

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАНУВАННЯ ВИТРАТ НА
ОРГАНІЗАЦІЮ ЗАХОДІВ
EVENT COSTS PLANNING SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-128
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Хаустов І.Р.

(прізвище та ініціали)

Керівник Голуб Т.В.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Хаустова Івана Романовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення планування витрат на організацію заходів
Event Costs Planning Software

керівник проєкту (роботи) Голуб Тетяна Василівна, к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Голуб Т.В., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання «11» квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Хаустов І.Р.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Голуб Т.В.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 80 с., 23 рис., 1 табл., 3 дод., 11 джерел.

ПЛАНУВАННЯ ВИТРАТ, ЗАХІД, СПІЛЬНА РОБОТА, РОЗДІЛЕНИЙ ДОСТУП, ГОЛОСУВАННЯ, ВЕБСИСТЕМА.

Об'єктом роботи є процес планування організації заходів.

Предметом роботи є програмне забезпечення планування витрат на організацію заходів.

Мета роботи – розробити програмне забезпечення планування витрат на організацію заходів для підтримки спільної роботи працівників над прийняттям спільних рішень і внесенням цих рішень.

Матеріали, методи та технічні засоби: мова програмування Python, фреймворк веброзробки Django, серверне програмне забезпечення для роботи з базами даних PostgreSQL.

Результати. Виконано повний цикл розробки програми, починаючи від вивчення предметної області з формулюванням вимог до програми на основі цього вивчення, охоплюючи проєктування і створення бази даних, визначення структури програми, реалізацію вебсистеми, базове тестування цієї системи.

Висновки. Створена програма може використовуватися для підтримки роботи над плануванням витрат при організації заходів, надаючи для цього не тільки можливості внесення планів і фактичних витрат, але і коментування, голосування, а також розділення доступу різних працівників до даних процесів, що розширює зручність її застосування для планування витрат за заходами.

Галузь використання. Організація заходів у частині планування витрат, включаючи зокрема весілля, конференції, масові зустрічі тощо.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 23 figures, 1 table, 3 appendixes, 11 sources.

COST PLANNING, EVENT, JOINT WORK, SHARED ACCESS, VOTING, WEBSITE.

The object of the work is the process of planning the organization of events.

The subject of the work is the cost planning software for the organization of events.

The purpose of the work is to develop software for planning the costs of organizing events to support the joint work of employees to make joint decisions and to implement these decisions.

Materials, methods and technical means: Python programming language, Django web development framework, server software for working with PostgreSQL databases.

Results. The full cycle of program development has been completed. It starts with the study of the subject area with the formulation of program requirements based on this study, covering the design and creation of a database, determining the structure of the program, and ends with the implementation of a web system and basic testing of this system.

Conclusions. The created program could be used to support work on cost planning in organizing events, providing not only the ability to make plans and actual costs, but also commenting, voting, and sharing access of different employees to these processes, which expands the convenience of its use for cost planning.

Field of use. Organization of events in terms of cost planning, including weddings, conferences, mass meetings and more.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	9
1.1 Особливості планування витрат на організацію заходів	9
1.2 Аналіз аналогів	11
1.3 Висновки за розділом 1	13
2 Розробка архітектури програми	14
2.1 Вибір засобів розробки програми	14
2.2 Функціональні вимоги до програмного забезпечення	15
2.3 Проектування бази даних	18
2.4 Висновки за розділом 2	29
3 Розробка програми	30
3.1 Порядок роботи користувача з програмою	30
3.2 Опис реалізації програми	31
3.3 Висновки за розділом 3	39
4 Експлуатація та тестування програми	40
4.1 Призначення програми	40
4.2 Умови виконання програми	40
4.3 Тестування і опис принципів експлуатації програми	40
4.4 Висновки за розділом 4	44
Висновки	46
Перелік джерел посилання	47
Додаток А Технічне завдання	48
Додаток Б Текст програми	52
Додаток В Слайди презентації	74

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Пандемія COVID-19, можливо, змусила сферу організації заходів зупинитися, але разом із цим навіть в такій ситуації відбулись зміни та були впроваджені інновації, що зрозуміло, адже за статистикою 60 % керівників вважають, що заходи є найважливішим маркетинговим каналом для досягнення бізнес-цілей компанії: Verified Market Research, провідна глобальна дослідницька та консалтингова фірма, очікує, що глобальна індустрія організації заходів принесе колосальні 2,194 трильйони доларів до 2028 року, що значно більше ніж 887 мільярдів доларів, які ринок отримав у 2020 році [1]-[3].

Звичайно, що заходи можуть носити різний характер, а це впливає і на те, який ефект вони виконують в результаті. Проте в будь-якому випадку важливо розуміти, що такий ефект за будь-якого типу заходів є, якщо захід організовано якісно. Так заходи для різних компаній можуть стати платформою для спілкування зі своєю аудиторією, незалежно від того, чи мета заходу – підвищити поінформованість про бренд за допомогою активації бренду та брендингу подій, чи створити значущий простір для спілкування, в результаті більшість маркетологів вважають, що заходи є важливим елементом успіху їхньої компанії [1]-[3].

Для підтримки сфери організації подій можуть використовуватися дуже різні програмні засоби, починаючи від найпростіших таблиць і документів, і завершуючи спеціалізованими рішеннями. Проте найпростіші варіанти роботи дозволяють створювати план витрат, вносити такі витрати, але ніяк не підтримують взаємодію між працівниками стосовно визначення конкретних витрат, сум цих витрат, що є важливим і нетривіальним. Тому метою роботи було розробити програмне забезпечення (ПЗ) планування витрат на організацію заходів для підтримки спільної роботи працівників над прийняттям спільних рішень і внесенням цих рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості планування витрат на організацію заходів

Бюджет заходу – це комплексна оцінка витрат і доходів: стандартний бюджет заходу може включати прогнозовані доходи та витрати заходу, а також фактичні доходи та витрати, важливо візуалізувати і записати ці значення та оцінити до початку заходу [4].

Планування заходів може бути складною справою, особливо коли йдеться про бюджет, адже достатньо багато потрібно розглянути та підготувати при його формуванні, тому кожному організатору заходів потрібен бюджет на планування, це допомагає підготуватися до кожного аспекту події та заощадити, а не витратити зайве, відповідно використовуючи шаблон бюджету на захід, можна детально спланувати та підготуватися до несподіваних моментів, загалом під час планування витрат потрібно враховувати зокрема наступні питання:

- подорожі та проживання: потрібно визначити, чи знадобиться надати житло всім, хто відвідає захід, а якщо так, то скільки це буде коштувати, це питання обов'язково потрібно помістити на перший план бюджету, щоб уникнути недооцінки фінансів, які знадобляться для успіху заходу;

- харчування: питання, скільки коштуватиме їжа, залежать від кількості людей, яких планується запросити на захід, а також від того, наскільки обширним буде меню, потрібно оцінити витрати відповідно до масштабу та типу заходу, який організується;

- компенсація персоналу: щоб уникнути фінансових труднощів, які можуть виникнути через поганий бюджет заходу, потрібно включити винагороду персоналу до статей плану заходу, керівники заходу або менеджери повинні визначити, скільки співробітників знадобиться для організації та сприяння успішному заходу, відповідно потрібно визначити, скільки часу буде витрачено і яка потрібна компенсація;

- витрати на об'єкт включають гроші, витрачені на оренду приміщення: потрібно скласти бюджет на основі ймовірних витрат, також потрібно врахувати приховані витрати, які можуть виникнути, наприклад, витрати, пов'язані з безпекою, освітленням або опаленням;

- логістика: потрібно врахувати, що можуть бути потрібні спеціальні контракти, дозволи чи страховка для проведення заходу, кожне з цих міркувань підпадає під логістику, важливий аспект будь-якого бюджету, можливість оцінити, скільки це буде коштувати, буде залежати в основному від місця проведення заходу;

- маркетинг – важливий аспект майже будь-якого заходу: він також може становити значний відсоток бюджету заходу, особливо коли організатори очікують отримати прибуток за рахунок вступних внесків або інших платежів, потрібно подумати про інструменти, які доведеться використовувати для реклами заходу, і зробити вибір на основі того, що найбільше відповідає бюджету [5].

