |
| Основний потік | Прецедент починається, коли Користувач вибрав квиток, якщо все введено правильно, то у базі даних добавляється запис про запит транзакції. Користувач виконує запит. |
| Альтернативний  потік | Користувач не підтверджує платіж протягом 5хв, або вирішує повернутись до пошуку квитка . |
| Постумови | Користувач отримує результат транзакції. |

Табл. 5 Опис прецедента “Підтвердження платежу”

|  |  |
| --- | --- |
| Прецедент | Видача результату покупки. |
| Короткий опис | Система повертає Користувачу повідолення про вдалу (квиток) або невдалу покупку. |
| Суб’єкти | Користувач, Система. |
| Передумови | Користувач намагався здійснити покупку квитка |
| Основний потік | Прецедент починається, коли Користувач вибрав квиток для покупки і відомий результат транзакції. Система надсилає інформацію Користувачу, про вдалу транзакцію або невдале завершення покупки. |
| Альтернативний  потік | Немає. |
| Постумови | Користувач отримує квиток в базу даних квитків Користувача. |

Табл. 6 Опис прецедента “Видача результату покупки”

|  |  |
| --- | --- |
| Прецедент | Переглянути інформацію про куплені квитки |
| Короткий опис | Користувач переглядає інформацію про придбані квитки. |
| Суб’єкти | Користувач, Система. |
| Передумови | Нема. |
| Основний потік | Прецедент починається, коли Користувач вибрав пункт “Переглянути інформацію про куплені квитки”. Система надсилає інформацію Користувачу, про наявні квитки. |
| Альтернативний  потік | Немає. |
| Постумови | Користувач переглядає свої квитки. |

Табл. 7 Опис прецедента “Переглянути інформацію про куплені квитки”

|  |  |
| --- | --- |
| Прецедент | Вибрати квиток. |
| Короткий опис | Користувач вибирає квиток для завантаження/відкриття в своєму девайсі. |
| Суб’єкти | Користувач, Система. |
| Передумови | Куплений квиток |
| Основний потік | Прецедент починається, коли Користувач придбав хоча б 1 квиток. Система надсилає інформацію Користувачу у вигляді завантаження файлу квитка. |
| Альтернативний  потік | Користувач повертається в попереднє меню. |
| Постумови | Користувач отримує завантажений файл квитка на свій девайс. |

Табл. 8 Опис прецедента “Вибрати квиток”

### 5.2.2 Діаграма послідовностей

Діаграма послідовності (sequence diagram) - діаграма, на якій показані взаємодії об'єктів, впорядковані за часом їх прояву.

Анатомія діаграм послідовності :

- Об'єкти зображуються у вигляді прямокутників і розміщуються над лініями життя (lifeline).

- Лінії життя зображуються вертикальними лініями.

- Напрям часу – згори вниз.

- Повідомлення – це горизонтальні стрілки з назвою повідомлення.

- За напрямом стрілки визначають відправника та отримувача.

- Активізація об'єктів (focus of control).

