ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка
До бакалаврської роботи
На ступінь вищої освіти бакалавр

На тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ КОМУНАЛЬНИХ ВИТРАТ КОРИСТУВАЧА МОВОЮ C#»**

Виконав: студент 4 курсу, групи ПД-43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Висоцький В.В
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент ……..
(прізвище та ініціали)

Київ-2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти- «Бакалавр
Спеціальність підготовки- 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Негоденко О.В.
« » _____2023року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ВИСОЦЬКОГО ВЛАДИСЛАВА ВІКТОРОВИЧА
(прізвище, ім'я, по батьковій)

1. Тема роботи : «Розробка програмного забезпечення для аналізу комунальних витрат користувача мовою C#»
Керівник роботи Гаманюк І.М. старший викладач
(прізвище, ім'я по батьковій, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26

2. Строк подання студентом роботи «1» червня 2023 року

3. Вхідні данні до роботи

   3.1. Науково-технічна література, пов'язана із питаннями аналізу витрат ресурсів

   3.2. Методи моделювання інформаційних систем

   3.3.Онлайн редактор UML-діаграм

   3.4.Інтегроване середовище розробки Visual Studio

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

   4.1. Контроль за ресурсами- запорука успішної економіки

   4.2. Аналіз наявних засобів та технологій для аналізу комунальних витрат

   4.3. Моделювання та проектування інформаційної системи

   4.4. Тестування системи

5. Перелік демонстраційного матеріалу

5.1.Розробка застосунку

5.2. Титульний слайд

5.3. Мета, об’єкт та предмет дослідження

5.4. Актуальність проблеми

5.5. Аналоги

5.6. Технічні завдання

5.7. Інструменти, використані для реалізації

5.8. Висновки

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.03.23 - 14.03.23	Виконано
2	Дослідження існуючих інструментів для аналізу комунальних витрат	15.03.23 - 17.03.23	Виконано
3	Проектування архітектури системи	01.04.23 - 10.04.23	Виконано
4	Розробка системи для аналізу комунальних витрат	18.04.23 - 27.04.23	Виконано
5	Висновки та оформлення роботи	01.05.23 – 03.05.23	Виконано
6	Розробка обов’язкових демонстраційних матеріалів	05.05.23 – 07.05.23	Виконано
7	Попередній захист роботи	24.05.2023	Виконано
8	Здача роботи	1.06.23	Виконано

Студент _____

(підпис) (прізвище та ініціали)

Керівник роботи _____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота складається з: вступу, чотирьох розділів, висновків, списку використаних джерел зі 8 найменувань. У тексті дипломної роботи міститься 27 рисунків. Загальний обсяг роботи 72 листа.

ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРБКИ (VS), БАЗА ДАННИХ SQL, UML-ДІАГРАМА.

Об’єкт дослідження – процес аналізу комунальних витрат.

Предмет дослідження- програмне забезпечення для аналізу комунальних витрат.

Мета роботи – спрощення процесу аналізу комунальних витрат за рахунок застосування комп’ютерного програмного забезпечення створеного на мові C#.

На основі результатів виконаних досліджень розроблена система аналізу комунальних витрат «Tosu» яка дозволяє планувати витрати.

Впровадження створеної інформаційної системи аналізу комунальних витрат дозволить отримати достовірну та своєчасну інформацію стосовно комунальних витрат користувача.

ЗМІСТ

ВСТУП

Аналіз комунальних витрат полягає у плануванні і контролю за витратами ресурсів як фінансових так і не фінансових. Процес роботи Tosu виглядає наступним чином. Користувач реєструється. Потім обирає тип витрат і або витрати за період або витрати за подію. І в тому і втому випадку потрібно ввести показники і дати. Якщо користувач введе ще данні про кількість витрат на певний період то програма розрахує йому середні витрати на період або захід. Якщо користувач буде витрачати більше ніж пропонується, то програма буде зменшувати кількість доступних витрат і так до кінця поки не залишиться 0. Програма ніяк не перешкоджає цим витратам, якщо кількість дозволених витрат буде дорівнювати 0, то програма просто сповістить про це.

Програма про аналіз витрат а не про примусовий контроль або фінансовий контроль.

Об’єкт дослідження – процес аналізу комунальних витрат.

Предмет дослідження- програмне забезпечення для аналізу комунальних витрат.

Мета роботи – спрощення процесу аналізу комунальних витрат за рахунок застосування комп’ютерного програмного забезпечення створеного на мові C#.

1. АНАЛІЗ КОМУНАЛЬНИХ ВИТРАТ

1.1. Аналіз комунальних витрат у сьогоденні

Метою аналізу комунальних витрат сьогодні є оптимізація витрат на комунальні платежі коштів користувачів. Кожен хто займається виплатами комунальних платежів – їх аналізує в будь якому випадку. Якщо ставитись до цих витрат зверхньо, то одного разу можна втрапити у борги.

У всіх нас існують певні рамки для витрат на ті чи інші примхи або так би мовити сфери витрат. Наприклад витрати на їжу, одяг, побут тощо. Або на походи до вистав, концертів тощо. А фінанси не безкінечні у кожної людини. Тому планування витрат є одним із рішень проблеми.

Контроль за витрачанням ресурсів актуальним буде в будь який час. Аналіз витрачення ресурсів- запорука успішної економії. Для вирішення цієї задачі можна створити програму на мові програмування c# з додаванням windows forms у середовищі програмування visual studio з підключенням SQL server.

Вирішення цієї проблеми може бути процес наступним чином. Користувач заходить у свій аккаунт і йому стає доступний перелік ресурсів таких як:

-Газ

-Вода

-Електрика

Лічильник або користувач передає у базу даних показники. Показники заносяться у базу даних. І користувач може подивитися або спланувати свої витрати за певний період.

Якщо користувач хоче на перед розрахувати свої витрати то він вводить у програму певну кількість одиниць вимірювання. Наприклад 30 кубів води на місяць. А також користувач може вказати певні події на які витрати будуть більшими ніж зазвичай. Наприклад на свято потрібно помити посуд, тому потрібно буде 5 кубів води.

Монетизація програми може відбуватися на мою думку двома способами:

-Через її продаж.

-Через додавання реклами у ПЗ.

Цільова аудиторія є звичайні люди. Або інакше платники податків.

Якщо робити таку програму для підприємства то її робота може бути виконана наступним чином. Усім платники податків встановлюють розумні лічильники які самі будуть відправляти на сервер показання раз у місяць. У сервері є база даних у якої є три показника вода, газ, електрика. Від кожної адреси. В базу даних заносяться показники.

Існує проблема, коли у певної квартирі є невеликий борг, а комунальники додають на боржника додатковий борг до якого він не має жодного відношення. Розрахунки платежів повинні бути прозорими тому кожен користувач може продивитися базу даних будь якої адреси і побачити скільки він заплатив або який борг у цієї адреси.

Платники податків будуть платити додатковий податок за автоматичну плату податків.

2. АНАЛІЗ НАЯВНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ДЛЯ АНАЛІЗУ КОМУНАЛЬНИХ ВИТРАТ

Засоби для аналізу комунальних витрат – Планування подумки. Паперові засоби. Excel таблиця.

Планування подумки – найпоширеніший підхід і найменш ефективний. Здібності мозку аналізувати і зберігати інформацію обмежені, а з огляду на велику кількість повсякденних справ, можна легко втратити важливе.

Паперові засоби включають різноманітні щоденники в тому числі які спеціально структуровані для ведення аналізу комунальних витрат. Одним із прикладів є блокнот на комп'ютері Akelpad. У якому зручно робити записи та видаляти їх. Приклад аналізу комунальних витрат наведено на рисунку 2.1.

Переваги Akelpad в його простоті та доступності. Не потрібно навчатися для його користування. Не потребує залучення спеціалістів.

Недоліками є- друковані версії коштують дорого. Можна легко помилитися у розрахунках. Якщо робити аналіз за великими обсягами даних то виникають важкості з підрахунками відносно часу.

Рисунок 2.1 – Приклад аналізу комунальних витрат у блокноті Akelpad

Excel Таблиці є більш зручним способом аналізу комунальних витрат. Приклад таблиці для аналізу комунальних витрат наведено на рисунку 2.2.

Дата	Показання лічильника	Витрати за попередню добу	Пропонуємі втрати
01.04.2023	2201	2	3
01.05.2023	2203	4	3,034483
01.06.2023	2207	5	3,071429
01.07.2023	2212	3	3,148148
01.08.2023	2215	2	3,346154
01.09.2023	2217	3	3,52
01.10.2023	2220	5	3,625
01.11.2023	2225	1	3,695652
01.12.2023	2226	4	4,045455
01.13.2023	2230	2	4,095238
01.14.2023	2232	3	4,4
01.15.2023	2235	2	4,578947
01.16.2023	2237	2	4,888889
01.17.2023	2239		5,176471

Рисунок 2.2 – Приклад аналізу комунальних витрат у таблиці Excel

Переваги електронних таблиць Excel- простий інтерфейс взаємодії з додатком. Можливість автоматизувати розрахунки із використанням простих і складних формул Excel. При розрахунках статистики за великий період помилитися важко і рахує все програма.

Недоліками є – потрібен досвід для користування а для деяких випадків навість спеціаліст. З часом таблиці розростаються і робота з ними ускладняється. Дані вводить людина, а це значить що може виникнути помилка. Додаток Wallet дозволяє виконувати довгострокове стратегічне планування бюджету для аналізу комунальних витрат.

Переваги Wallet це- автоматичне зчитування банківських транзакцій та автоматичний розподіл витрат за категоріями. Засоби планування бюджету, які ґрунтуються на даних про поточні витрати та доходи. До недоліків можна віднести- додаткові функції типу повідомлення та розумного помічника доступні лише за кошти.

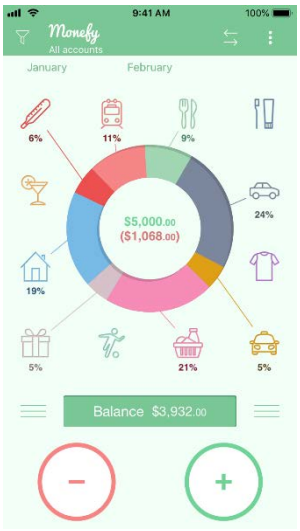


Рисунок 2.3 – Приклад аналізу комунальних витрат у додатку Wallet

Зведені результати аналізу характеристик розглянутих засобів наведено у таблиці 2.4.

Таблиця 2.4 – Зведені результати аналізу характеристик засобів для аналізу комунальних витрат.

Засіб	**Wallet**	**Akelpad**	**Excel**
Платформа	Android, iOs, Web	Windows	Windows,Android
Автоматичні розрахунки	+	-	Тільки якщо формулу впише людина
Автоматичне заповнення	+	-	-
Великий ризик помилитися у заповненні та розрахунках	-	+	+
Легкий у користуванні	+	+	-
З збільшенням обсягу даних стає важчим	-	+	+

3. МОДЕЛЮВАННЯ ТА ПРОЕТКУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Бачення

Дата внесення змін:

Версія: Кінцева версія

Дата 15.02.2023

Опис: Кінцевий варіант

Автор: ПД43 Висоцький В.В.

Вступ

Доступне та надійне програмне забезпечення для аналізу та планування комунальних витрат.

Позиціонування

Економічні передумови:

Поганий контроль комунальних витрат може понести за собою додаткові витрати через додавання додаткового боргу комунальниками.

Формулювання проблеми:

Аналоги такі як записи у блокнот, у віртуальних блокнот та Excel таблиці не мають автоматичних розрахунків. У випадку з Excel потрібно формули складати самому. Через описані вище проблеми, існує ризик помилитися.

Також вище перелічені аналоги не мають автоматичного планування під власні примхи користувача.

Місце системи:

Призначення системи полягає у використанні її звичайним користувачем або платником податків.

Можливе застосування на підприємстві яке буде вести облік платників і їх витрати.

Можливе використання програми у обставинах із обмеженими ресурсами.

Зацікавлені особи

Платники податків які прагнуть більше контролю та аналізу за комунальними витратами.

Підприємства які ведуть облік витрат платників.

Основні задачі високого рівня:

Запропонування користувачу середні витрати за період. Та їх розрахунок.

Задачі рівня користувача:

Реєстрація користувача та ресурсу витрати якого прагнуть аналізувати.

Введення показників лічильника, або інакше витрат.

Реєстрація подій та фіксовані витрати на кожну з них.

Огляд

Перспективи продукту:

Програмне забезпечення буде отримувати показники від користувача та проводити процес калькуляції.

Переваги систем:

Можливість автоматичного розрахунку планування на певний період.

У майбутньому можливість підключення оплати через програмне забезпечення.

Припущення і залежності:

Відсутність монетизації.

Вартість і ціноутворення:

Ціна відсутня так як створення програмного забезпечення в цьому випадку мотивується чистим ентузіазмом.

У майбутньому є можливість додати рекламу або продаж ліцензій на користування «Tosu». Також не виключено що програму можуть викупити підприємці для власних потреб.

Ліцензування і встановлення:

Створити ліцензування системи.

Встановити систему для ліцензування.

Основні властивості системи

Реєстрація користувача.

Реєстрація ресурсу який витрачається.

Введення показників лічильника.

Реєстрація фіксованих витрат для певної події.

Інші вимоги та обмеження:

У майбутньому створити можливість оплати для комунальних витрат.

Пропоную наступне бачення системи у Таблиці 3.1.

Таблиця 3.1 – Бачення системи

Id	Дата	Показання лічильника	Витрата за період	Пропонується витрата за період
1	01.04.23	2201	2	3
2	02.04.23	2203	2	90-2/29=3.03
3	03.04.23	2207	4	90-(2+4)/28=3
4	04.04.23	2212	5	90-(2+4+5)/27=2.93

Пропоновані витрати рахуються наступним чином:

Середнє за день*кількість діб у періоді який вказали + залишок за попередній місяць- витрати в поточному місяці/ кількість діб, що залишилося до кінця місяця.