Також окремо потрібно враховувати, що необхідно з часом покращувати бюджет: дослідження витрат, наприклад, щодо вибору місця проведення, може зайняти час, тому план повинен містити функції, які забезпечують гнучкість, це дозволить планувальникам покращувати бюджет з часом або коли вони отримають доступ до деталей, про які вони, можливо, не знали на початку процесу планування бюджету, окрім того потрібно передбачати в структурі план на випадок надзвичайних ситуацій, так включення плану на випадок надзвичайних ситуацій допомагає організатору заходів підготуватися до того, що щось може піти не так, фактично це письмове зобов'язання планувати несподівані події (цей план має передбачати, наприклад, що потрібно робити, якщо заходу на відкритому повітрі раптом загрожує несподівана гроза) [5].

1.2 Аналіз аналогів

Застосунок Pro Party Planner (рис. 1.1) [6]-[7] служить планувальником заходів: коли користувач знає, скільки гостей прийде, він створює список завдань, щоб підготувати все для відповідної події, окрім того надаються можливості встановлювати нагадування, після цього потрібно встановити бюджет і почати складати список покупок для події, в результаті можна буде відстежувати свій бюджет на їжу, прикраси та сувеніри для вечірок [7].



Рисунок 1.1 – Застосунок Pro Party Planner [7]

Цю програму в межах аналізу аналогів було виокремлено, оскільки вона є одним з представників програм відстежування витрат, які було застосовано для планування заходів. Проте варто зазначити, що фактично це інструмент для створення власного переліку справ. Це зручно для невеликих вечірок, коли важливим є перелік завдань, а бюджет всього процесу є не

дуже критичним. У випадку ж подій з великими вітрами і великою кількістю людей відповідно доводиться більше приділяти увагу плануванню і визначенню, скільки саме коштів можна витратити на певний захід, яким чином ці кошти будуть розподілені між статтями витрат. Окрім того з програмою тоді працює ціла група користувачів, представників організатора заходу. Тому для таких випадків подібні програми є не досить придатними.

Іншою групою застосунків, які можуть використовуватися для планування витрат для заходів є застосунки, які стосуються керування проєктами. Зокрема одним з найвідоміших таких застосунків є Trello (рис. 1.2) [8]-[9].

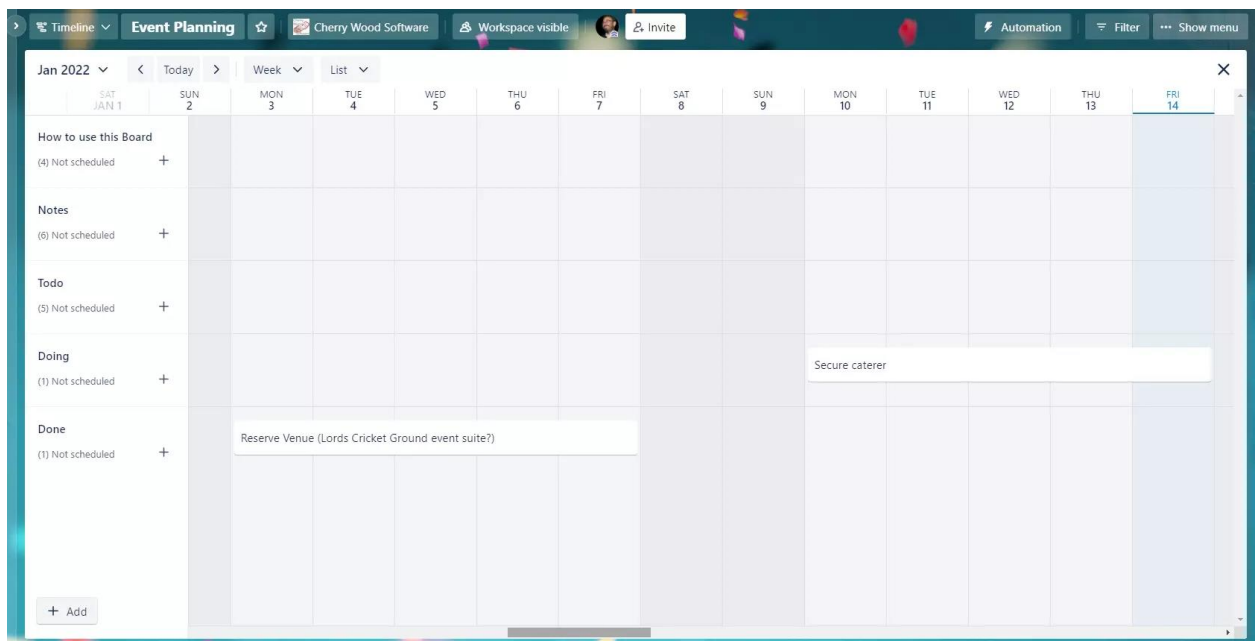


Рисунок 1.2 – Застосунок Trello [9]

Trello не є спеціалізованим рішенням для роботи з бюджетами подій, але логіка розподілу проєкту на завдання, визначення строків виконання цих завдань, визначення відповідальних може бути застосована для даної сфери також. Проте ця програма направлена саме на керування проєктом, а не на укладання бюджету, тому вона має багато різних засобів, які взагалі не будуть використовуватися в цій сфері, відповідно буде витрачатися зайвий час на навчання роботі з програмою, час на реалізацію потрібних дій.

Програма при цьому не враховує особливостей, пов'язаних з тим, що проекти є заходами, відповідно це специфічні варіанти подій, специфічні статті витрат, а без наявності набору готових статей стає складнішим створювати новий бюджет.

Тому важливою є розробка ПЗ, яке дозволить виконувати саме внесення витрат за заходами, планування цих витрат, внесення фактичних витрат за необхідності, буде надавати можливість зручного розділення витрат за групами, заходів за групами, насичуючи при цьому взаємодію працівників засобами комунікації, направленими на розв'язання проблем, які виникають під час цієї роботи. Окрім того потрібно розподіляти доступ різних користувачів до бюджетів різних заходів.

1.3 Висновки за розділом 1

У розділі виконано огляд особливостей планування витрат на організацію заходів, розглянуто аналоги програми, яка розробляється.

Для виконання розробки програми було визначено наступні завдання в роботі:

- визначення вимог до розробки;
- проєктування бази даних;
- проєктування ПЗ планування витрат на організацію заходів;
- програмна реалізація вебзастосунку планування витрат на організацію заходів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів розробки програми

Під час вибору засобів проєктування та подальшої розробки програми спочатку було прийнято рішення (табл. 2.1) про вибір підходу серед веброзробки на основі фреймворків та стандартної веброзробки на основі фактично засобів мови програмування.