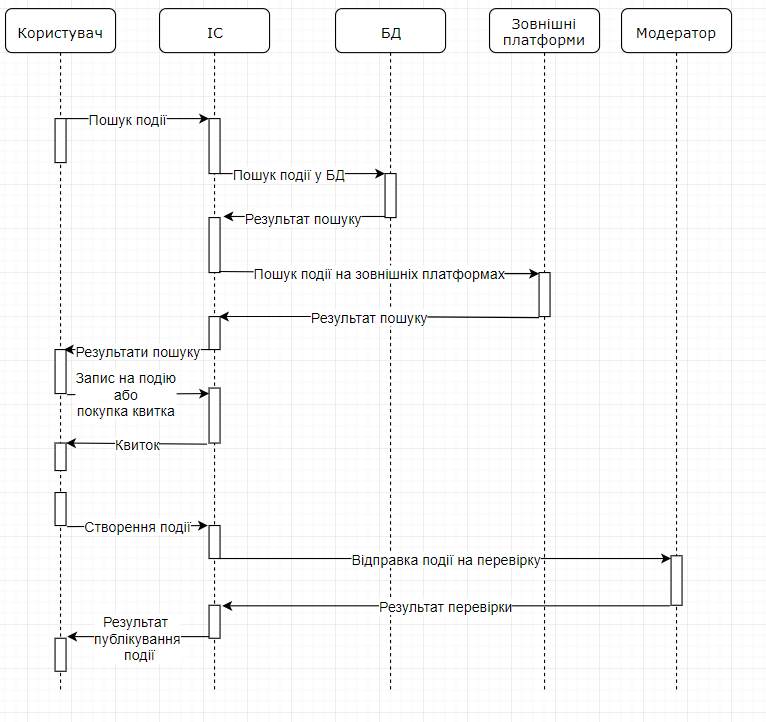
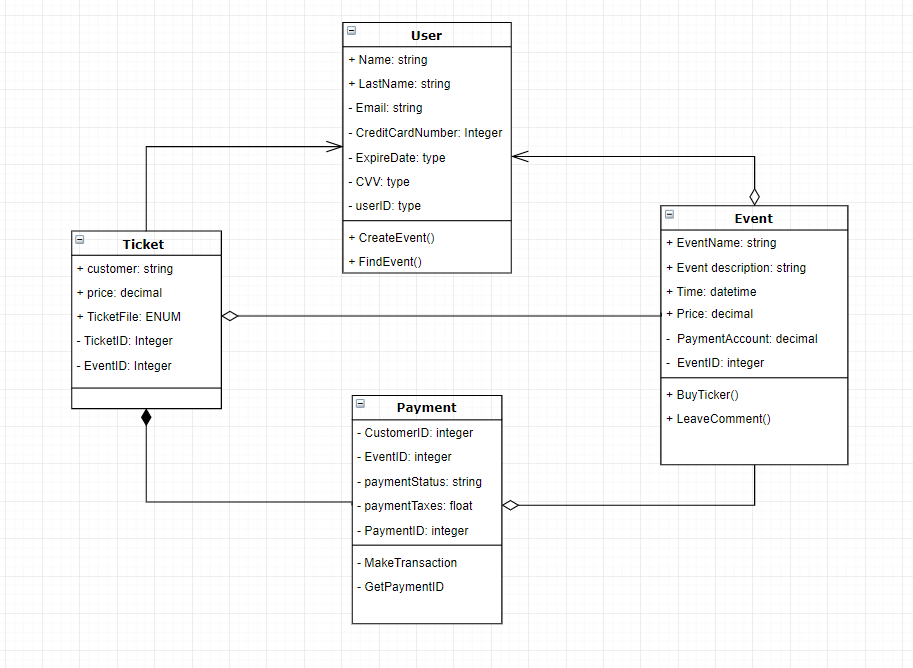


Рис. 12 Діаграма послідовностей

### 5.2.3 Діаграма класів

На діаграмах класів показані різноманітні класи, які утворюють систему, а також їх взаємозв’язки. Діаграми класів називають “статичними діаграмами”, оскільки на них показано класи разом з атрибутами і операціями (методами), а також статичний взаємозв’язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи «є частиною» інших класів, — але не показано методи, які при цьому викликаються.

Діаграми класів можуть використовуватись для генерації каркасного програмного коду (в реальній мові програмування).