Ще є таблиця даних щодо витрачення ресурсу за подіями наведено у Таблиці 3.2.

Таблиця 3.2 – Таблиця витрат за подіями

Id	Дата	Посилання на подію	Показання лічильника до події	Показання лічильника після події	Витрата під час події
1	01.04.23	1	2201	2202	1
2	02.04.23	2	2204	2206	2
3	03.04.23	1	2207	2208	1

На наступна таблиця для даних щодо подій.

Таблиця 3.3 – Таблиця посилання на події

Id	Назва події	Посилання на подію	Очікуванні витрати за подію
1	Подія 1	1	1
2	Подія 2	2	2
3	Подія 3	3	5

Що це дає:

Отримання інформації. Чи можна витрачати воду більше ніж зазвичай, чи менше ніж зазвичай.

Отримання інформації. Які події я можу використати, або не використати сьогодні, щоб вкластися у добовий ліміт.

Якщо розглянути взаємодіє між системою та користувачем у контекстній діаграмі, то вигляд має такий (Рис3.4). На діаграмі вказано, що користувач звертається до системи програмного забезпечення під назвою «Tosu». Щоб ввести показники лічильника для подальшого аналізу витрат.

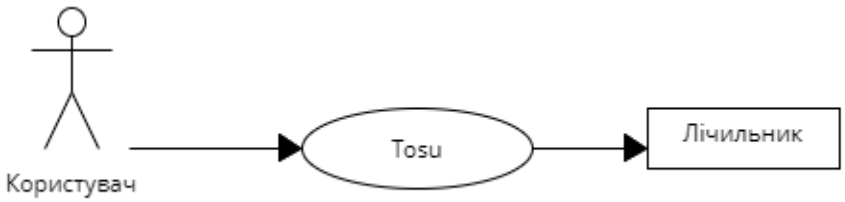


Рисунок 3.4 – Контекстна діаграма

3.1 Модель предметної галузі

На діаграмі вказано (Рис 3.5), що модель User відноситься до моделі KindOfSpending, як один до багатьох. Тобто один юзер може мати багато KindOfSpending. Також відношення моделі KindOfSpending до SpendingPerEvent а також до SpendingPerPeriod є одним до багатьох. Тобто один KindOfSpending може мати багато SpendingPerEvent і в цей же час багато SpendingPerPeriod. І теж саме відношення у SpendingPerEvent до Event.

Розглянемо як працює система. Користувач під час реєстрації вводить ім’я і прізвище. Потім юзер може ввести декілька «видів витрат». Під час введення «видів витрат» юзер повинен ввести назву витрати, номер показника. .

Далі юзер може обрати «витрати за період» або «витрати за подію». Якщо юзер обирає «витрати за період», то він повинен ввести дату, показник, а програма введе середні витрати і юзер Id.

При введені «витрат за подію». Користувач повинен ввести дату, показник до і після події. Після чого він отримує пропоновані витрати за подію.

У моделі Event потрібно ввести назву події та витрати на цю подію.

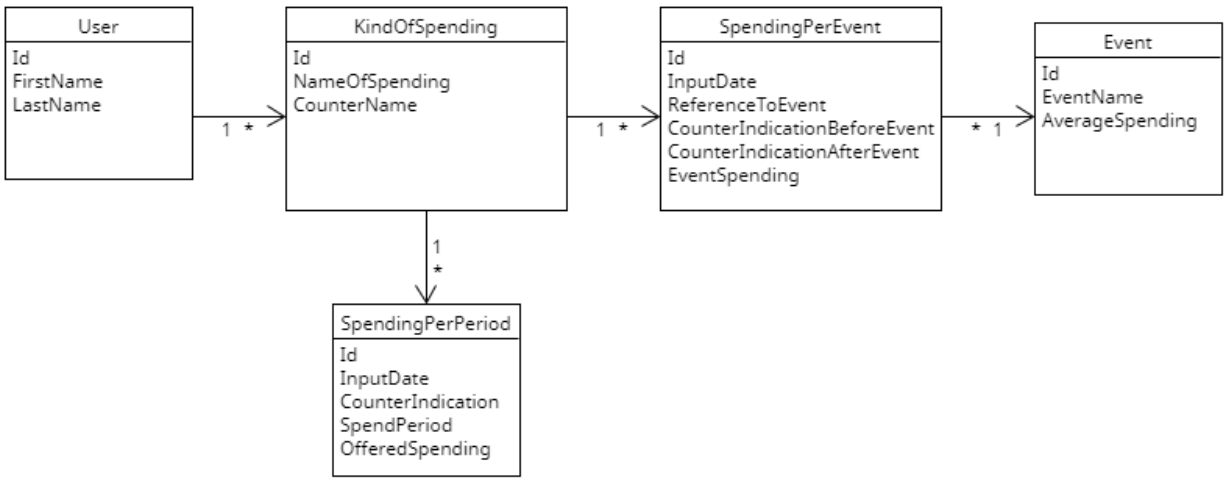


Рисунок 3.5 – Діаграма предметної галузі або діаграма моделювання даних

Нижче приведені фрагменти коду які мають відношення до діаграми предметної галузі.(Рис 3.6, Рис 3.7, Рис 3.8, Рис 3.9, Рис 3.10)

```
internal class EventDescription : IId
{
    public int Id { get; set; }
    public string EventName { get; set; }
    public double AverageSpending { get; set; }
    public string Description { get; set; }
    public EventDescription(string eventName, string description)
    {
        EventName = eventName;
        Description = description;
    }

    public EventDescription(string eventName, double averageSpending, string description)
    {
        EventName = eventName;
        AverageSpending = averageSpending;
        Description = description;
    }
    public override string ToString()
    {
        return string.Format(Id + " " + EventName + " " + AverageSpending + " " + Description);
    }
}
```

Рисунок 3.6 – Модель EventDescription

```
internal class EventSpending : IId
{
    public int Id { get; set; }
    public int EventId { get; set; }
    public DateTime DateTimeEventBefore { get; set; }
    public double CounterValueBefore { get; set; }
    public DateTime DateTimeEventAfter { get; set; }
    public double CounterValueAfter { get; set; }
    public double SpendingEvent { get; set; }
    public double SpendingSpeed { get; set; }
    public double AverageSpending { get; set; }

    public EventSpending(int eventId, DateTime dateTimeEventBefore, double counterValueBefore)
    {
        EventId = eventId;
        DateTimeEventBefore = dateTimeEventBefore;
        CounterValueBefore = counterValueBefore;
    }

    public EventSpending(int eventId, DateTime dateTimeEventBefore, double counterValueBefore,
        DateTime dateTimeEventAfter, double counterValueAfter, double spendingEvent, double spendingSpeed,
        double averageSpending) : this(eventId, dateTimeEventBefore, counterValueBefore)
    {
        DateTimeEventAfter = dateTimeEventAfter;
        CounterValueAfter = counterValueAfter;
        SpendingEvent = spendingEvent;
        SpendingSpeed = spendingSpeed;
        AverageSpending = averageSpending;
    }

    public override string ToString()
    {
        return string.Format(Id + " " + EventId + " " + DateTimeEventBefore +
        " " + CounterValueBefore + " " + DateTimeEventAfter + " " + CounterValueAfter +
        " " + SpendingEvent + " " + SpendingSpeed + " " + AverageSpending);
    }
}
```

Рисунок 3.7 – Модель EventSpending

```
internal class KindOfSpending : IId
{
    public int Id { get; set; }
    public string NameOfSpending { get; set; }
    public string CounterName { get; set; }

    public KindOfSpending(string nameOfSpending, string counterName)
    {
        NameOfSpending = nameOfSpending;
        CounterName = counterName;
    }
    public override string ToString()
    {
        return string.Format(Id + " " + NameOfSpending + " " + CounterName);
    }
}
```

Рисунок 3.8 – Модель KindOfSpending

```
internal class State : IId
{
    public int Id { get; set; }
    public DateTime CurrentDate { get; set; }
    public double CounterValue { get; set; }
    public double ExpensesForPeriod { get; set; }
    public double SuggestedSpending { get; set; }
    public State(DateTime currentDate, double counterValue)
    {
        CurrentDate = currentDate;
        CounterValue = counterValue;
    }
    public State(DateTime currentDate, double counterValue, double expenses, double suggestedSpending)
    {
        CurrentDate = currentDate;
        CounterValue = counterValue;
        ExpensesForPeriod = expenses;
        SuggestedSpending = suggestedSpending;
    }

    public override string ToString()
    {
        return string.Format(" {0} {1: 0.00} {2} {3: 0.00} {4: 0.00}", Id, CounterValue, CurrentDate.Date.
    }
}
```

Рисунок 3.9 – Модель State

```
internal class User : IId
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public User(string firstName, string lastName)
    {
        FirstName = firstName;
        LastName = lastName;
    }
    public override string ToString()
    {
        return String.Format(Id + " " + FirstName + " " + LastName);
    }
}
```

Рисунок 3.10 – Модель User

3.2 Визначення функціональних вимог

- облік витрат на за період;
- обчислення запропонованої витрати за період;
- облік витрат на подію;
- облік подій;

Далі на діаграмі прецедентів розглянемо функціональні вимоги(Рис 3.11)

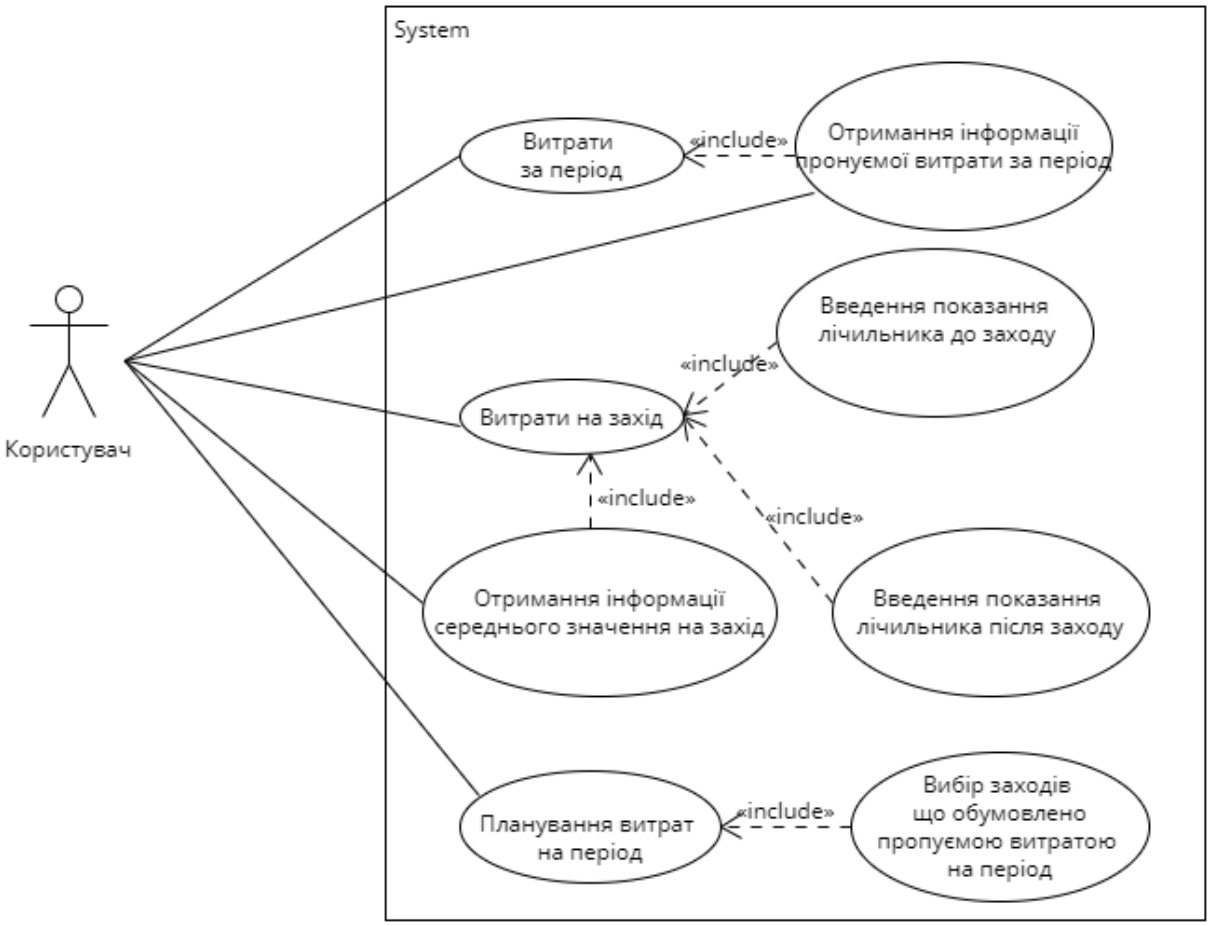


Рисунок 3.11 – Діаграма прецедентів

Прецедент: Витрати за період.

Рівень: "Задача, визначена користувачем".

Виконавець: Користувач.

Передумови: Застосування увімкнуто, обрана витрата яку буде заповнювати користувач.

Результат: Витрати за період внесені у базу даних.

Основний успішний сценарій:

Система питає у користувача показання лічильника;

Користувач вводить показання лічильника;

Система питає у користувача дату;

Користувач вводить дату;

Система питає у користувача кількість запланованих витрат;

Користувач вводить кількість запланованих витрат;

Система питає у користувача період для запланованих витрат;

Користувач вводить період для запланованих витрат;

Розширення (альтернативні потоки):

При введенні неправильних значень:

Система пропонує користувачу ввести значення заново.

Спеціальні вимоги:

Можливість монетизації програми через рекламу або продаж ліцензій. Або продаж самої програми.

Можливість нарощування системи, а саме: додавання нових даних; нової логіки додавання даних, нового середовища відображення даних.