Таблиця 2.1 – Вибір підходу до веброзробки

Критерій	Веброзробка на основі фреймворків	Стандартна веброзробка
Повторне використання коду	Активне	Доволі низьке, але можливе за рахунок виділення функцій і бібліотек
Гнучкість до змін	Висока	Низька
Час розробки	Менший, але потребує часу на вивчення фреймворку	Більший
Наявні вбудовані засоби	Підключення бібліотек мови програмування, зазвичай мають засоби адміністрування, шаблонізації, спрощення роботи з даними, але залежить від фреймворку	Можливе підключення бібліотек у відповідність з мовою програмування

Підхід на основі використання вебфреймворку дозволяє зменшити ресурси, необхідні на реалізацію програми, тому було обрано його. А в якості

самого фреймворку було визначено використовувати Django [10], а мовою програмування – відповідно Python.

2.2 Функціональні вимоги до програмного забезпечення

Вимоги до ПЗ було розподілено за ролями користувачів, а серед користувачів визначено працівника, учасника заходу та адміністратора.

Функціональні вимоги до ПЗ зі сторони учасника заходу (рис. 2.1):

- відображення бюджету заходу;
- відображення розподілу бюджету між типами і групами витрат;
- відображення бюджетів заходів за тегом;
- пошук серед бюджетів заходів;
- відображення доступних бюджетів заходів за майданчиком проведення;
- відображення бюджетів заходів за власною участю.



Рисунок 2.1 – Діаграма прецедентів учасника заходу

Функціональні вимоги до ПЗ зі сторони працівника служби організації заходів (рис. 2.2):

- створення бюджету заходу;
- редагування бюджету заходу;
- створення бюджету заходу на основі шаблону;
- відображення бюджету заходу;
- відображення розподілу бюджету між типами і групами витрат;
- відображення бюджетів заходів, відкритих для групи;
- додавання статей витрат бюджету заходу;
- редагування статей витрат бюджету заходу;
- внесення витрат за статтями бюджету;
- створення груп працівників;
- керування доступом працівників у групі;
- визначення доступу групи працівників до бюджету;
- проведення голосування за статтею витрат;
- коментування статті витрат;
- відображення повної інформації статті витрат в бюджеті заходу;
- визначення бюджету в якості шаблону;
- створення порівняння бюджетів заходів;
- відображення порівнянь бюджетів групою;
- відображення нових коментарів за всіма статтями з бюджету;
- відображення всіх голосувань за бюджетом заходу;
- визначення тегів бюджету заходу;
- відображення бюджетів заходів за тегом;
- пошук серед бюджетів заходів;
- відображення бюджетів заходів за майданчиком проведення;
- визначення учасників заходу.

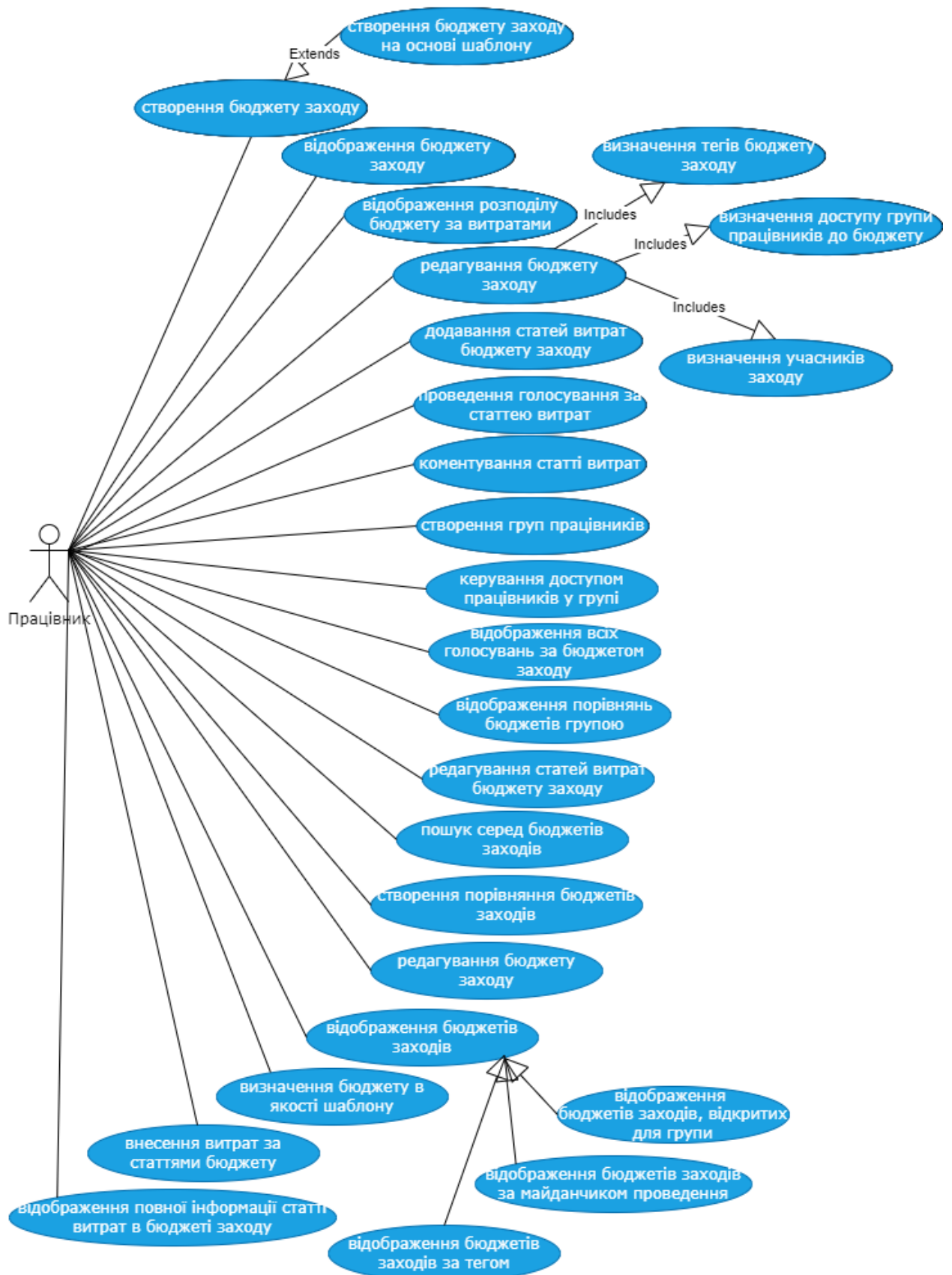


Рисунок 2.2 – Діаграма прецедентів працівника

Функціональні вимоги до ПЗ зі сторони адміністратора (рис. 2.3):

- додавання, відображення і редагування типів витрат;
- додавання, відображення і редагування груп витрат;
- додавання, відображення і редагування майданчиків
- додавання, відображення і редагування типів заходів.



Рисунок 2.3 – Діаграма прецедентів адміністратора

2.3 Проектування бази даних

Структура бази даних (БД) ґрунтувалась на послідовності прийнятих рішень, що безпосередньо на неї впливали: вибір системою управління БД PostgreSQL, сама предметна область за результатами аналізу, який був виконаний.