Рис. 13 Діаграма класів

## 5.3 Вартісний аналіз

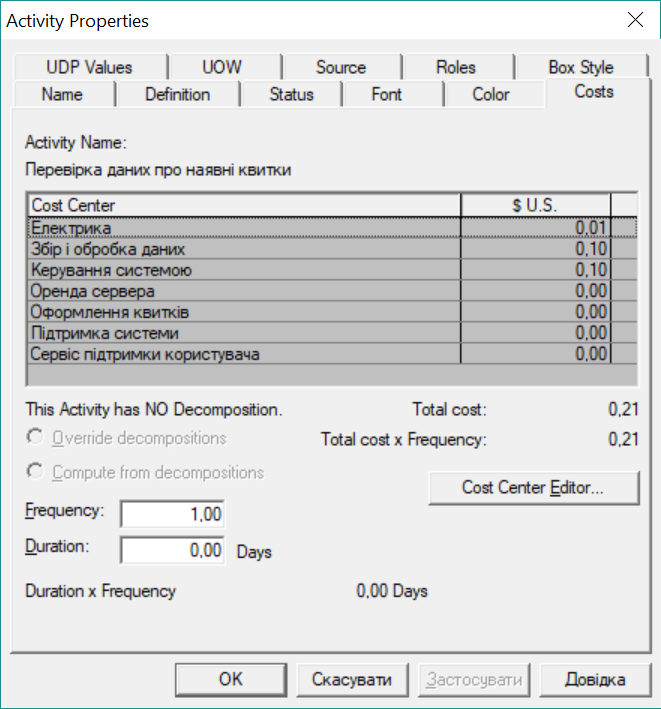
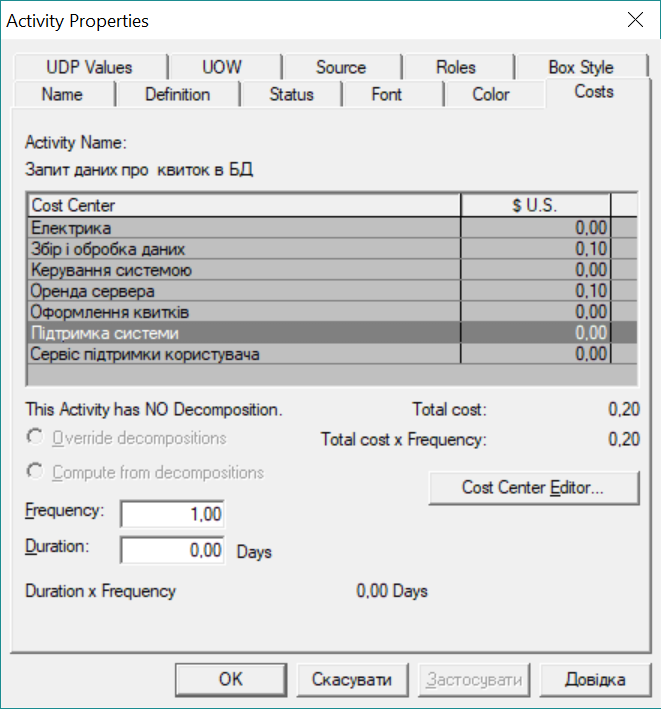
 

Рис. 14 Перевірка даних про квитки Рис.15 Запит даних до БД

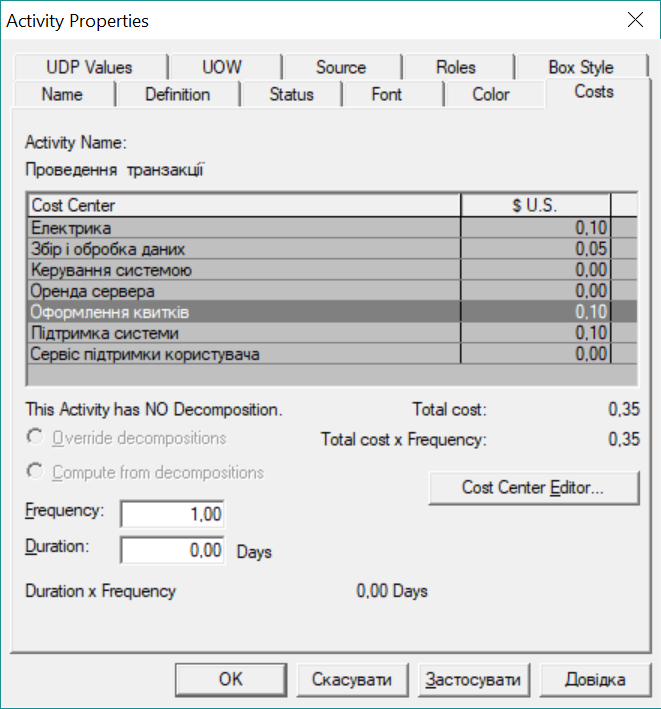
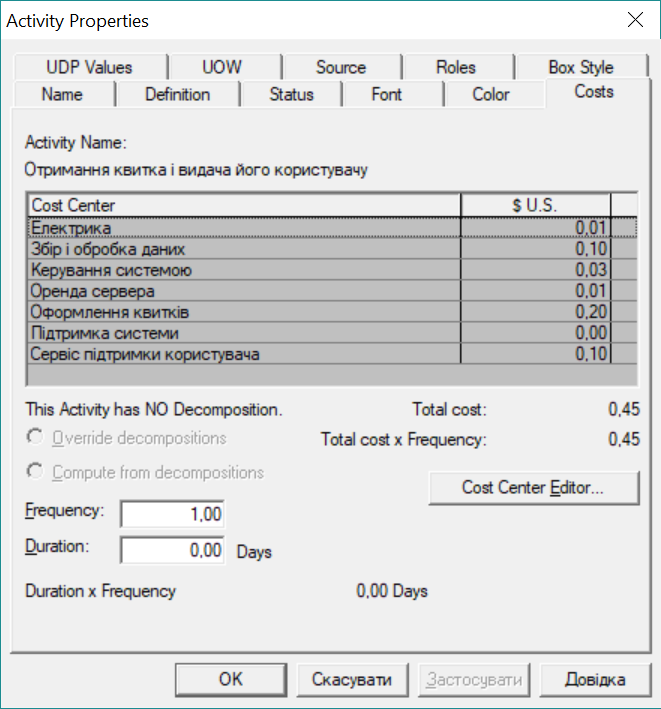
 