Список технологій і типів даних: Віндовс форми мови C#, платформа .NET Framework.

Частота використання: щорічно.

Прецедент: створення витрат на подію.

Рівень: "Задача, визначена користувачем".

Виконавець: Користувач.

Передумови: Застосування увімкнуто обраний тип витрат.

Результат: Витрати на подію створені і внесені у базу даних.

Основний успішний сценарій:

Система питає у користувача інформацію про назву події.

Користувач вводить назву події;

Система питає у користувача інформацію про спеціальний номер події.

Користувач вводить спеціальний номер події;

Система питає у користувача інформацію очікувані витрати на подію.

Користувач вводить очікувану витрати на подію;

Розширення (альтернативні потоки):

При введенні неправильних значень:

Система пропонує користувачу ввести значення заново.

Спеціальні вимоги:

Можливість монетизації програми через рекламу або продаж ліцензій. Або продаж самої програми.

Можливість нарощування системи, а саме: додавання нових даних; нової логіки додавання даних, нового середовища відображення даних.

Список технологій і типів даних: Віндовс форми мови C#, платформа .NET Framework.

Частота використання: щорічно.

Відкриті питання: Вивчити питання взаємодії цього модуля з іншими модулями.

3.3 Визначення нефункціональних вимог

Додаткова специфікація

Дати внесення змін:

Версія: Кінцева версія

Дата: 15.02.2023

Опис: Кінцевий варіант

Автор: ПД 43 Висоцький В.В.

Вступ

Нефункціональні вимоги:

- операційна система Windows;
- середовище розробки Visual Studio;
- мова програмування C#;
- база даних SQL Server.

У цьому документі описані всі вимоги до системи що не увійшли до опису прецедентів.

Наскрізна функціональність

Реєстрація подій і обробка помилок:

Всі події реєструються в окремому файлі, всі помилки реєструються в окремому файлі.

Бізнес правила, які можна підключити:

Монетизація через рекламу або продаж ліцензій. Або продаж самої програми

Безпека:

Необхідно виконувати аутентифікацію усіх користувачів.

Зручність використання

Людський фактор

Користувачі системи будуть працювати з великим монітором (не з мобільним телефоном), тому необхідне таке:

текст повинен бути видно з відстані 1 метра;

Швидка, проста і коректна обробка інформації - це головні принципи системи.

Повідомлення про введення хибної інформації необхідно супроводжувати звуковими сигналами.

Надійність

Можливість відновлення інформації:

Необхідно забезпечити локальне збереження даних. Це питання потребує детальної проробки.

Продуктивність

Наша задача - виконати прецеденти не довше ніж за 1 хвилину.

Конфігурування

Система може налаштовуватися. Це питання потребує проробки.

Обмеження

Керівництво проекту вимагає використовувати технологію .NET.

Придбані компоненти

Придбання компонентів на цей час не планується.

Безкоштовні компоненти на основі відкритого коду

Використання безкоштовних компонентів не планується.

4 МОДЕЛЬ ПРОЕКТУВАННЯ

Проектування діяльності

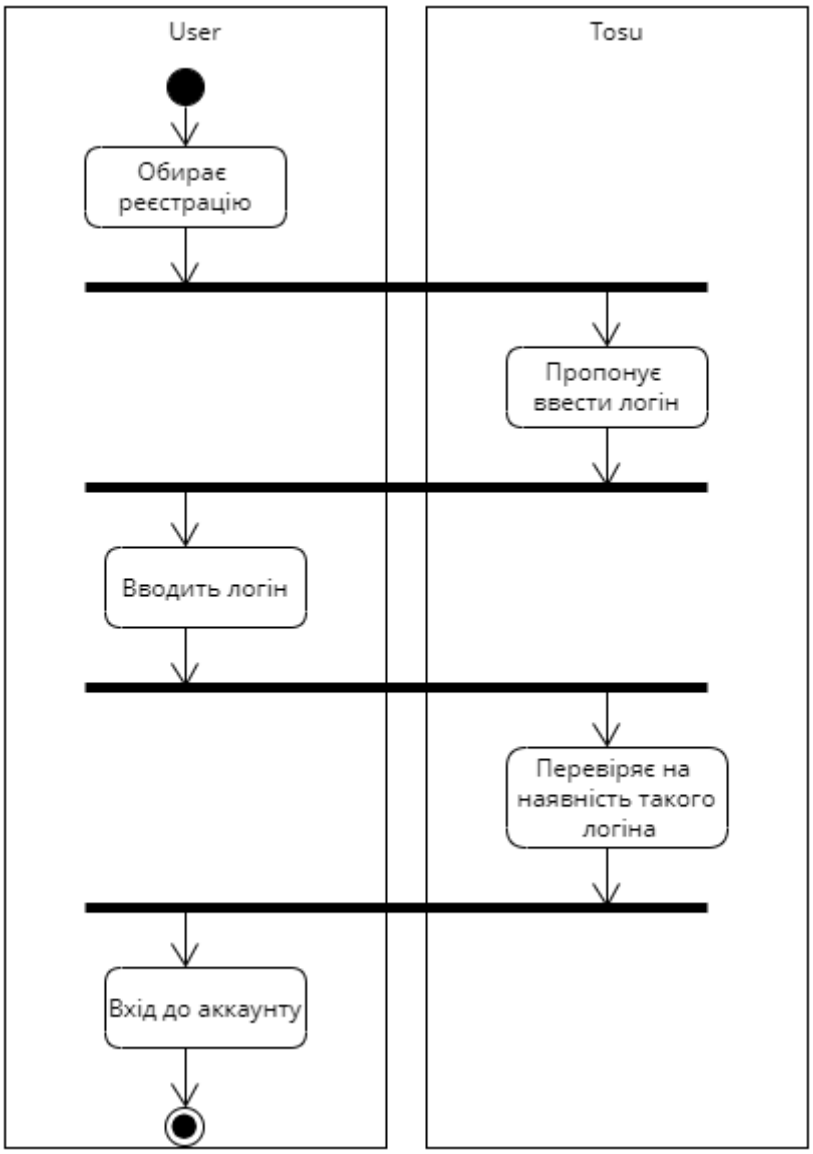


Рисунок 3.12 – Діаграма діяльності прецеденту реєстрації

На цій діаграмі зображений процес реєстрації. Після точки входу у програму користувач обирає реєстрацію. Після чого до програми іде запит на реєстрацію. Програма відповідає пропозицією ввести логін. Користувач вводить. У системі проходить пошук на наявність такого логіна. І тільки потім вхід до аккаунту.

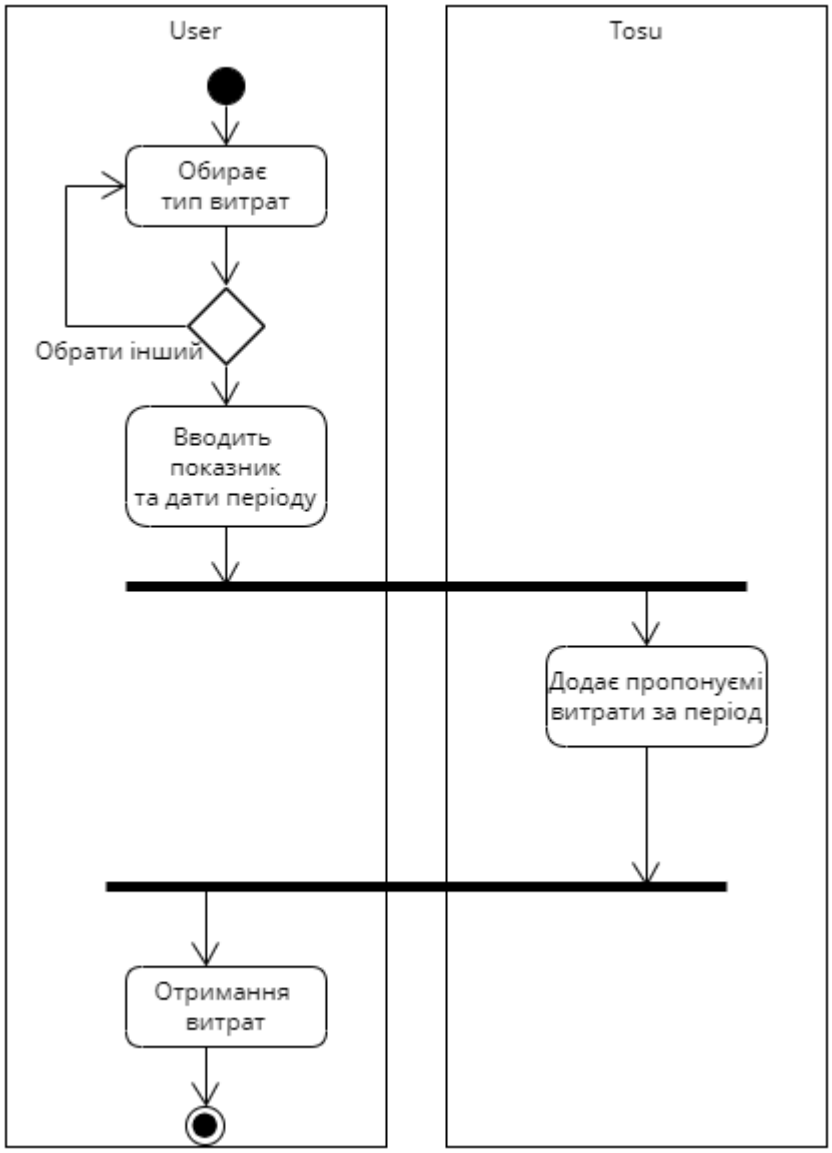


Рисунок 3.13 – Діаграма діяльності прецеденту витрат за період

На цій діаграмі зображений прецедент витрат на період. Спочатку користувач обирає тип витрат. Наприклад воду, газ тощо. Або обирає інший. Потім користувач вводить показник та дати за період у який були витрати. До системи надходить запит. Відбувається калькуляція значень. І програма додає пропонуємі витрати на добу якщо користувач хоче їх отримати. В кінці користувач отримує свої витрати.

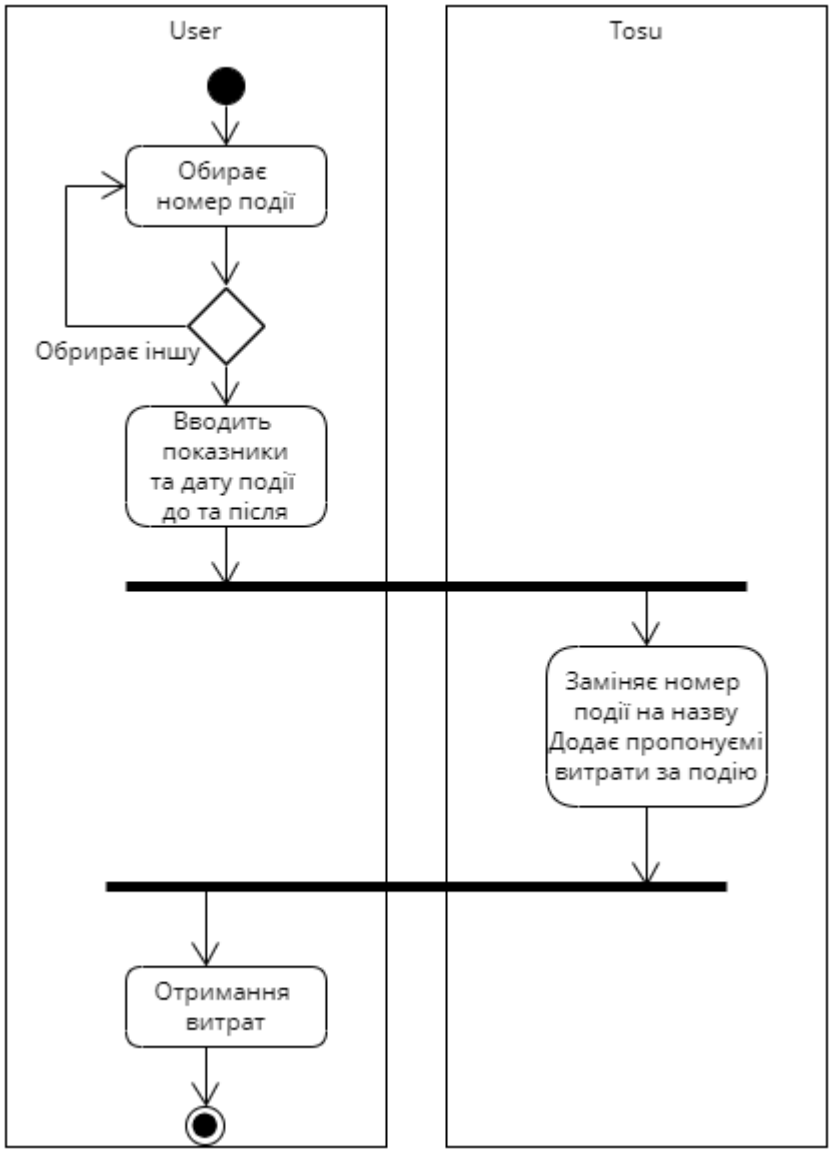


Рисунок 3.14 – Діаграма діяльності прецеденту витрат за подію

На цій діаграмі зображений прецедент обирання витрат на події. Спочатку користувач обирає спеціальний номер події, який до цього був занесений у базу даних. Є можливість обрання іншої події. Потім користувач вводить показники та дату до і після події. Після чого йде запит у програму, яка в свою чергу заміняє спец номер на назву події та додає пропоновану витрату за подію якщо цього бажає користувач. В кінці користувач отримує витрату за подію.