Для збереження типів заходів, адже передбачається, що програма буде працювати з різними їх різновидами (весілля, вечірки, конференції тощо), створено таблицю EventType зі складом полів:

- id – номер типу події;
- type_name – назва типу події в програмі.

Збереження самого заходу і відповідно бюджету цього заходу, адже це його основна характеристика в програмі, створено таблицю EventBudget зі складом полів:

- id – номер заходу;

– eventtype_id – тип заходу, до якого відноситься цей захід, визначає зовнішній ключ до таблиці EventType, реалізуючи через нього зв'язок один до багатьох: кожен захід відноситься до одного типу заходів, а один тип заходів має різні конкретні заходи, які стосуються цього типу;

– event_title – назва заходу;

– plan_start – запланований початок заходу;

– plan_end – заплановане завершення заходу;

– event_information – повна інформація про захід, тобто те, яким чином він описується для його учасників;

– planned_budget – запланований бюджет заходу;

– actual_budget – фактичний бюджет заходу;

– notation – особливості заходу, тобто інформація про захід, яка використовується працівника, відповідальними за планування, організацію заходу (не публічна інформація на відміну від event_information);

– participants_quantity – кількість учасників, що беруть участь в цьому заході;

– venue_id – місце проведення заходу, визначає зовнішній ключ до таблиці Venue, реалізуючи через нього зв'язок один до багатьох: заходи розглядаються як такі, що проводяться тільки на одному майданчику одночасно, а кожен майданчик може містити багато заходів, які в різний час, а можливо і одночасно проводяться на цьому майданчику;

– staffgroup_id – група працівників, відповідальних за захід, якій доступна можливість роботи з бюджетом заходу, визначає зовнішній ключ до таблиці StaffGroup, реалізуючи через нього зв'язок один до багатьох: одна подія може керуватися однією групою працівників, а ця група в свою чергу може керувати багатьма подіями в різний час;

– status – стан планування бюджету заходу: це поле необхідне для того, щоб визначити, коли роботу над плануванням бюджету ще не розпочато, коли вже розпочато, коли завершено, коли відкрито повторно;

– opened – публічність доступу до бюджету: деякі бюджети можуть відкриватися всім користувачам, деякі тільки учасникам і працівникам, відповідальним за організацію заходу;

– creator – працівник, який створив цей бюджет і відповідно йому належить основна керуюча роль при визначенні засобів доступу до цього бюджету, визначає зовнішній ключ до таблиці Staff, реалізуючи через нього зв'язок один до багатьох: захід може бути створений персонально одним працівником, а цей працівник може в різний час створити велику кількість інших заходів.

Шаблон заходу необхідний для того, щоб при створенні нових бюджетів одразу мати під рукою набір даних, що характеризує раніше виконаний бюджет, для уникнення зайвої роботи. Для збереження шаблонів заходів створено таблицю EventTemplate зі складом полів:

- id – номер шаблону заходу;
- staff_id – працівник, який визначив цей шаблон для себе і зможе його використовувати в переліку шаблонів при створенні відповідного нового бюджету, визначає зовнішній ключ до таблиці Staff, реалізуючи через нього зв'язок один до багатьох;
- eventbudget_id – захід, бюджет якої розглядається в основі даного шаблону, визначає зовнішній ключ до таблиці EventBudget, реалізуючи через нього зв'язок один до багатьох.

Шаблон заходу є реалізацією зв'язку багато до багатьох між працівниками і бюджетами заходів, що пов'язано з тим, що кожен працівник може зберігати декілька бюджетів заходів в якості власних шаблонів, а кожен бюджет заходу відповідно може використовуватися різними іншими працівниками для створення шаблонів для власної роботи при створенні нових бюджетів заходів.

Для збереження типів статей витрат, тобто узагальненого розділу статей витрат (транспорт, проживання), створено таблицю CostType зі складом полів:

- id – номер типу статті витрат;
- type_name – назва типу статті витрат;
- short_information – коротка інформація про тип статті витрат, що може використовуватися як підказка.

Але таке розділення потребує уточнення, щоб при створенні нового бюджету мати певний орієнтир серед вже можливий більш конкретних статей витрат у складі бюджету. Тобто потрібний стандартний набір статей в межах типу статей, який в подальшому буде уточнюватися в бюджеті, але з яким будуть ці уточнення співвідноситися, що необхідно потім для сприйняття цього бюджету загалом. Для збереження цих даних створено таблицю CostGroup зі складом полів:

- id – номер групи статей витрат;
- group_name – назва групи статті (фактично це назва статті, що використовується шаблонно при створенні бюджетів та для визначення того, що мав на увазі працівник, визначаючи деяку статтю витрат у складі бюджету заходу);
- short_information – коротка інформація про групу статей витрат, що може використовуватися як підказка;
- costtype_id – тип статей витрат, до якого належить ця група статей: визначає зовнішній ключ до таблиці CostType, реалізуючи через нього зв'язок один до багатьох: один тип статей витрат може мати багато груп, а кожна група відноситься тільки до одного загального типу статей витрат.

Самі конкретні витрати в складі бюджету заходу зберігаються в програмі через таблицю Cost зі складом полів:

- id – номер статті витрат;
- costgroup_id – група статей витрат, з якою співвіднесено цю статтю витрат з бюджету, визначає зовнішній ключ до таблиці CostGroup, реалізуючи через нього зв'язок один до багатьох: кожна витрата належить до деякої групи статей витрат, а кожна група статей може включати багато різних конкретних статей витрат;

- short_name – коротка назва статті витрат;
- long_name – повна назва статті витрат;
- planned_budget – запланований бюджет за цією статтею витрат;
- actual_budget – фактичний бюджет за цією статтею витрат;
- notation – тлумачення статті витрат за необхідності.

Керування бюджетом заходу здійснюють його організатори, яких в системі презентують працівники. Для збереження цих даних створено таблицю Staff зі складом полів:

- id – номер працівника;
- username – логін працівника в програмі;
- password – пароль працівника в програмі;
- first_name – ім'я працівника або ім'я та по батькові, якщо використовується повна форма;
- last_name – прізвище працівника;
- contacts – перелік контактів, які використовуються працівником.

Однак для зручної роботи з бюджетами з працівників потрібно створювати групи, щоб вони могли взаємодіяти всі разом і зручно було потім задіювати такі групи на інших заходах. Для збереження даних груп працівників створено таблицю StaffGroup зі складом полів:

- id – номер групи працівників;
- name – назва групи працівників, що використовується для зручного звернення і пошуку при створенні бюджетів у подальшому.

Група сама по собі, за своєю назвою, не існує, потрібно визначити її учасників. Для збереження цих даних створено таблицю StaffGroupParticipant зі складом полів:

- id – номер запису про працівника як учасника групи;
- staffgroup_id – номер групи працівників, до якої належить даний працівник, визначає зовнішній ключ до таблиці StaffGroup, реалізуючи через нього зв'язок один до багатьох;

- `staff_id` – працівник, який бере участь у цій групі, визначає зовнішній ключ до таблиці `Staff`, реалізуючи через нього зв'язок один до багатьох;

- `vote_right` – визначає, чи має цей працівник у цій групі право голосувати за особливості бюджету: може знадобитися, наприклад, щоб відділити працівників, які беруть участь саме в реалізації заходу, від працівників, які саме планують бюджет.