Рис.16 Проведення транзакції Рис.17 Отримання квитка

# 6. ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКА БАЗИ ДАНИХ

За допомогою середовища Microsoft Office Access було розроблено базу даних для даної ІС. Відповідно до діаграми класів були створені таблиці з відповідними полями та ідентифікаторами, а також побудовано діаграму бази даних зі зв’язками між таблицями. В додатку можуть виконуватися запити на вставку, модифікацію та видалення даних. На кожному рисунку з таблицями БД описується кожна таблиця яку ми будемо використовувати у нашій БД, а також поля які у ній будуть присутні , тип кожного поля (число, дата час, текст), і властивість поля, тобто чи поле має певну маску , його розмір, чи зв’язок з іншою таблицею(відношенням) .

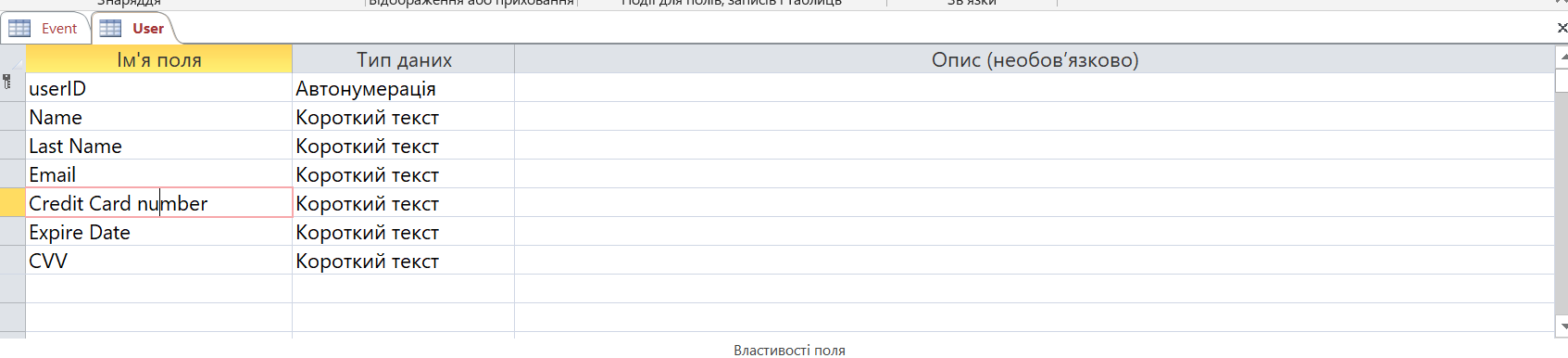
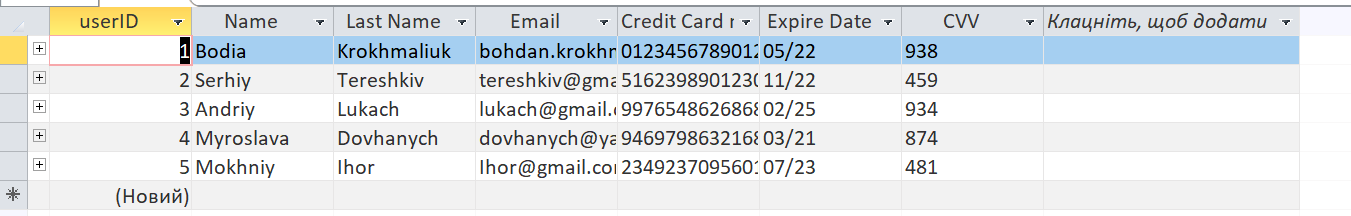
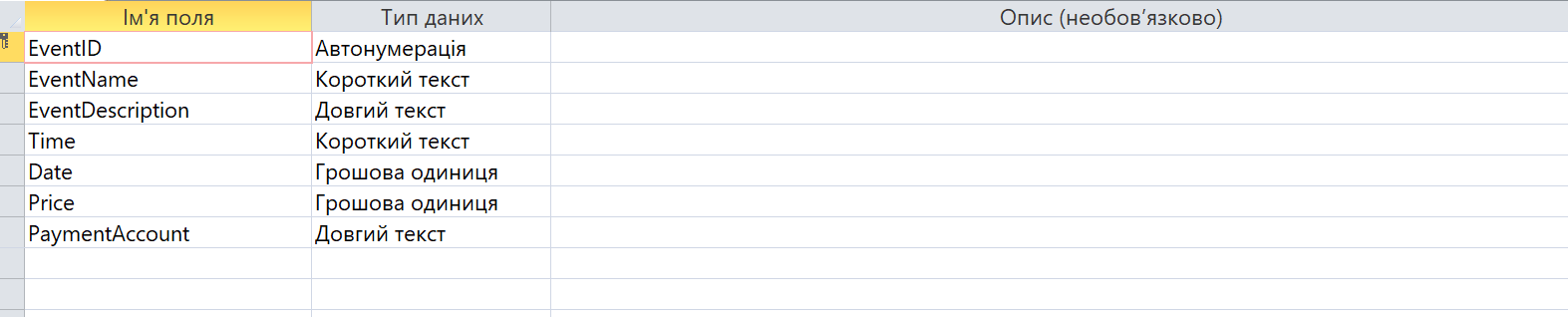
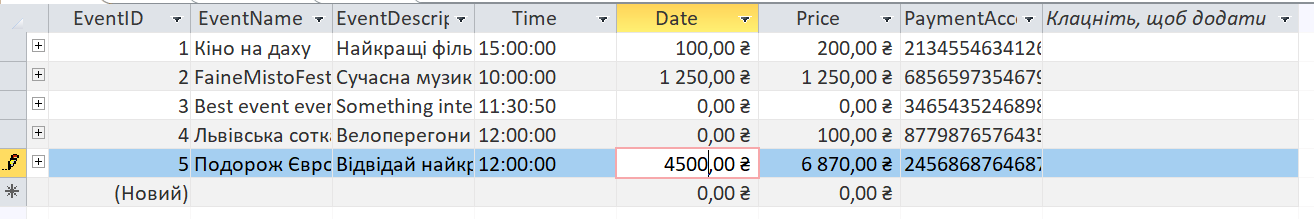
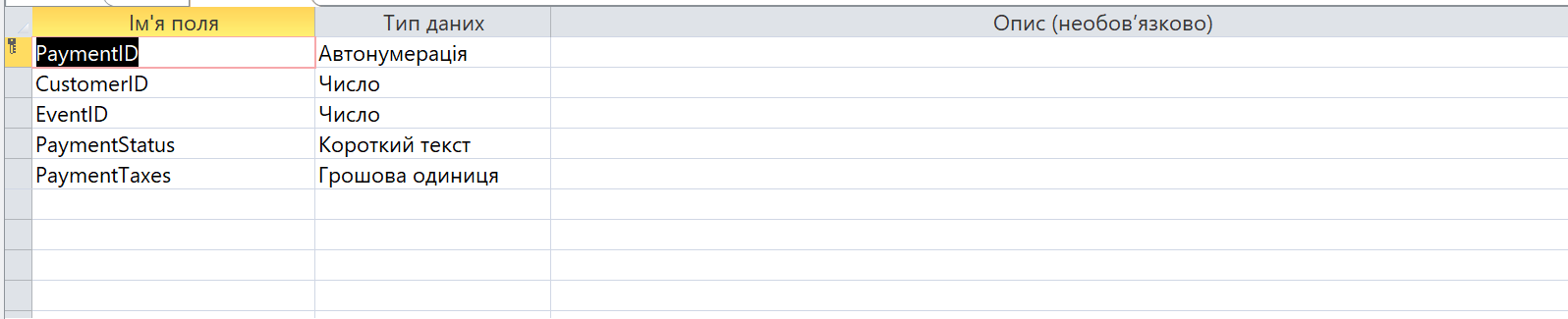
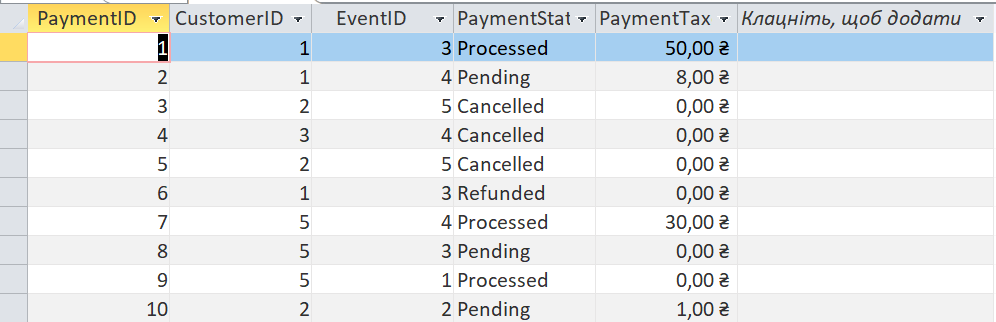
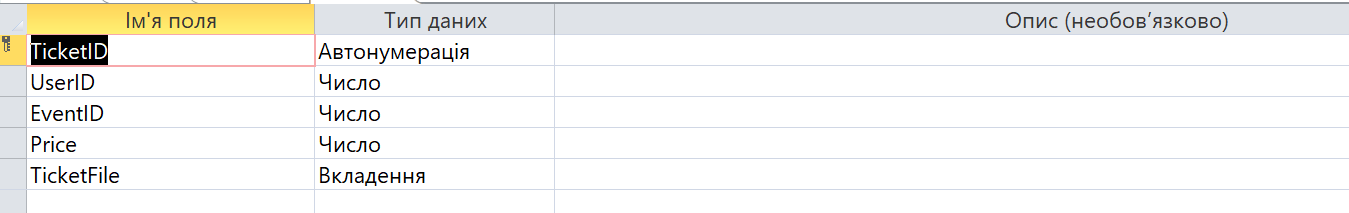
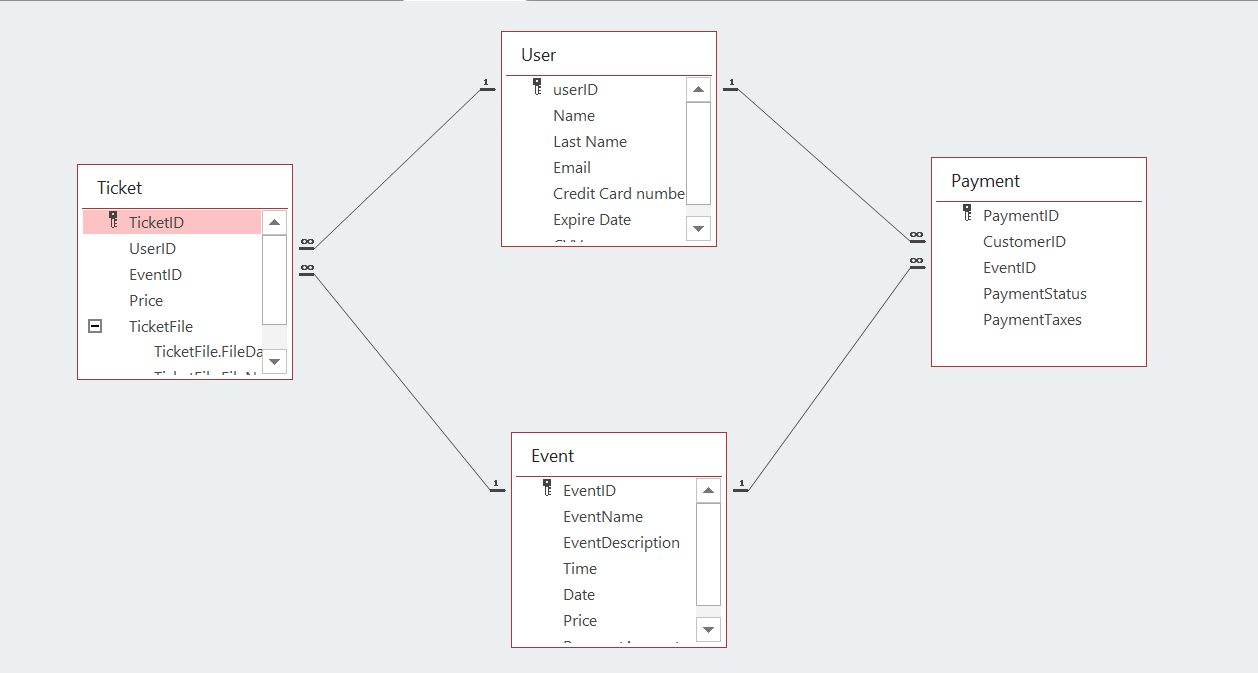
 