4.1 Проектування класів та їх взаємодію

Так як вся діаграма класів системи надто велика, то вона буде презентована у декількох частинах. Presentation Layer

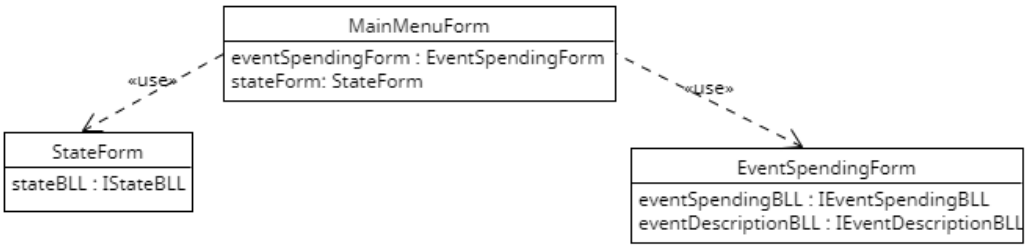


Рисунок 3.16 – Діаграма класів рівня Presentation Layer

Business Logic Layer

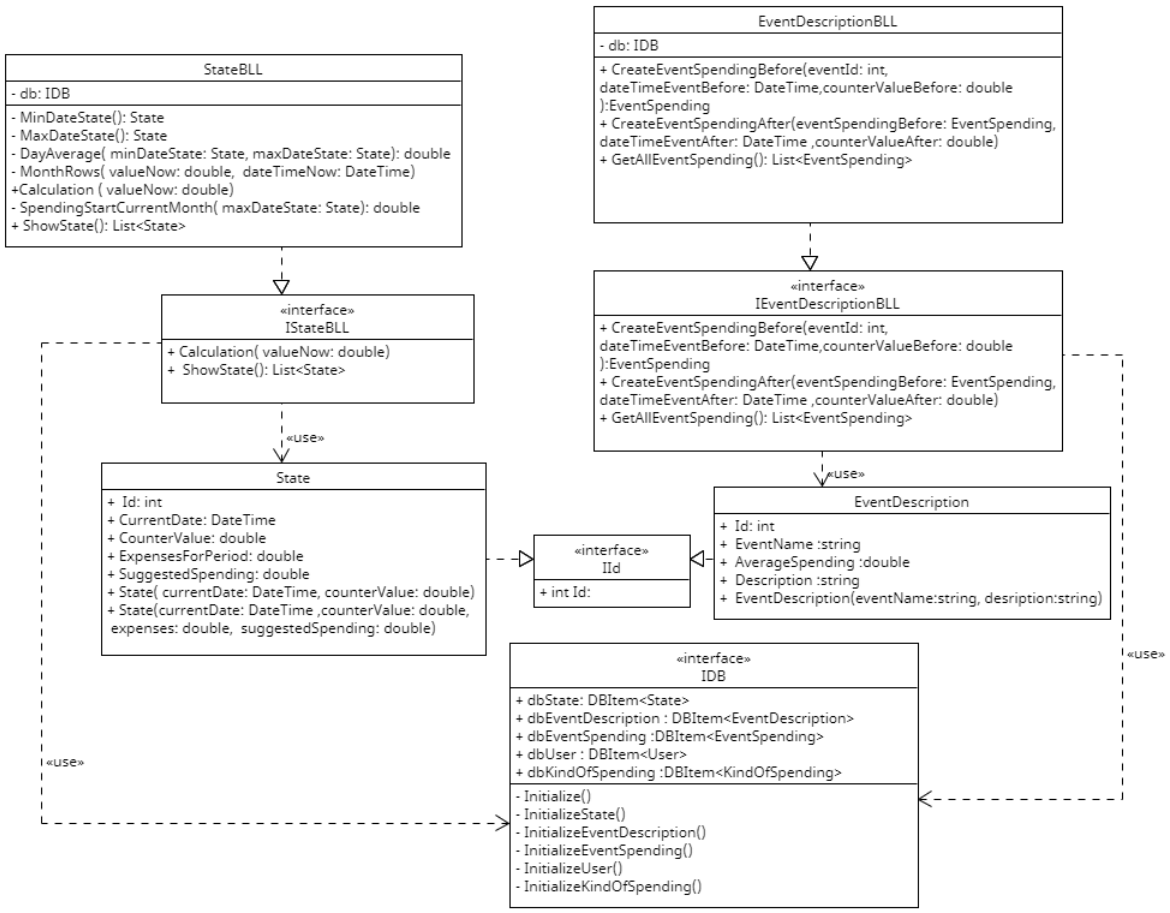


Рисунок 3.17 – Перша частина діаграми класів рівня business logic layer

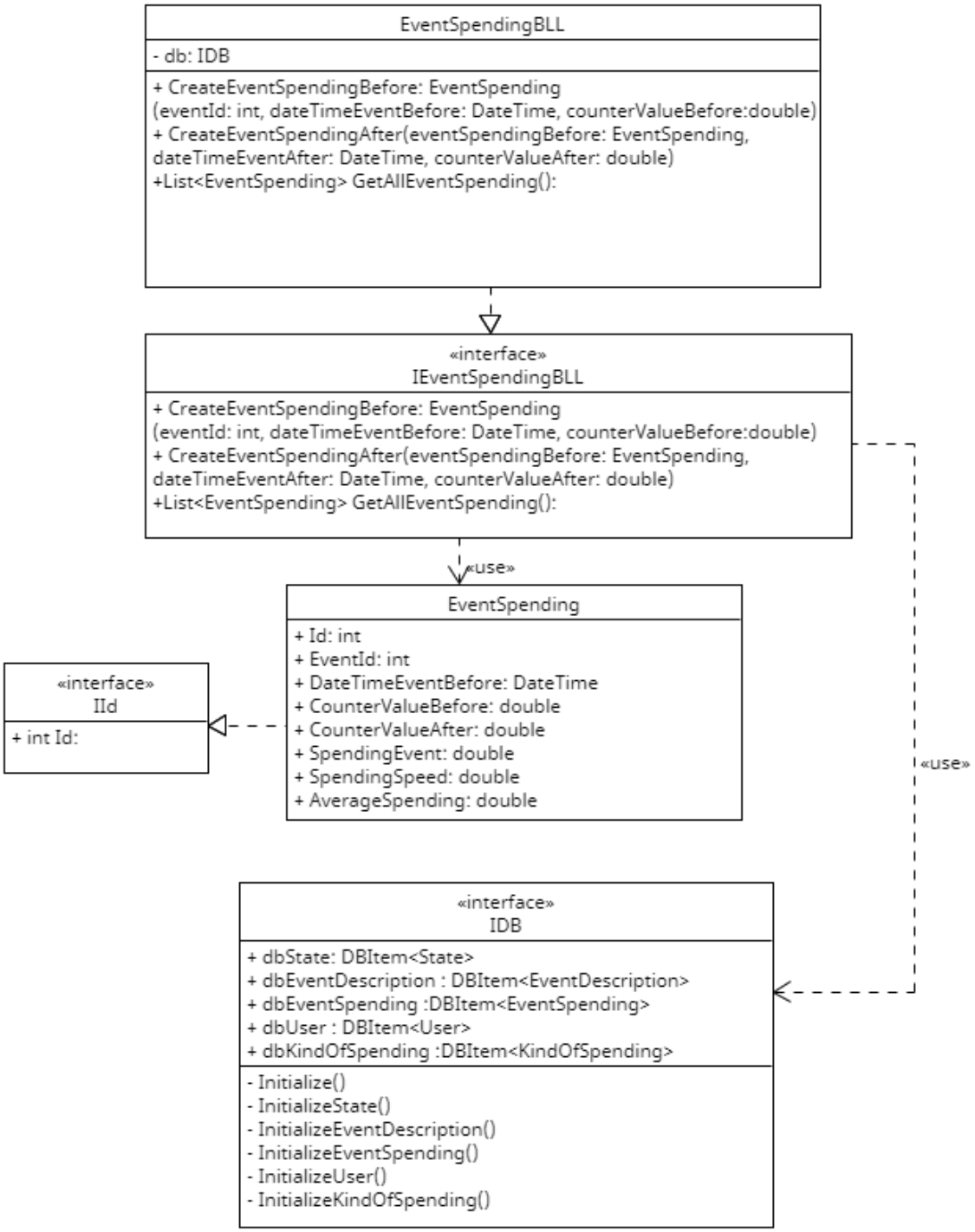


Рисунок 3.18 – Друга частина діаграми класів рівня business logic layer

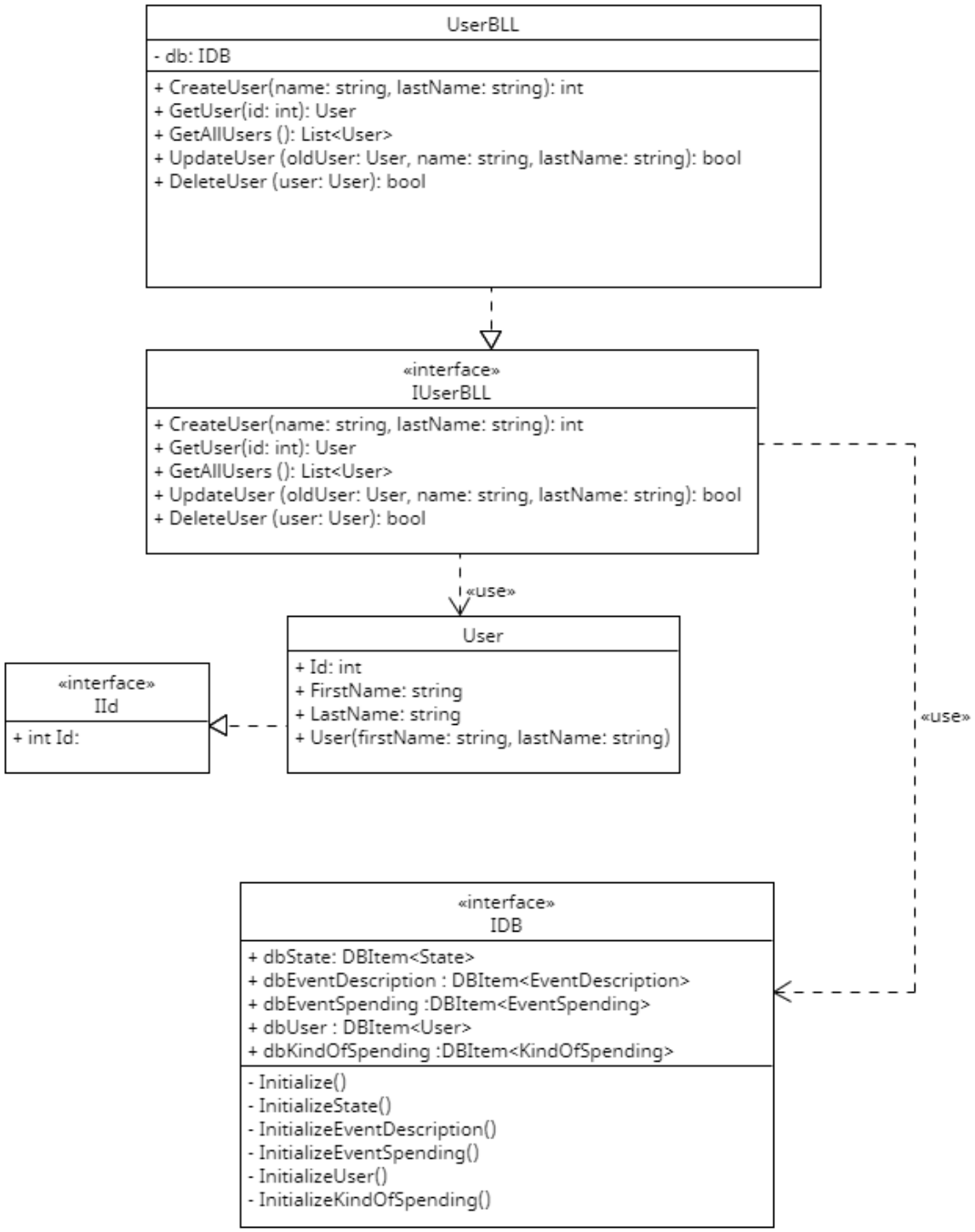


Рисунок 3.19 – Третя частина діаграми класів рівня business logic layer

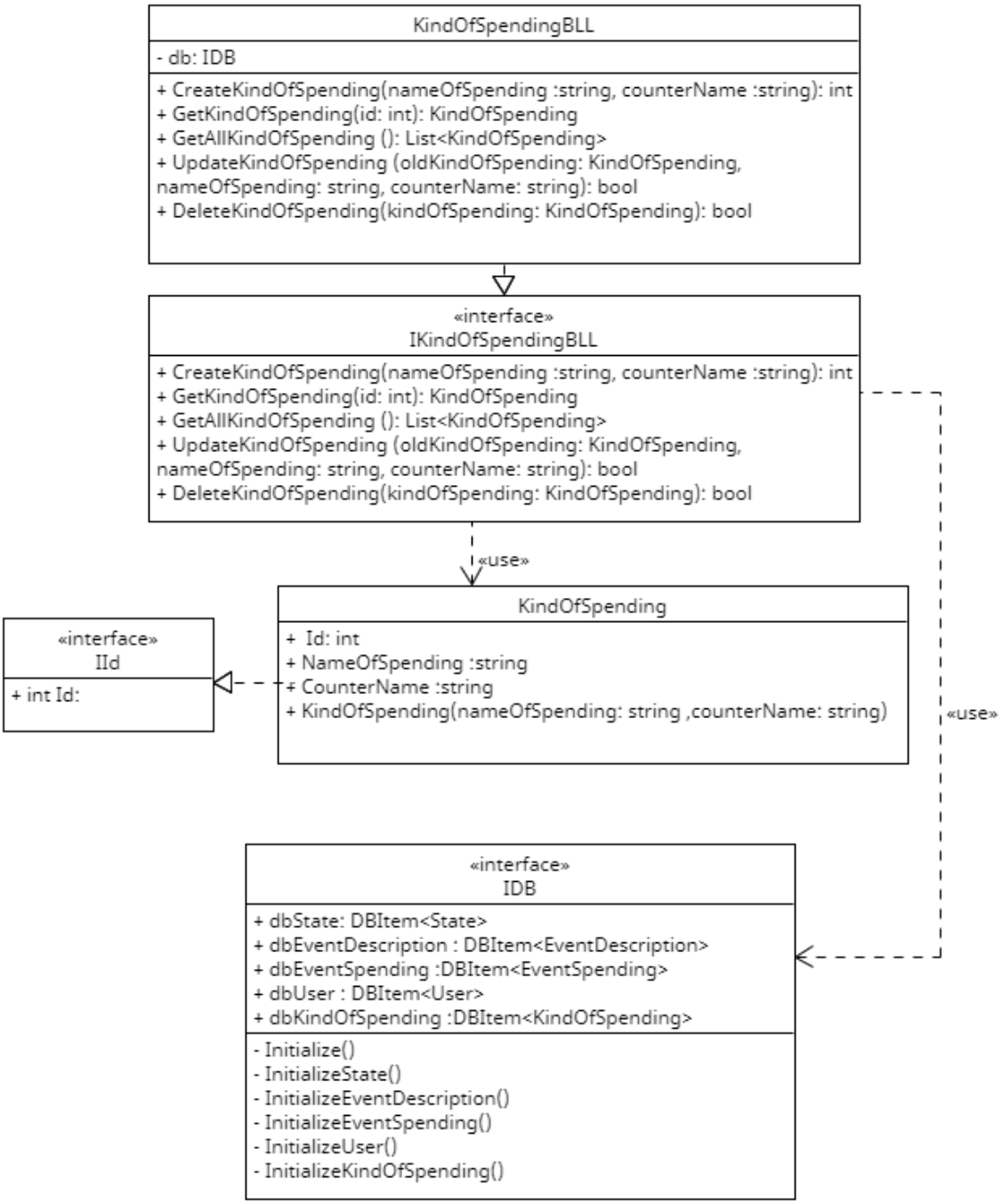


Рисунок 3.20 – Четверта частина діаграми класів рівня business logic layer Data access layer