Таблиця `StaffGroupParticipant` визначає право голосу працівників у групі, але за своєю суттю є реалізацією зв'язку багато до багатьох між працівниками та групами працівників: один працівник може входити в різні групи працівників, а кожна група для того і утворюється, щоби містити набір працівників.

Під час роботи над статтями витрат у складі бюджету не завжди позиції різних сторін сходяться, група може розділятися, тоді важливо, щоб кожен учасник групи висловив свою позицію, а в результаті підвести підсумок і прийняти загальне рішення. Саме для цього в системі потрібні голосування. Для збереження даних голосувань за статтями витрат створено таблицю `CostVote` зі складом полів:

- `id` – номер голосування за статтю витрат;
- `cost_id` – стаття витрат бюджету заходу, голосування за яку відбувається, визначає зовнішній ключ до таблиці `Cost`, реалізуючи через нього зв'язок один до багатьох: одна стаття бюджету заходу потенційно може мати декілька голосувань протягом її існування, але кожне голосування стосується однієї статті;
- `vote_title` – назва голосування або це фактично може бути запитання, винесене на голосування;
- `vote_start` – час початку голосування;
- `vote_end` – час завершення голосування;
- `percent` – відсоток голосів, які потрібно отримати для завершення голосування успішно;

– `costvote_position_id` – варіант голосування, який переміг у голосуванні, визначає зовнішній ключ до таблиці `CostVotePosition`, реалізуючи через нього зв'язок один до одного: у голосуванні може бути тільки один переможний варіант (якщо він взагалі визначений).

Кожне голосування складається з декількох варіантів для вибору, для збереження яких створено таблицю `CostVotePosition` зі складом полів:

- `id` – номер варіанту голосування;
- `costvote_id` – голосування, до якого стосується цей варіант, визначає зовнішній ключ до таблиці `CostVote`, реалізуючи через нього зв'язок один до багатьох: кожен варіант в голосування відноситься тільки до одного конкретного голосування і не може використовуватися в різних голосуваннях, а кожне голосування складається з деякого набору варіантів для вибору;
- `position_text` – текст цього варіанту, запропонованого для голосування;
- `votes_quantity` – кількість голосів, яку було віддано за цей варіант в голосуванні.

Самі результати голосування працівників зберігаються окремо для того, щоб відновити і перевірити, чи голосував вже конкретний працівник, чи ще ні. Для збереження цих даних створено таблицю `StaffCostVote` зі складом полів:

- `id` – номер голосу, відданого працівником у деякому голосуванні;
- `costvote_position_id` – варіант, якому працівник віддав перевагу;
- `staff_id` – працівник, який голосував.

Таблиця `StaffCostVote` визначає голосування працівників за статтями витрат, за своєю суттю є реалізацією зв'язку багато до багатьох між працівниками та варіантами вибору при голосуванні: один працівник може віддавати свої голоси в різний час за різні варіанти з різних голосувань, а кожен варіант з одного голосування може мати багато відданих за нього голосів різних працівників.

Обговорення статей витрат в бюджеті заходу реалізується через коментарі працівників. Для збереження коментарів працівників створено таблицю CostComment зі складом полів:

- id – номер коментаря працівника про статтю витрат у бюджеті заходу;
- cost_id – номер статті витрат з бюджету заходу, визначає зовнішній ключ до таблиці Cost, реалізуючи через нього зв'язок один до багатьох: один коментар стосується тільки однієї статті витрат в бюджеті заходу, але кожна стаття витрат може мати багато коментарів;
- staff_id – працівник, який створив коментар, визначає зовнішній ключ до таблиці Staff, реалізуючи через нього зв'язок один до багатьох: один працівник може мати багато коментарів, але кожен коментар має автором тільки одного працівника;
- comment_text – текст коментаря;
- created – дата створення коментаря.

Працівники можуть виконувати порівняння бюджетів, при чому зберігати такі порівняння. Для збереження порівнянь створено таблицю EventBudgetComparison зі складом полів:

- id – номер порівняння;
- eventbudget1_id – бюджет заходу, який бере участь в порівнянні, визначає зовнішній ключ до таблиці EventBudget, реалізуючи через нього зв'язок один до багатьох: першим у порівняння може бути вказано бюджет тільки одного заходу, з іншого боку кожен захід може використовуватися в різних порівняннях в якості першої позиції;
- eventbudget2_id – бюджет іншого заходу, що бере участь у порівнянні, визначає зовнішній ключ до таблиці EventBudget, реалізуючи через нього зв'язок один до багатьох: другим у порівняння може бути вказано бюджет тільки одного заходу, з іншого боку кожен захід може використовуватися в різних порівняннях в якості другої позиції;

– `staff_id` – працівник, який сформулював для себе таке порівняння, визначає зовнішній ключ до таблиці `Staff`, реалізуючи через нього зв'язок один до багатьох: один працівник може мати багато порівнянь, для кожного порівняння має бути вказано тільки один власник;

– `staffgroup_id` – група, яка може працювати з цим порівнянням (для випадків, коли працівник хоче продемонструвати свої спостереження іншим), визначає зовнішній ключ до таблиці `StaffGroup`, реалізуючи через нього зв'язок один до багатьох: одне порівняння може стосуватися тільки однієї групи працівників, але кожна група працівників може мати багато порівнянь різних заходів;

Для збереження майданчика або приміщення, де проводиться захід, створено таблицю `Venue` зі складом полів:

- `id` – номер майданчика;
- `title` – назва майданчика;
- `situated` – адреса, за якою знаходиться цей майданчик;
- `event_level` – рівень проведення подій на цьому майданчику;
- `information` – опис цього майданчика.

Учасники подій мають можливість переглядати дані деяких бюджетів подорожей, а для збереження даних самих учасників, облікових записів цих учасників, створено таблицю `Participant` зі складом полів:

- `id` – номер учасника;
- `username` – логін учасника в програмі;
- `password` – пароль учасника в програмі;
- `person_information` – персональна інформація про учасника.

Для збереження того, з якими невідкритими бюджетами можуть працювати учасники, створено таблицю `EventParticipant` зі складом полів:

- `id` – номер запису про доступ учасника до заходу;
- `eventbudget_id` – захід, доступ до якого має учасник, визначає зовнішній ключ до таблиці `EventBudget`, реалізуючи через нього зв'язок один до багатьох;

– participant_id – учасник, який має доступ до цього заходу, визначає зовнішній ключ до таблиці Participant, реалізуючи через нього зв'язок один до багатьох.

Таблиця EventParticipant визначає доступ учасників до даних про бюджет заходів, реалізуючи зв'язок багато до багатьох: один учасник може брати участь в різних заходах, а кожен захід може мати багато учасників.

Для збереження даних про перегляд коментарів різних працівників щодо статей витрат, що необхідно, щоб не заплутатися в створених коментарях за різними статтями, створено таблицю StaffCostComment зі складом полів:

- id – номер запису про перегляд коментаря;
- cost_comment – коментар про статтю витрат, який було переглянуто працівником, визначає зовнішній ключ до таблиці CostComment, реалізуючи через нього зв'язок один до багатьох;
- staff_id – працівник, який переглянув коментар, визначає зовнішній ключ до таблиці Staff, реалізуючи через нього зв'язок один до багатьох.

Таблиця StaffCostComment визначає перегляд коментарів працівника і реалізує зв'язок багато до багатьох між таблицями Staff і CostComment: один працівник може переглядати багато коментарів про різні статті витрат в різних бюджетах, а кожна стаття витрат може переглядатися різними працівниками.