Рис. 18 Сутність та таблиця “User”

  Рис. 19 Сутність та таблиця “Event”

  Рис. 20 Сутність та таблиця “Payment”

Рис. 21 Сутність та таблиця “Ticket”

Рис. 22 Звязки між сутностями

# ВИСНОВОК

Виконавши цю курсову роботу, я навчився створювати логічні та фізичні моделі системи в пакеті ERwin, генерувати звіти про її структуру, а також властивості і взаємозв’язки між її частинам, створювати базу даних; на основі створеної в пакеті ERwin моделі генерувати готову для подальшої роботи базу даних, а також виконувати обернену дію – визначити структуру (модель) бази даних представлень існуючого файлу бази даних. Навчився оцінювати розміри фізичної бази даних на основі її моделі. Виконав індивідуальне завдання для інформаційної системи “ Соціальна мережа подій”

# СПИСОК ЛІТЕРАТУРИ

1. Гвоздева В. А. Основы построения автоматизированных информационных систем : учебник / В. А. Гвоздева, Ю. И. Лаврентьева. – М. : ИД "Форум": ИНФРА-М, 2007. – 320 с.
2. Шаховська Н. Б., Литвин В. В. Проектування інформаційних систем: навчальний посібник / Н. Б. Шаховська, В. В. Литвин. -Львів: 'Магнолія- 2006", 2011. - 380 с.
3. Вендров А. М. Проектирование программного обеспечения эко- номических информационных систем : учебник / А. М. Вендров. – 2-е изд., перераб. и доп. – М. : Финансы и статистика, 2006. – 544 с.
4. Грекул В. И. Проектирование информационных систем : учебн. пособ. / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – М. : БИНОМ. Лаборатория знаний, 2008. – 300 с.
5. Избачков Ю. С. Информационные системы : учебник / Ю. С. Избачков, В. Н. Петров. – 2-е изд. – СПб. : Питер, 2006. – 656 с.
6. Ларман, К. Применение UML и шаблонов проектирования: Уч. Пос / К. Ларман. - М.: Издательский дом «Вильямс», 2001. - 496 с.
7. Гранд, М. Шаблоны проектирования в Java / М. Гранд. - М.: Новое знание, 2004. - 559 с.
8. Крачтен, Ф. Введение в Rational Unified Process / Ф. Кратчен. - М.: Издательский дом «Вильямс», 2002. - 240 с.
9. Йордан, Э. Объектно-ориентированный анализ и проектирование систем / Э. Йордан, С. Аргила. - М.: Издательство «ЛОРИ», 2007. - 264 с.