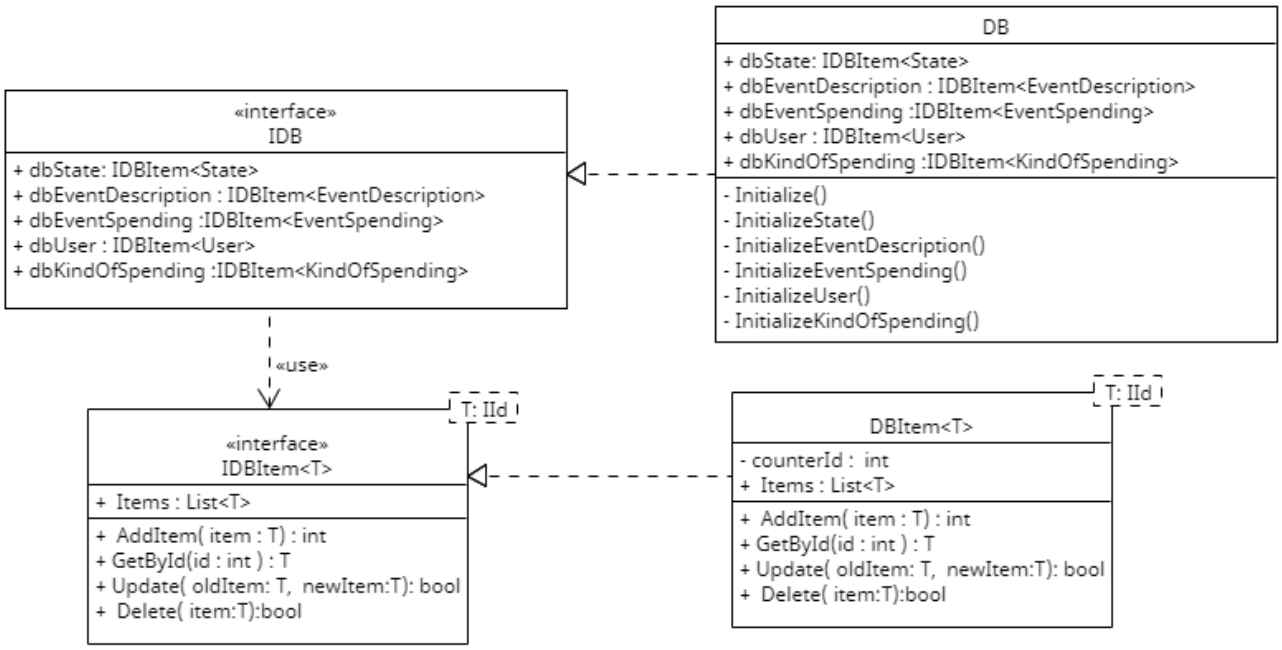


Рисунок 3.21 – Діаграма класів рівня Data access layer

4.2 Проектування станів системи.

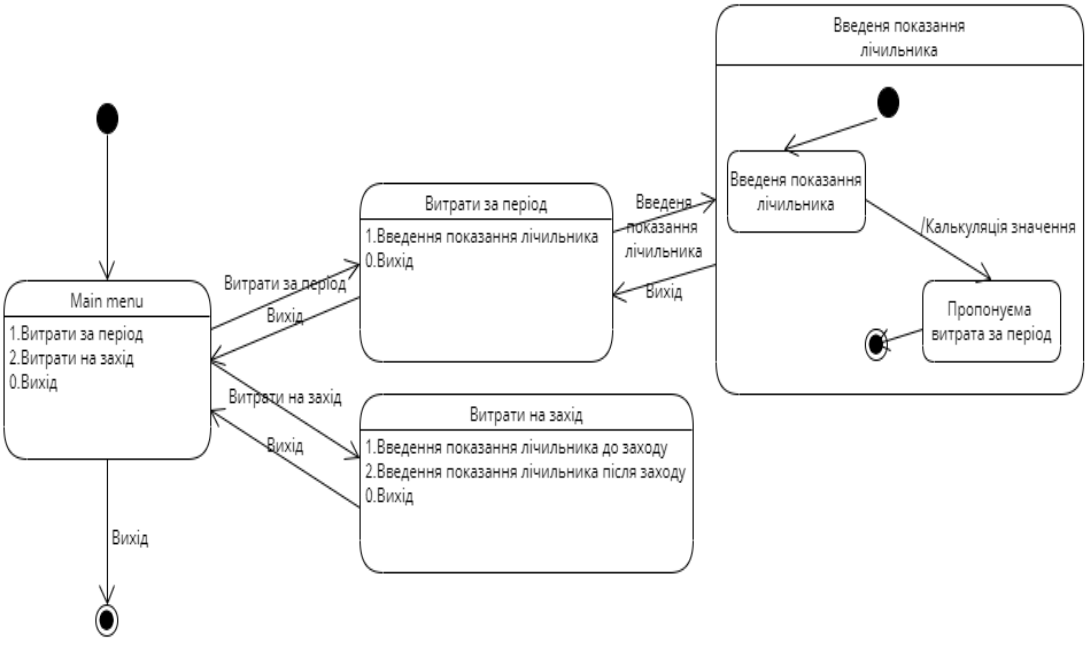


Рисунок 3.22 – Діаграма станів

Після точки входу користувач потрапляє у головне меню. Є три варіанти. перший витрати за період. Другий витрати на захід. І вихід із програми. Якщо

користувач обирає витрати за період, то він перейде до наступного меню, де буде доступні на вибір обрання типу витрати і запит для введення лічильника. Або вийти із програми. Якщо користувач обирає введення показання. То відбувається калькуляція значень. І з звичайними витратами буде пропонована витрата за період. Є варіант обрати витрати на захід. Там користувач повинен обрати тип витрат і ввести показання лічильника до заходу і після. Також ввести спеціальний номер події. Після чого він отримує свої витрати і пропонована витрата за подію.

Проектування станів об’єктів їх кількості.