Для того щоб спростити пошук відкритих бюджетів заходів зокрема, а також спростити перехід між різними бюджетами, визначено сутність тегу, для збереження даних якого створено таблицю Tag зі складом полів:

- id – номер тегу;
- name – назва тегу.

Описана структура БД приведена на рис. 2.4.

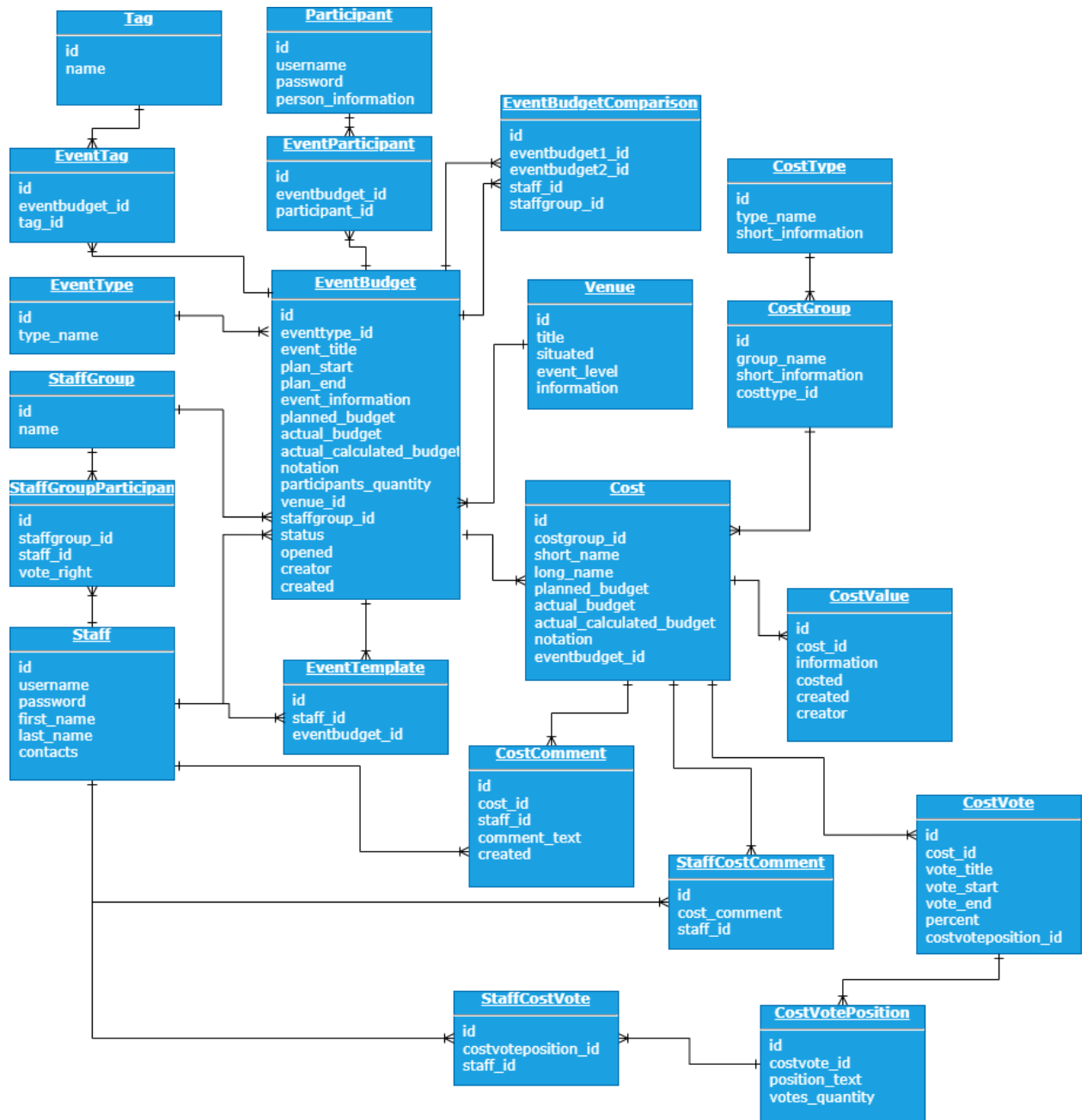


Рисунок 2.4 – Схема БД

Оскільки теги використовуються для відзначення бюджетів заходів, то зв'язок між заходами і тегами зберігається в таблиці EventTag, що реалізує зв'язок багато до багатьох, адже кожен тег має використовуватися в різних бюджетах, а кожен бюджет може мати багато тегів, відповідно таблиця створена зі складом полів:

- id – номер запису про тег в заході;

- eventbudget_id – бюджет заходу, якого стосується тег, визначає зовнішній ключ до таблиці EventBudget, реалізуючи через нього зв'язок один до багатьох;

- tag_id – тег, який визначений для цього бюджету заходу, визначає зовнішній ключ до таблиці Tag, реалізуючи через нього зв'язок один до багатьох.

Для збереження даних про витрату за статтею в бюджеті створено таблицю CostValue зі складом полів:

- id – номер витрати за статтею;
- cost_id – стаття витрат, до якої належить ця витрата, визначає зовнішній ключ до таблиці CostComment, реалізуючи через нього зв'язок один до багатьох: одна стаття витрат може мати багато витрат за нею, але одна витрата стосується однієї конкретної статті;
- information – опис даної витрати;
- costed – витрачена сума;
- created – дата створення запису про витрату;
- creator – працівник, який створив запис, визначає зовнішній ключ до таблиці Staff, реалізуючи через нього зв'язок один до багатьох: один працівник може мати багато витрат за статтями, а кожна витрата створена може бути одним працівником.

2.4 Висновки за розділом 2

Порівняно варіанти реалізації вебсистеми на основі фреймворків та без них, обрано підхід на основі використання фреймворків, а для розробки використано фреймворк Django, виконано моделювання сценаріїв роботи працівника, учасника заходу та адміністратора з програмою, описано створену структуру БД.

3 РОЗРОБКА ПРОГРАМИ

3.1 Порядок роботи користувача з програмою

Порядок роботи користувача з програмою заснований на меню програми. Меню учасника є достатньо обмеженим варіантом меню працівника, тому саме його послідовність роботи було розглянуто в даному випадку (рис. 3.1). Порядок роботи з адміністратором визначається тільки набором моделей даних, які доступні йому для перегляду і змінювання, тобто фактично структурою доступної частини БД. Працівник має розпочати роботу зі своїм інтерфейсом з авторизації, після цього йому буде доступний вибір дій з меню і подальший вибір знову з того ж набору дій.