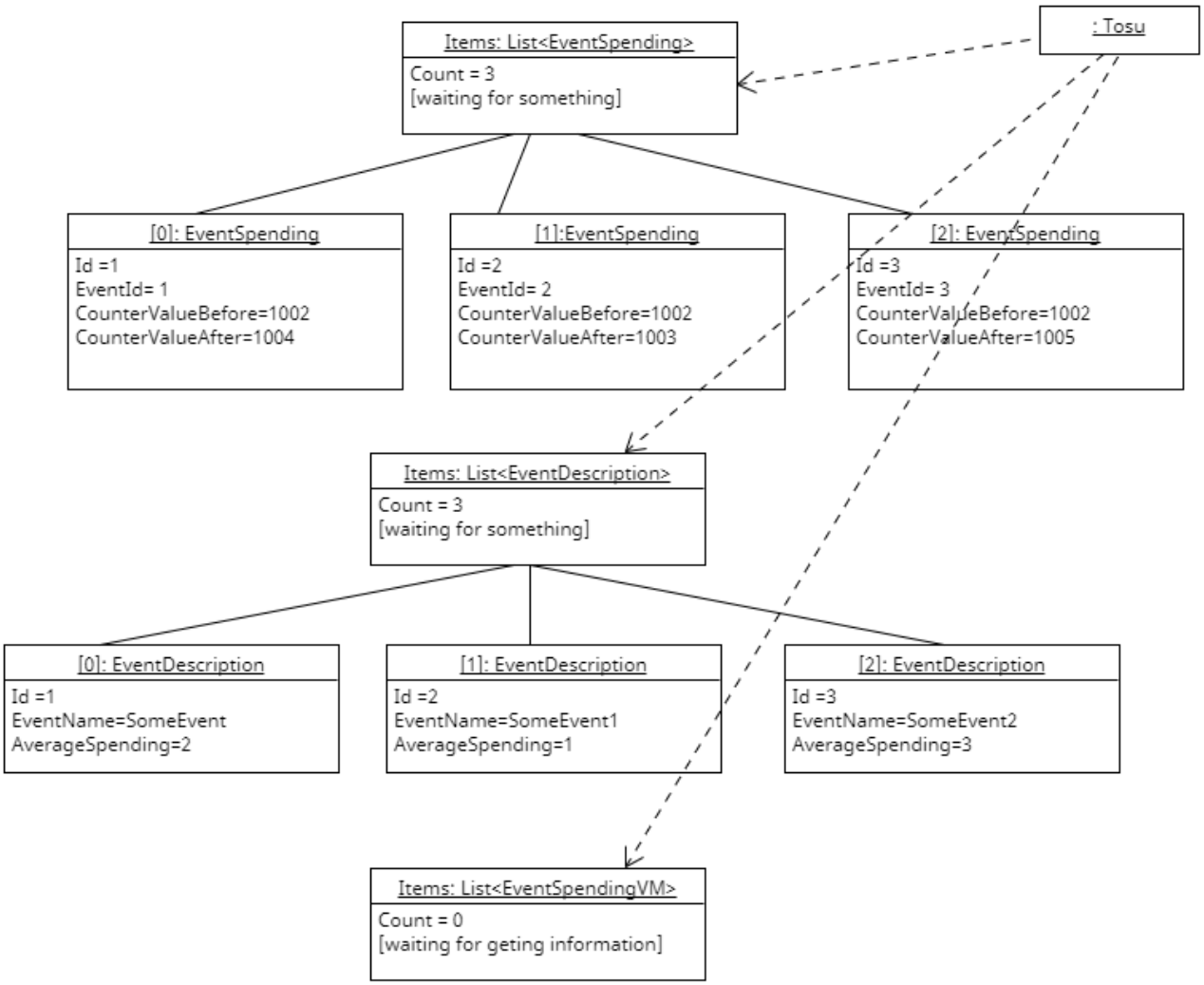


Рисунок 3.23 – Діаграма об’єктів до події

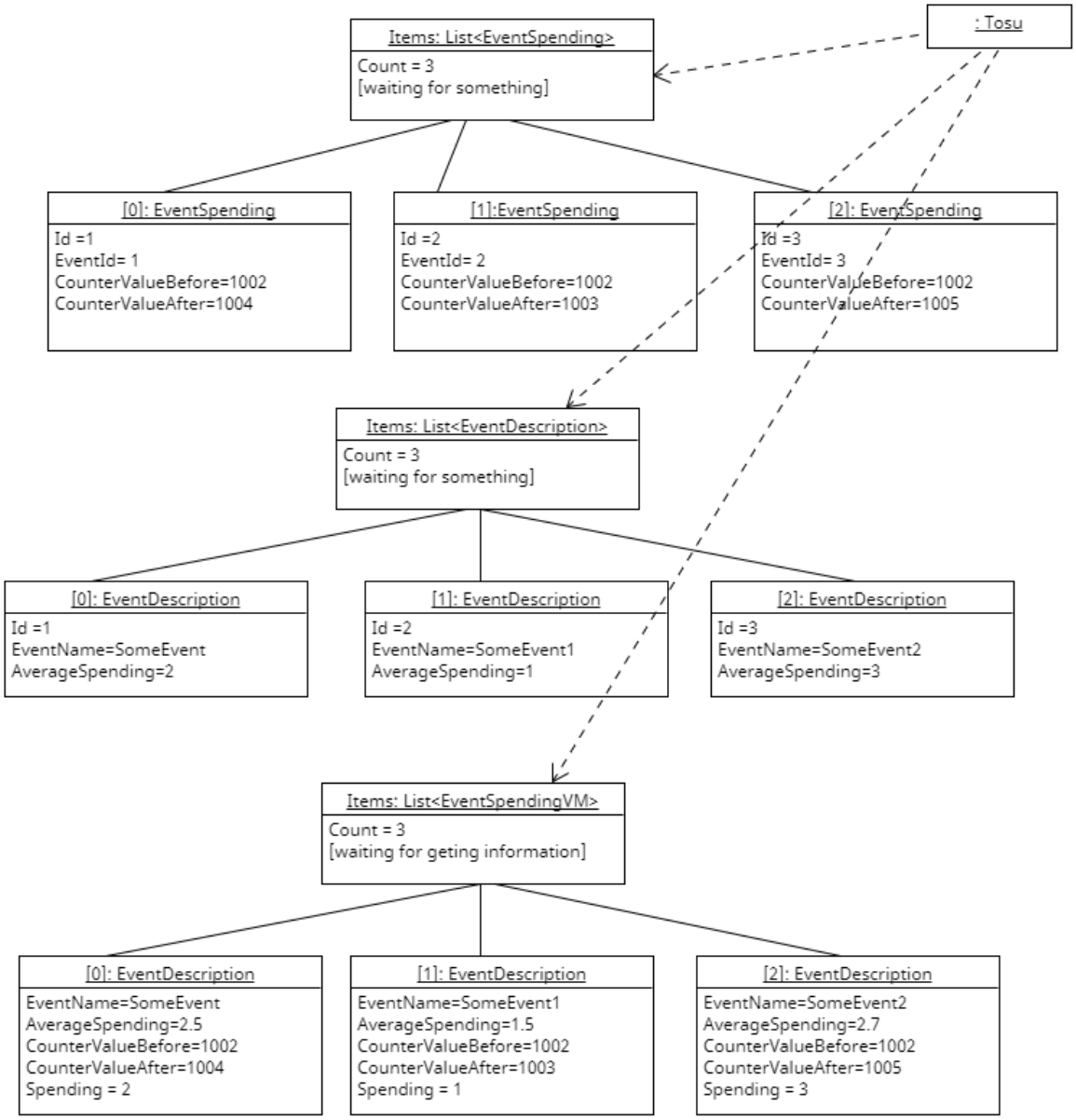


Рисунок 3.24 – Діаграма об’єктів після події

4.3 Проектування розгортання системи

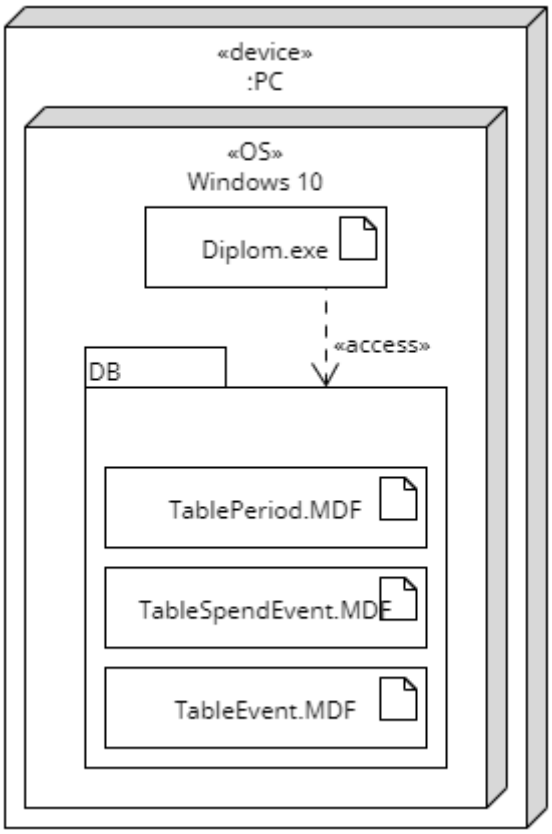


Рисунок 3.25 – Діаграма розгортання

Програма «Tosu» використовує персональний комп’ютер. Операційну систему відновс. Проект називається Diplom і має тип файлу exe. Проект має базу даних SQL у якій існує три таблиці. Таблиця для витрат на період. Таблиця для витрат подій. Таблиця для номерів подій.

4.4 Проектування Архітектури

На рисунку 3.5 зображена діаграма на якій три рівня архітектури класів програми «Tosu». Перший це presentation layer. Це рівень інтерфейсу програми. Тут користувач може взаємодіяти із програмою.

Другий це Bussiness Logick Layer. Це рівень калькуляції значень, розрахунку. Тут відбувається процес додавання та зміни даних. У цього рівня існують моделі, які окремо можна виділити. Вони приймають значення від користувача.

Третій це Data access layer. Це рівень зберігання та інформації. Інакше кажучи база даних. Сюди вносяться правки та зміни для їх зберігання.

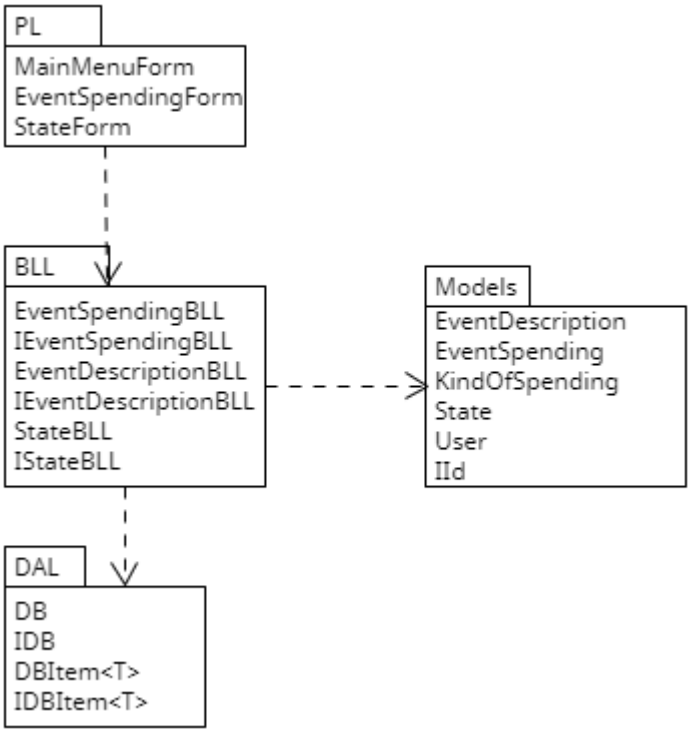


Рисунок 3.26 – Діаграма архітектури

Діаграма артефактів

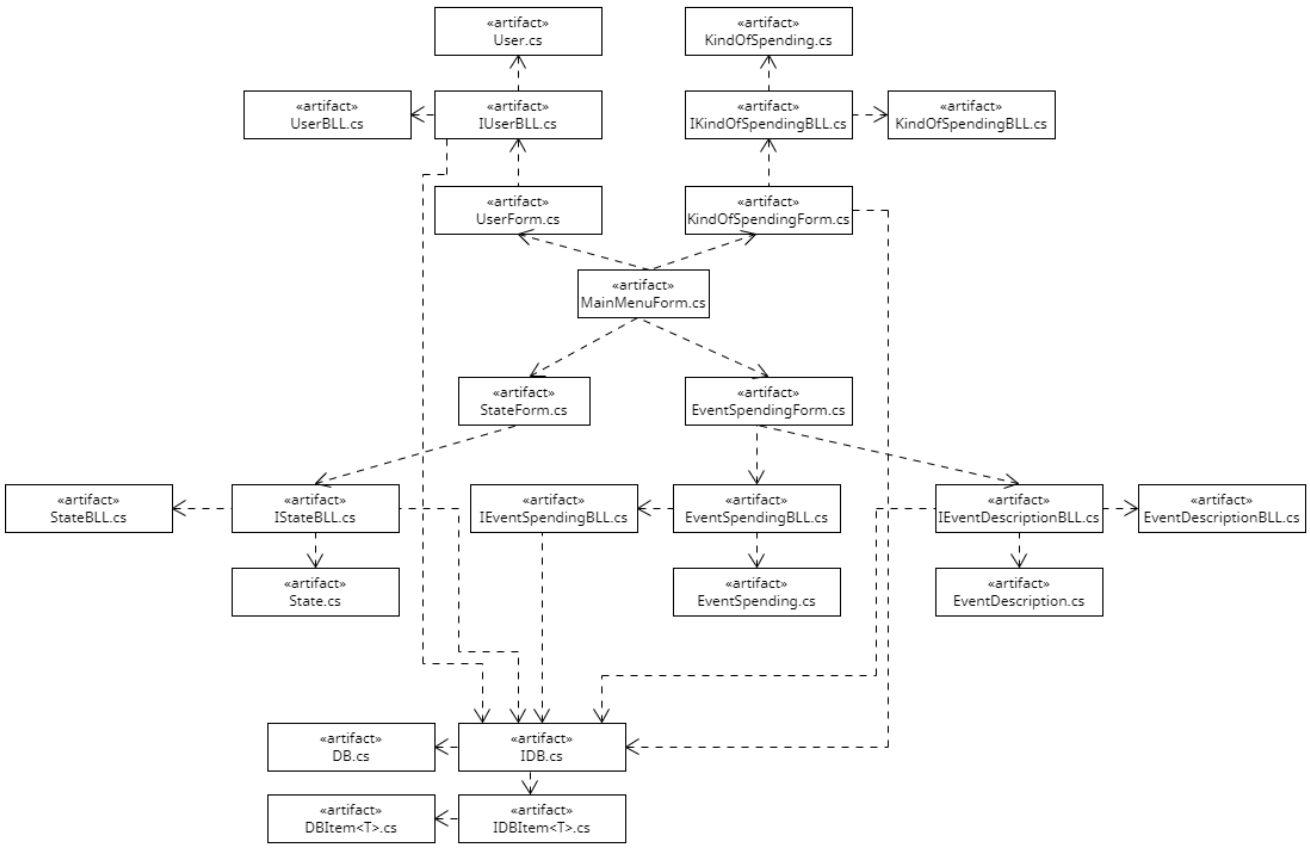


Рисунок 3.27 – Діаграма артефактів

5. ТЕСТУВАННЯ СИСТЕМИ

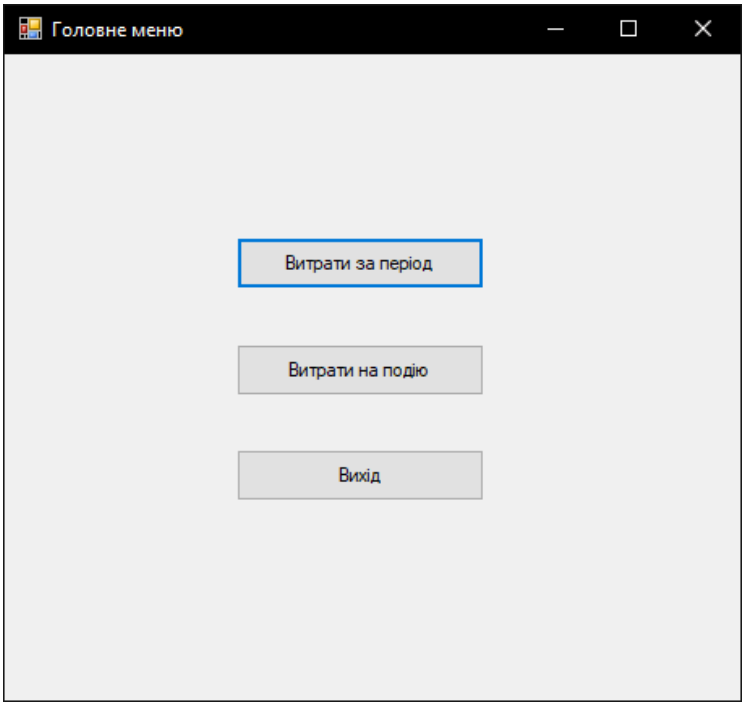


Рисунок 3.28 – Головне меню

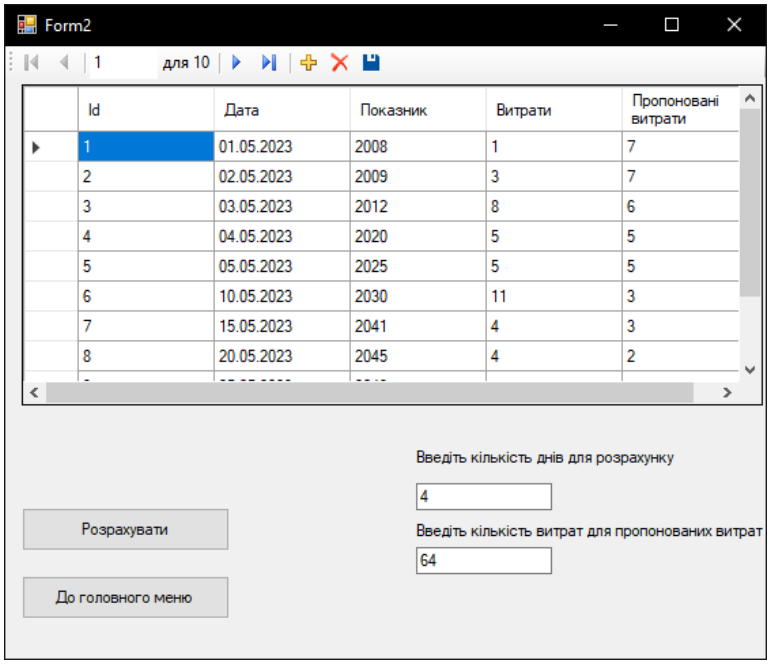


Рисунок 3.29 – Розрахунок витрат

6. ВИСНОВКИ

1. Опрацьовано документ бачення системи з визначенням місця і ролі інформаційної системи. Складено контекстну діаграму.
2. Проаналізовано предметну галузь. Змодельовано модель предметної галузі. Відпрацьовано діаграму предметної галузі.
3. Визначено функціональні вимоги. Змодельовано модель прецедентів. Відпрацьовано діаграма прецедентів.
4. Визначено нефункціональні вимоги. Опрацьовано документ додаткова специфікація.
5. Здійснено проектування діяльності. Опрацьовано діаграми діяльності.
6. Здійснено проектування взаємодії об’єктів певних методів. Створено діаграми послідовності.
7. Розроблено проектування три-рівневої архітектури. Опрацьовано діаграму пакетів.
8. Здійснено проектування взаємодії об’єктів. Опрацьовано діаграму класів .
9. Змодульовано меню застосунку. Опрацьовано діаграму станів системи.
10. Розроблено проектування розгортання системи. Опрацьовано діаграму розгортання системи.
11. Визначено об’єм виконаної роботи. Опрацьовано діаграму артефактів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Read Online the Free Book “Fundamentals of Computer Programming with C#” [Електронний ресурс] – Режим доступу до ресурсу: https://introprogramming.info/english-intro-csharp-book/read-online/.
2. UMLetino. *umletino*. URL: https://www.umletino.com/umletino.html
3. Free Book + Video Course "Programming Basics with C#" [Електронний ресурс] // Faber Publishing, Sofia,. – 2019. – Режим доступу до ресурсу: https://csharp-book.softuni.org/.
4. Навчання кодуванню в Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/
5. Software framework [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Software_framework.
6. C Sharp (programming language) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/C_Sharp_(programming_language).
7. Shengalts A. akelpad [Електронний ресурс] / Aleksander Shengalts. – 2003. – Режим доступу до ресурсу: https://akelpad.sourceforge.net/ru/index.php.
8. Microsoft Excel [Електронний ресурс]. – 1987. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/microsoft-365/excel.

ДОДАТОК А- Код програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CounterProject.DAL;
using CounterProject.Models;

namespace CounterProject.BLL
{
    internal class EventDescriptionBLL
    {
        DB db;
        public EventDescriptionBLL(DB db)
        {
            this.db = db;
        }
        public int CreateEvent(string eventName, string description)
        {
            EventDescription eventDescription = new EventDescription(eventName, description);
            int id = db.dbEventDescription.AddItem(eventDescription);
            return id;
        }
        public List<EventDescription> GetEvents()
        {
            return db.dbEventDescription.Items;
        }
    }
```

```
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CounterProject.DAL;
using CounterProject.Models;

namespace CounterProject.BLL
{
   internal class EventSpendingBLL
   {
      DB db;

      public EventSpendingBLL(DB db)
      {
         this.db = db;
      }
      public EventSpending CreateEventSpendingBefore(int eventId, DateTime dateTimeEventBefore, double counterValueBefore)
      {
         EventSpending eventSpending = new EventSpending(eventId, dateTimeEventBefore, counterValueBefore);
         db.dbEventSpending.AddItem(eventSpending);
         return eventSpending;
      }
      public void CreateEventSpendingAfter(EventSpending eventSpendingBefore, DateTime dateTimeEventAfter, double counterValueAfter)
      {
```

```
int id = eventSpendingBefore.Id;
int eventId = eventSpendingBefore.EventId;
DateTime dateTimeEventBefore = eventSpendingBefore.DateTimeEventBefore;
double counterValueBefore = eventSpendingBefore.CounterValueBefore;

double spendingEvent = counterValueAfter - counterValueBefore;
double spendingTime = (dateTimeEventAfter - dateTimeEventBefore).TotalHours;
double spendingSpeed = spendingTime != 0 ? spendingEvent / spendingTime : 0;

double averageSpending = 0;
double sum = 0;
int qty = 0;
foreach (EventSpending es in db.dbEventSpending.Items)
{
   if (es.EventId == eventId)
   {
      if (es.Id != id)
      {
         sum += es.SpendingEvent;
      }
      else
      {
         sum += spendingEvent;

      }
      qty++;
   }
}
if (qty != 0)
```

```
            {
                averageSpending = sum / qty;
            }
            else
            {
                averageSpending = 0;
            }
            foreach (EventDescription ed in db.dbEventDescription.Items)
            {
                if (ed.Id == eventId)
                {
                    ed.AverageSpending = averageSpending;
                }
            }
            EventSpending eventSpendingAfter = new EventSpending(eventId, dateTimeEventBefore, counterValueBefore, dateTimeEventAfter, counterValueAfter, spendingEvent, spendingSpeed, averageSpending);
            bool result = db.dbEventSpending.Update(eventSpendingBefore, eventSpendingAfter);
        }
        public List<EventSpending> GetAllEventSpending()
        {
            return db.dbEventSpending.Items;
        }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using CounterProject.DAL;
using CounterProject.Models;

namespace CounterProject.BLL
{
    internal class StateBll
    {
        DB db;

        public StateBll(DB db)
        {
            this.db = db;
        }
        private State MinDateState()
        {
            State result = default(State);
            DateTime minDate = DateTime.MaxValue;
            foreach (State s in db.dbState.Items)
            {
                if (s.CurrentDate < minDate)
                {
                    minDate = s.CurrentDate;
                    result = s;
                }
            }
            return result;
        }
        private State MaxDateState()
        {
            State result = default(State);
```

```
    DateTime maxDate = DateTime.MinValue;
    foreach (State s in db.dbState.Items)
    {
        if (s.CurrentDate > maxDate)
        {
            maxDate = s.CurrentDate;
            result = s;
        }
    }
    return result;
}

private double DayAverage(State minDateState, State maxDateState)
{
    int dayQty = (maxDateState.CurrentDate - minDateState.CurrentDate).Days;
    double spanValue = maxDateState.CounterValue - minDateState.CounterValue;
    double dayAverage = dayQty != 0 ? spanValue / dayQty : 0;
    return dayAverage;
}
private void MonthRows(double valueNow, DateTime dateTimeNow)
{
    State minDateState = MaxDateState();
    int dayQty = (dateTimeNow - minDateState.CurrentDate).Days;
    double spanValue = valueNow - minDateState.CounterValue;
    double dayAverage = dayQty != 0 ? spanValue / dayQty : 0;

    DateTime currentDay = minDateState.CurrentDate;
    int oldCurrentMonth = currentDay.Month;
    int newCurrentMonth = oldCurrentMonth;
    int qtyDays = 0;
```

```
            double currentValue = minDateState.CounterValue;
            while (currentDay < dateTimeNow)
            {
                if (newCurrentMonth > oldCurrentMonth)
                {
                    oldCurrentMonth = newCurrentMonth;
                    currentValue = currentValue + qtyDays * dayAverage;
                    State state = new State(currentDay, currentValue);
                    db.dbState.AddItem(state);
                    qtyDays = 0;
                }
                currentDay = currentDay.AddDays(1);
                qtyDays++;
                newCurrentMonth = currentDay.Month;
            }
        }
        public void Calculation(double valueNow)
        {
            DateTime dateTimeNow = DateTime.Now;

            MonthRows(valueNow, dateTimeNow);

            State state = new State(dateTimeNow, valueNow);
            db.dbState.AddItem(state);

            State minDateState = MinDateState();
            int daysPerMonth = DateTime.DaysInMonth(dateTimeNow.Year,
dateTimeNow.Month);
            double dayAverage = DayAverage(minDateState, state);
```

```
        double spendingCurrentMonth = state.CounterValue -
SpendingStartCurrentMonth(state);
        int daysLeft = daysPerMonth - state.CurrentDate.Day;
        state.SuggestedSpending = (dayAverage * daysPerMonth -
spendingCurrentMonth) / daysLeft;

        int index = db.dbState.Items.IndexOf(state);
        state.ExpensesForPeriod = state.CounterValue - db.dbState.Items[index -
1].CounterValue;
    }
    private double SpendingStartCurrentMonth(State maxDateState)
    {
        double result = 0;
        DateTime currentDay = maxDateState.CurrentDate;
        int oldCurrentMonth = currentDay.Month;
        int newCurrentMonth = oldCurrentMonth;
        DateTime firstDayOfMonth = currentDay;
        foreach (State s in db.dbState.Items)
        {
            if (s.CurrentDate.Month == currentDay.Month && s.CurrentDate.Day == 1)
            {
                result = s.CounterValue;
            }
        }
        return result;
    }
    public List<State> ShowState()
    {
        return db.dbState.Items;
    }
```

```
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CounterProject.Models;

namespace CounterProject.DAL
{
    internal class DB
    {
        public DBItem<State> dbState { get; set; } = new DBItem<State>();
        public DBItem<EventDescription> dbEventDescription { get; set; } = new DBItem<EventDescription>();
        public DBItem<EventSpending> dbEventSpending { get; set; } = new DBItem<EventSpending>();
        public DBItem<User> dbUser { get; set; } = new DBItem<User>();
        public DBItem<KindOfSpending> dbKindOfSpending { get;set; } = new DBItem<KindOfSpending>();
        public DB()
        {
            Initialize();
        }
        private void Initialize()
        {
            InitializeState();
            InitializeEventDescription();
            InitializeEventSpending();
```

```
            InitializeUser();
            InitializeKindOfSpending();
        }
        private void InitializeState()
        {
            State state1 = new State(DateTime.Parse("01.04.23"), 2201, 2, 3);
            dbState.AddItem(state1);
            State state2 = new State(DateTime.Parse("02.04.23"), 2203, 2, 3.3);
            dbState.AddItem(state2);
            State state3 = new State(DateTime.Parse("03.04.23"), 2207, 4, 3);
            dbState.AddItem(state3);
            State state4 = new State(DateTime.Parse("04.04.23"), 2212, 5, 2.93);
            dbState.AddItem(state4);
        }
        private void InitializeEventDescription()
        {
            EventDescription eventDescription1 = new EventDescription("event1",
"description1");
            dbEventDescription.AddItem(eventDescription1);
            EventDescription eventDescription2 = new EventDescription("event2",
"description2");
            dbEventDescription.AddItem(eventDescription2);
            EventDescription eventDescription3 = new EventDescription("event3",
"description3");
            dbEventDescription.AddItem(eventDescription3);
        }
        private void InitializeEventSpending()
        {
```

```
            EventSpending eventSpending1 = new EventSpending(1, DateTime.Parse("01.04.2023 8:30"), 2202, DateTime.Parse("01.04.2023 9:30"), 2203, 1, 1, 1);
            dbEventSpending.AddItem(eventSpending1);
            EventSpending eventSpending2 = new EventSpending(1, DateTime.Parse("02.04.2023 8:30"), 2204, DateTime.Parse("02.04.2023 9:30"), 2205, 1, 1, 1);
            dbEventSpending.AddItem(eventSpending2);
            EventSpending eventSpending3 = new EventSpending(2, DateTime.Parse("02.04.2023 10:30"), 2205, DateTime.Parse("02.04.2023 11:30"), 2206, 1, 1, 1);
            dbEventSpending.AddItem(eventSpending3);
            EventSpending eventSpending4 = new EventSpending(3, DateTime.Parse("02.04.2023 12:30"), 2207, DateTime.Parse("02.04.2023 13:30"), 2208, 1, 1, 1);
            dbEventSpending.AddItem(eventSpending4);
            EventSpending eventSpending5 = new EventSpending(3, DateTime.Parse("03.04.2023 8:30"), 2208, DateTime.Parse("03.04.2023 9:30"), 2209, 1, 1, 1);
            dbEventSpending.AddItem(eventSpending5);
        }
        private void InitializeUser()
        {
            User user1 = new User("UserN1", "UserF1");
            dbUser.AddItem(user1);
            User user2 = new User("UserN2", "UserF2");
            dbUser.AddItem(user2);
        }
        private void InitializeKindOfSpending()
        {
```

```
            KindOfSpending kindOfSpending1 = new KindOfSpending("KindOfSpending1", "Counter1");
            dbKindOfSpending.AddItem(kindOfSpending1);
            KindOfSpending kindOfSpending2 = new KindOfSpending("KindOfSpending2", "Counter2");
            dbKindOfSpending.AddItem(kindOfSpending2);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CounterProject.Models;

namespace CounterProject.DAL
{
    internal class DBItem<T> where T : IId
    {
        private int counterId = 1;
        public List<T> Items { get; set; }

        public DBItem()
        {
            Items = new List<T>();
        }
        public int AddItem(T item)
        {
            item.Id = counterId++;
```

```
            Items.Add(item);
            return item.Id;
        }
        public T GetById(int id)
        {
            return Items.FirstOrDefault(item => item.Id == id);
        }
        public bool Update(T oldItem, T newItem)
        {
            newItem.Id = oldItem.Id;
            bool result = Items.Remove(oldItem);
            Items.Add(newItem);
            return result;
        }
        public bool Delete(T item)
        {
            return Items.Remove(item);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CounterProject.Models
{
    internal class EventDescription : IId
    {
```

```
        public int Id { get; set; }
        public string EventName { get; set; }
        public double AverageSpending { get; set; }
        public string Description { get; set; }
        public EventDescription(string eventName, string description)
        {
            EventName = eventName;
            Description = description;
        }

        public EventDescription(string eventName, double averageSpending, string
description)
        {
            EventName = eventName;
            AverageSpending = averageSpending;
            Description = description;
        }
        public override string ToString()
        {
            return string.Format(Id + " " + EventName + " " + AverageSpending + " " +
Description);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CounterProject.Models
{
    internal class EventSpending : IId
    {
        public int Id { get; set; }
        public int EventId { get; set; }
        public DateTime DateTimeEventBefore { get; set; }
        public double CounterValueBefore { get; set; }
        public DateTime DateTimeEventAfter { get; set; }
        public double CounterValueAfter { get; set; }
        public double SpendingEvent { get; set; }
        public double SpendingSpeed { get; set; }
        public double AverageSpending { get; set; }

        public EventSpending(int eventId, DateTime dateTimeEventBefore, double counterValueBefore)
        {
            EventId = eventId;
            DateTimeEventBefore = dateTimeEventBefore;
            CounterValueBefore = counterValueBefore;
        }

        public EventSpending(int eventId, DateTime dateTimeEventBefore, double counterValueBefore, DateTime dateTimeEventAfter, double counterValueAfter, double spendingEvent, double spendingSpeed, double averageSpending) : this(eventId, dateTimeEventBefore, counterValueBefore)
        {
            DateTimeEventAfter = dateTimeEventAfter;
            CounterValueAfter = counterValueAfter;
            SpendingEvent = spendingEvent;
```