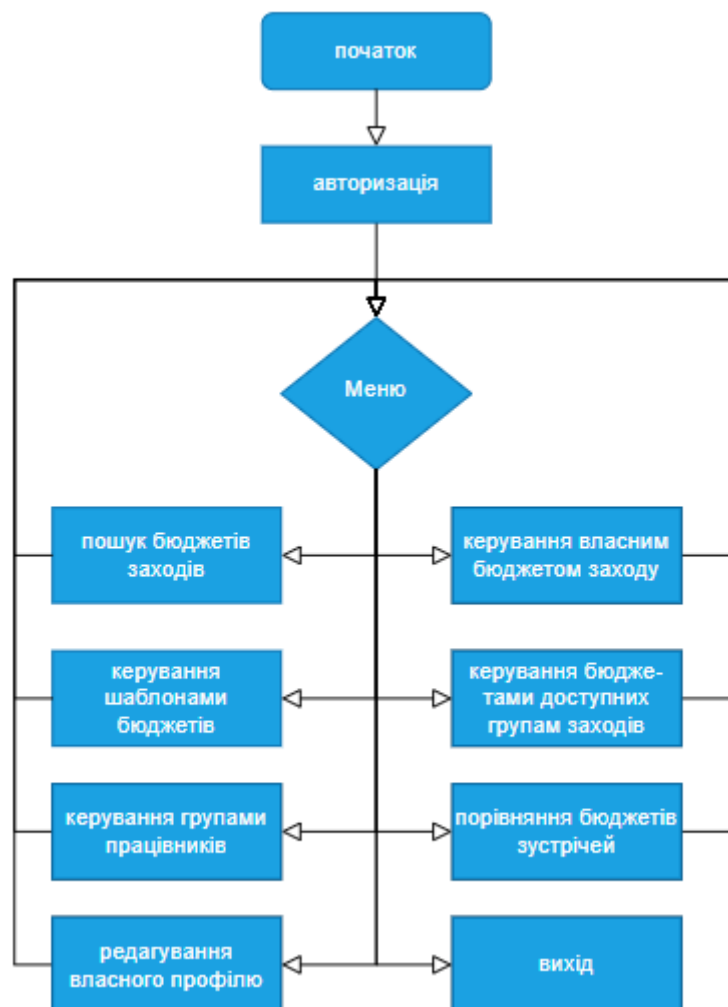


Рисунок 3.1 – Функціональна схема програми для працівника

3.2 Опис реалізації програми

Програму реалізовано через застосування вебфреймворку, яким є Django. Розділення коду виконано за відповідною логікою, що складається з моделей, представлень і відповідних ним шаблонів.

Класи моделей визначені за структурою таблиць БД. Для моделей, пов'язаних з сутностями статей витрат бюджету та бюджету заходу загалом, було створено методи `save`, які забезпечили переобчислення сумарних витрат в залежності від значень, які задавалися в витратах за статтями, а також в сумарних витратах, зафіксованих працівниками за статтями.

У свою чергу класи представлень мають зовсім іншу структуру і забезпечують наповнення даними для функцій програми. Відповідні шаблони побудовані з використанням шаблону [11].

Створення самого бюджету заходу реалізовано через клас `EventBudgetCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує спочатку перевірку того, що авторизований користувач є працівником, і якщо це так, то вносить дані про тип заходу, про майданчик, на якому відбувається захід, про групу працівників, якій доступна робота з бюджетом цього заходу, про автора цього бюджету, яким одразу стає авторизований працівник, та про дату створення запису про бюджет, а усі інші поля моделі бюджету заповнюються автоматично шляхом перенесення значень з полів форми.

Створення самого бюджету заходу на основі раніше зафіксованого шаблону реалізовано через клас `EventBudgetTemplateCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії,

та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує реалізацію шаблонізованого створення бюджету. Це фактично створення бюджету на основі прикладу. Спочатку перевіряється, чи авторизований користувач є працівником, а якщо це так, то обирається бюджет заходу, який стосується шаблону, який вказано в параметрі `template`. Якщо тип заходу не вказано в якості відповідного параметра, то значення береться з шаблону. Якщо майданчик проведення заходу не вказано в якості відповідного параметра, то значення береться з шаблону. Якщо групу працівників, яка може працювати з бюджетом не вказано через відповідний параметр, то значення береться з шаблону. Також заповнюються дані про автора бюджету та дату його внесення. Після цього реалізується заповнення статей витрат цього бюджету. Воно реалізується у тому випадку, якщо в якості значення параметра `repeat` вказано яке-небудь значення. Якщо ж значення пропущено, то ці дії не виконуються. Дії реалізуються наступним чином. Вибираються всі витрати, які стосуються бюджету заходу, який задано в шаблоні. Далі за кожною витратою вноситься відповідна аналогічна витрата для бюджету, який створюється. Автоматично для кожної такої витрати переноситься значення групи витрат, короткого імені, довгого імені та запланованого сумарного обсягу бюджету, а всі інші значення залишаються порожніми:

```
if self.kwargs['repeat']:
    costs = event_template.costs
    for cost in costs:
        Cost.objects.create(eventbudget_id = self.object,
                             costgroup_id = cost.costgroup_id, short_name = cost.short_name,
                             long_name = cost.long_name, planned_budget = cost.planned_budget)
```

Редагування бюджету заходу реалізовано через клас `EventBudgetUpdateView`, який було створено наслідуванням класу авторизації

користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення редагування `UpdateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує спочатку перевірку, чи є авторизований користувач працівником, який є автором бюджету подорожі, і в тому випадку, якщо це так, то забезпечується можлива зміна типу події, якщо вона була перевизначена в параметрах, майданчика проведення події, групи працівників, які можуть працювати зі статтями цього бюджету, інші поля змінюються автоматично на основі значень полів форми, якщо вони змінювались.

Відображення повної інформації про бюджет заходу реалізовано через клас `EventBudgetDetailView`, який було створено наслідуванням класу представлення деталізації `DetailView`. Центральним методом, який підтримує роботу цього класу є метод формування контексту даних `get_context_data(self, **kwargs)`, який відбирає в якості даних тільки інформацію про бюджет цього заходу загалом, а також додає до цієї інформації перелік статей витрат, що входять до бюджету цього заходу, а також кількість коментарів до статей витрат, які ще не були побачені авторизованим у даний момент працівником. Останнє визначається наступним чином:

```
auth_user = self.request.user
staff = Staff.objects.get(id = auth_user.id)
comments =
CostComment.objects.filter(cost_id__eventbudget_id__id =
self.object.pk)
notseen_comments = comments.exclude(staffs__contains = staff)
ebdv_contextdata['notseen_comments'] = notseen_comments.count()
return ebdv_contextdata
```

Відображення переліку бюджетів, автором яких є авторизований працівник, реалізовано через клас `CreatedEventBudgetsView`, який було

створено наслідуванням класу представлення роботи з переліком даних ListView. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає всі бюджети заходів, у полі автора яких вказано авторизованого працівника.

Відображення переліку бюджетів, які доступні для всіх груп працівників, до яких входить авторизований працівник, реалізовано через клас `GroupsEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних ListView. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає спочатку всі групи, у яких бере участь авторизований працівник, а далі створює перелік з бюджетів заходів, які відкриті цим групам:

```
sgroups = StaffGroupParticipant.objects.filter(staff_id = staff)
groups = []
for sgroup in sgroups:
    if sgroup.staff_id not in groups:
        groups.append(sgroup.staff_id)
budgets = EventBudget.objects.filter(staffgroup_id__in = groups)
budgets = budgets.order_by('-created')
```

Відображення переліку бюджетів за заданою групою працівників реалізовано через клас `GroupEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних ListView. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає всі бюджети заходів, у полі доступу групи яких вказано значення, що відповідає значенню параметра `group`.

Відображення переліку бюджетів, які доступні для авторизованого учасника, реалізовано через клас `ParticipantEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних ListView. Центральним методом, який підтримує роботу цього класу є метод

заповнення результатів запиту `get_queryset(self)`, який визначає спочатку всі записи про участь цього учасника в заходах, а далі створює перелік з бюджетів заходів, які відповідають цим учасням.