```
            SpendingSpeed = spendingSpeed;
            AverageSpending = averageSpending;
        }

        public override string ToString()
        {
            return string.Format(Id + " " + EventId + " " + DateTimeEventBefore + " " + CounterValueBefore + " " + DateTimeEventAfter + " " + CounterValueAfter + " " + SpendingEvent + " " + SpendingSpeed + " " + AverageSpending);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CounterProject.Models
{
    internal interface IId
    {
        int Id { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CounterProject.Models
{
    internal class KindOfSpending : IId
    {
        public int Id { get; set; }
        public string NameOfSpending { get; set; }
        public string CounterName { get; set; }

        public KindOfSpending(string nameOfSpending, string counterName)
        {
            NameOfSpending = nameOfSpending;
            CounterName = counterName;
        }
        public override string ToString()
        {
            return string.Format(Id + " " + NameOfSpending + " " + CounterName);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CounterProject.Models
{
    internal class State : IId
    {
```

```
        public int Id { get; set; }
        public DateTime CurrentDate { get; set; }
        public double CounterValue { get; set; }
        public double ExpensesForPeriod { get; set; }
        public double SuggestedSpending { get; set; }
        public State(DateTime currentDate, double counterValue)
        {
            CurrentDate = currentDate;
            CounterValue = counterValue;
        }
        public State(DateTime currentDate, double counterValue, double expenses, double
suggestedSpending)
        {
            CurrentDate = currentDate;
            CounterValue = counterValue;
            ExpensesForPeriod = expenses;
            SuggestedSpending = suggestedSpending;
        }

        public override string ToString()
        {
            return string.Format(" {0} {1: 0.00} {2} {3: 0.00} {4: 0.00}", Id, CounterValue,
CurrentDate.Date.ToShortDateString(), ExpensesForPeriod, SuggestedSpending);
        }
    }
}
using CounterProject.Models;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace CounterProject
{
    internal class User : IId
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

        public User(string firstName, string lastName)
        {
            FirstName = firstName;
            LastName = lastName;
        }
        public override string ToString()
        {
            return String.Format(Id + " " + FirstName + " " + LastName);
        }
    }
}
using CounterProject.BLL;
using CounterProject.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CounterProject.PL
{
    internal class StatePL
    {
        StateBll stateBll;

        public StatePL(StateBll stateBll)
        {
            this.stateBll = stateBll;
        }

        public void CreateState()
        {
            Console.Write("Value?: ");
            double valueNow = double.Parse(Console.ReadLine());
            stateBll.Calculation(valueNow);
        }
        public void ShowState()
        {
            foreach (State s in stateBll.ShowState())
            {
                Console.WriteLine(s);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplomat
{
    public partial class Form1 : Form
    {
        private Button button1;
        private Button button2;
        private Button button3;
        Form2 settingsForm = new Form2();
        Form3 settingsForm1 = new Form3();
        public Form1()
        {
            InitializeComponent();
        }

        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(143, 113);
```

```
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(153, 32);
this.button1.TabIndex = 0;
this.button1.Text = "Витрати за період";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.button1_Click);
//
// button2
//
this.button2.Location = new System.Drawing.Point(143, 179);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(153, 32);
this.button2.TabIndex = 1;
this.button2.Text = "Витрати на подію";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);
//
// button3
//
this.button3.Location = new System.Drawing.Point(143, 244);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(153, 32);
this.button3.TabIndex = 2;
this.button3.Text = "Вихід";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);
//
// Form1
//
this.ClientSize = new System.Drawing.Size(454, 399);
```

```
    this.Controls.Add(this.button3);
    this.Controls.Add(this.button2);
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.Text = "Головне меню";
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);

}

private void button1_Click(object sender, EventArgs e)
{
    settingsForm.Show();

}

private void button2_Click(object sender, EventArgs e)
{
    settingsForm1.Show();

}

private void button3_Click(object sender, EventArgs e)
{
    this.Close();
}

private void Form1_Load(object sender, EventArgs e)
{
```

```
        }
    }
}using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplomat
{
    public partial class Form2 : Form
    {
        DataTable table = new DataTable();
        int indexRow;

        public Form2()
        {
            InitializeComponent();
        }

        private void tableBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.tableBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.database1DataSet);
```

```
}

private void Form2_Load(object sender, EventArgs e)
{
   this.tableTableAdapter.Fill(this.database1DataSet.Table);

}
int sum = 0, c = 0; int a = 0; int b = 0;
double z = 0;



private void button1_Click(object sender, EventArgs e)
{

   c = int.Parse(textBox1.Text);

   DataGridViewRow newDataRow = tableDataGridView.Rows[indexRow];
   for (int x = 0, y = 1; x < c; x++, y++)
   {
      a = int.Parse(tableDataGridView.Rows[x].Cells[2].Value.ToString()); ;
      b = int.Parse(tableDataGridView.Rows[y].Cells[2].Value.ToString()); ;
      sum = (-1) * (a - b);
      textBox1.Text = sum.ToString();
      tableDataGridView.Rows[x].Cells[3].Value = textBox1.Text;

      textBox2.Text = Convert.ToString( Convert.ToDouble(
int.Parse(textBox2.Text)- sum ) );
      z = Convert.ToDouble(int.Parse(textBox2.Text) / c);
      tableDataGridView.Rows[x].Cells[4].Value = z;
```

```
            }

        }


    }
}
```

ДОДАТОК Б- Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для аналізу комунальних витрат користувача мовою C#

Виконав студент 4 курсу

групи ПД 43

Висоцький Владислав Вікторович

Керівник роботи

Старший викладач кафедри Гаманюк Ігор Михайлович

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи-** спрощення процесу аналізу комунальних витрат за рахунок застосування компютерного програмного забезпечення створеного на мові C#.
- **Об'єкт дослідження-** аналіз комунальних витрат
- **Предмет дослідження-** програмне забезпечення для аналізу комунальних витрат.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати аналоги предметної галузі.
2. Проаналізувати функціональні та не функціональні вимоги до програмного забезпечення.
3. Проаналізувати програмні засоби реалізації.
4. Розробити діаграми станів, предметної галузі, прецендетнів, класів.
5. Розробити Програмне забезпечення на мові програмування C# під назвою «Tosu».

3

АНАЛІЗ АНАЛОГІВ

Засіб	Wallet	Akelpad	Excel	Tosu
Платформа	Android, iOs, Web	Windows	Windows,Android	Windows
Автоматичні розрахунки	+	-	Тільки якщо формулу впише людина	+
Автоматичне заповнення показників	+	-	-	+
Великий ризик помилитися у заповненні та розрахунках	-	+	+	-
Легкий у користуванні	-	+	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. *Функціональні вимоги:*

- *облік витрат за добу;*
- *обчислення пропонуємої витрати на добу;*
- *облік витрат за захід;*
- *облік заходів;*

2. *Нефункціональні вимоги:*

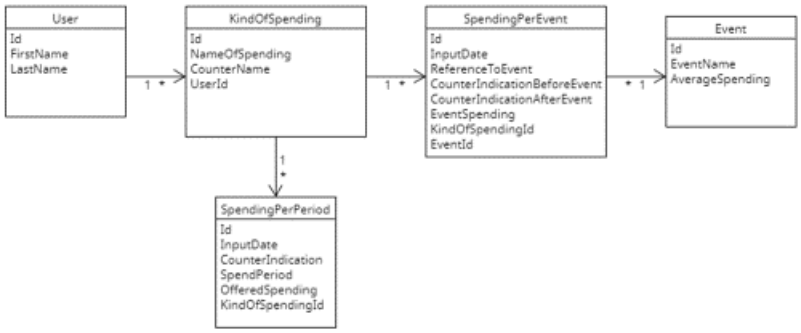
- *операційна система Windows;*
- *середовище розробки Visual Studio;*
- *мова програмування C#;*
- *база даних SQL Server.*

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

Діаграма предметної галузі

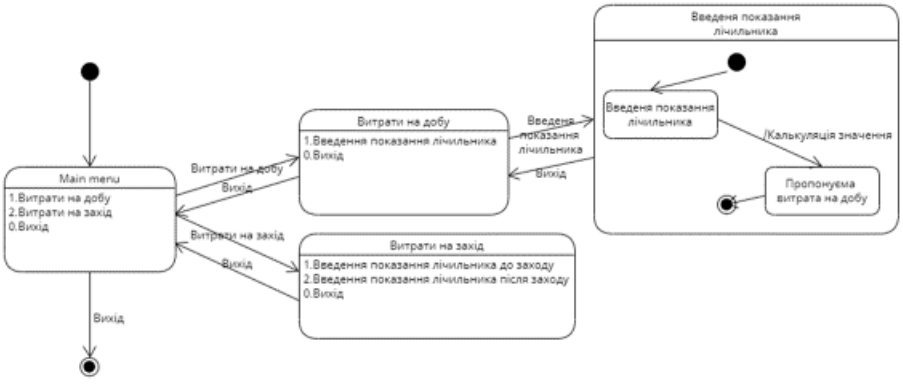


7

Діаграма варіантів використання системи

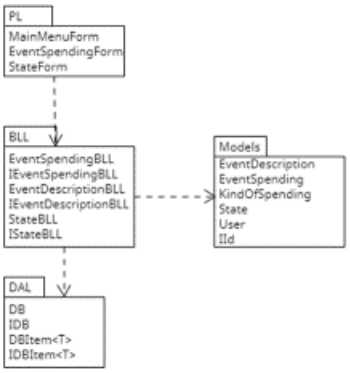
8

Діаграма станів



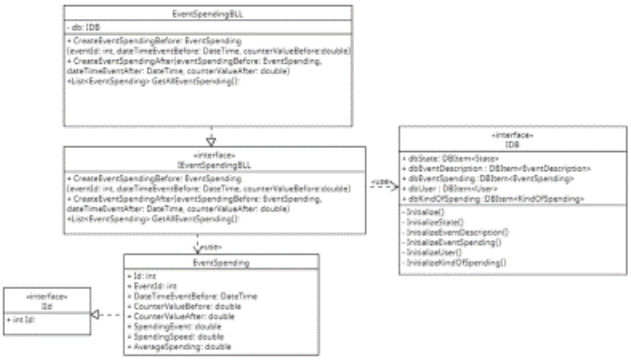
9

Діаграма архітектури



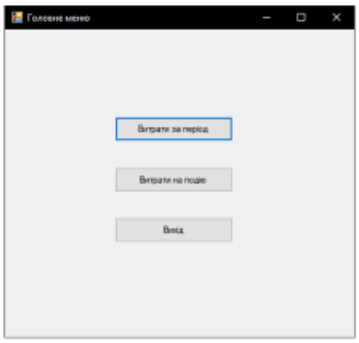
10

Діаграма класів рівня Business Logic Layer

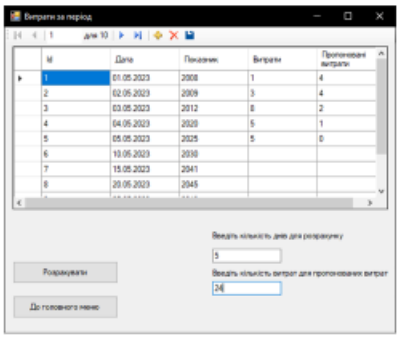


11

ЕКРАННІ ФОРМИ



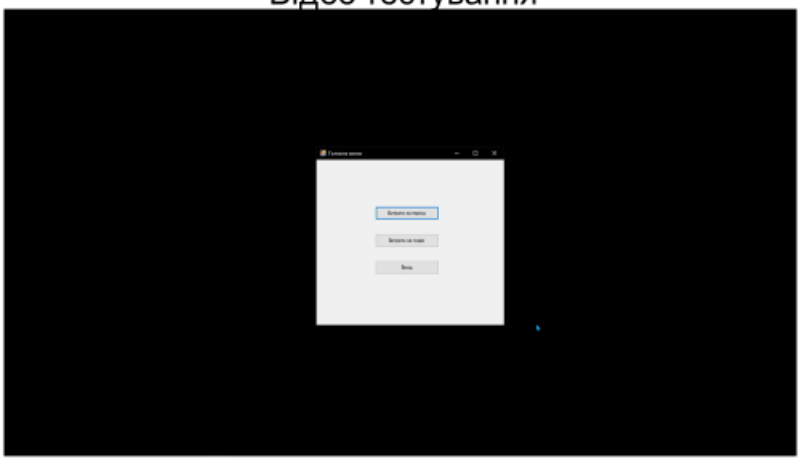
Головне меню



Витрати за період

12

Відео тестування



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1 Висоцький В.В. розробка програмного забезпечення щодо аналізу комунальних витрат з використанням uml діаграми станів/ Гаманюк І.М. В.В. Висоцький// застосування програмного забезпечення в інфокомунікаціних технологіях: матеріали всеукраїнської науково-технічної конференції. Збірник тез. 20 квітня 2023, ДУТ, м. Київ – К.: ДУТ, 2023 – С.7.

14

ВИСНОВКИ

1.Опрацьовано документ бачення системи з визначенням місця і ролі інформаційної системи. Складено контекстну діаграму.
2.Проаналізовано предметну галузь. Змодельовано модель предметної галузі. Відпрацьовано діаграму предметної галузі.
3.Визначено функціональні вимоги. Змодельовано модель прецедентів. Відпрацьовано діаграма прецедентів.
4.Визначено нефункціональні вимоги. Опрацьовано документ додаткова специфікація.
5.Здійснено проектування діяльності. Опрацьовано діаграми діяльності.
6.Здійснено проектування взаємодії об'єктів певних методів. Створено діаграми послідовності.
7.Розроблено проектування три-рівневої архітектури. Опрацьовано діаграму пакетів.
8.Здійснено проектування взаємодії об'єктів. Опрацьовано діаграму класів .
9.Змодульовано меню застосунку. Опрацьовано діаграму станів системи.
10.Розроблено проектування розгортання системи. Опрацьовано діаграму розгортання системи.
11.Визначено об'єм виконаної роботи. Опрацьовано діаграму артефактів.

15