Відображення переліку бюджетів, які відзначені заданим тегом, реалізовано через клас `TagEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних `ListView`. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає спочатку значення параметра `tag`, далі за цим текстовим значенням встановлює значення об'єкта тега, а тоді перевіряє, ким є авторизований користувач: учасником або працівником. В залежності від цього формує відповідний перелік бюджетів аналогічно способам, описаним вище у відповідних представленнях, а тоді відбирає з цих бюджетів ті, які були відзначені заданим тегом.

Відповідно перевірка того, чи є авторизований користувач учасником, а тоді заповнення і відбір бюджетів реалізується наступним чином:

```

participant = Participant.objects.get(id = auth_user.id)
if participant:
    pevents = EventParticipant.objects.filter(participant_id =
participant)
    events = []
    for event in pevents:
        if event.eventbudget_id not in events:
            events.append(event.eventbudget_id)
    events = events.order_by('-created')
    events = events.filter(tags__contains = tag)
    return events

```

Відображення переліку бюджетів, які відбувались на заданому майданчику, реалізовано через клас `VenueEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних

ListView. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає спочатку значення параметра `venue`, тобто номер майданчика, а далі перевіряє, ким є авторизований користувач: учасником або працівником. В залежності від цього формує відповідний перелік бюджетів, а тоді відбирає з цих бюджетів ті, які відбувались на заданому майданчику.

Пошук бюджетів реалізовано через клас `FindEventBudgetsView`, який було створено наслідуванням класу представлення роботи з переліком даних `ListView`. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає спочатку значення параметрів пошуку, що включають тип заходу, назву або частину назви заходу, максимальні витрати бюджету, мінімальні витрати, інтервал, у який має вкластися реалізація заходу, майданчик проведення заходу та набір тегів. В залежності від ролі користувача в програмі формується відповідний перелік бюджетів, а тоді відбираються з цих бюджетів ті, які відповідають встановленим значенням параметрів пошуку. Перелік тегів перевіряється таким чином, що якщо бюджет містить хоч один такий тег, то він залишається в переліку бюджетів заходів.

Додавання статті витрат до бюджету заходу реалізовано через клас `CostCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує спочатку перевірку того, що авторизований користувач є працівником, і має доступ до цього бюджету заходу, і якщо це так, то вносить дані про бюджет, до якого ця стаття належить, про групу витрат, а усі інші поля статті витрат бюджету заповнюються автоматично шляхом перенесення значень з полів форми.

Редагування статті витрат з бюджету заходу реалізовано через клас `CostUpdateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення редагування `UpdateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує спочатку перевірку того, що авторизований користувач є працівником, і має доступ до цього бюджету заходу, і в тому випадку, якщо це так, то забезпечується можлива зміна групи статті витрат, інші поля змінюються автоматично на основі значень полів форми, якщо вони змінювались.

Додавання витрат до статті бюджету заходу реалізовано через клас `CostValueCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Фактично заповнення значень запису про витрати реалізується автоматично через відповідні поля форми.

Редагування витрат до статті бюджету заходу реалізовано через клас `CostValueUpdateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення редагування `UpdateView`. Фактично заповнення оновлення запису про витрати реалізується автоматично через відповідні поля форми, якщо воно було виконано.

Додавання коментаря до статті витрат реалізовано через клас `CostCommentCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`,

який забезпечує спочатку перевірку того, що авторизований користувач є працівником, і має доступ до цього бюджету заходу, і якщо це так, то вносить дані про статтю витрат, до якої належить коментар, автоматично заповнюється дата створення коментаря та працівник, який є автором коментаря, текст коментаря заповнюється автоматично шляхом перенесення цього значення з поля форми.

Відображення повної інформації про статтю витрат бюджет заходу реалізовано через клас `CostDetailView`, який було створено наслідуванням класу представлення деталізації `DetailView`. Центральним методом, який підтримує роботу цього класу є метод формування контексту даних `get_context_data(self, **kwargs)`, який відбирає в якості даних тільки інформацію про цю статтю витрат загалом, а також додає до цієї інформації перелік витрат, що відносяться до цієї статті, перелік голосувань, які стосуються цієї статті витрат, також формується перелік переглянутих і непереглянутих цим працівником коментарів до цієї статті витрат.

Додавання бюджету заходу до шаблону реалізовано через клас `EventTemplateCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує внесення даних про автора шаблону як авторизованого користувача, про бюджет заходу, який є основою шаблону.

Відображення переліку бюджетів, які були створені авторизованим працівником, реалізовано через клас `TemplatesView`, який було створено наслідуванням класу представлення роботи з переліком даних `ListView`. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає спочатку авторизованого користувача як працівника, а тоді визначає шаблони, які були створені цим працівником.

Додавання бюджетів до порівняння реалізовано через клас `EventBudgetComparisonCreateView`, який було створено наслідуванням класу авторизації користувача `LoginRequiredMixin`, що визначає необхідність попередньої авторизації перед виконанням цієї дії, та класу представлення створення `CreateView`. Центральним методом, який підтримує роботу цього класу є метод підтримки роботи відповідної форми `form_valid(self, form)`, який забезпечує внесення даних про перший і другий бюджети та про групу працівників, якій доступно це порівняння.

Відображення переліку порівнянь, створених авторизованим працівником, реалізовано через клас `MyEventBudgetComparisonsView`, який було створено наслідуванням класу представлення роботи з переліком даних `ListView`. Центральним методом, який підтримує роботу цього класу є метод заповнення результатів запиту `get_queryset(self)`, який визначає всі порівняння, які були створені авторизованим користувачем, якщо він є працівником.

3.3 Висновки за розділом 3

Виконано опис порядку роботи працівника з програмою з представленням функціональної схеми програми. Представлено основні розроблені класи програми.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для надання працівникам служб з організації заходів засобів планування бюджету цих заходів, командної роботи над цими бюджетами, винесення спільних рішень щодо статей витрат в бюджеті заходу, аналізу існуючих бюджетів, а учасникам подій – надання можливостей ознайомлення з витратами за їхніми заходами.

4.2 Умови виконання програми

Технічні засоби в мінімальній конфігурації, достатній для роботи програми, охоплюють 1 Гб пам'яті жорсткого диску, 4 Гб оперативної пам'яті, 4 ядра процесору (частота 2,8 ГГц).

4.3 Тестування і опис принципів експлуатації програми

Для перевірки успішності роботи програми було виконано її тестування за сценаріями роботи користувачів, визначених діаграмами прецедентів. Відповідність встановлювалась у тому випадку, якщо отриманий результат при виконанні сценарію відповідав очікуваному. Так перевірялись всі сценарії, що дозволило стверджувати про успішну реалізацію програми.

Загалом логіка роботи програми відповідає процесу функціонування, описаному в розділі 3. Для різних користувачів передбачено різні варіанти доступу. Використовується три посилання для доступу трьох різних типів користувачів, що дозволяє розмежовувати повністю їх логіку роботи.

Основна функціональність доступна працівникам служби організації заходів, які після авторизації можуть взаємодіяти з програмою за допомогою

основного меню, яке включає розмежовані основні сценарії використання програми (рис. 4.1).

The screenshot displays the 'EventCostsPlan' application interface. On the left is a dark sidebar with a red header 'EventCostsPlan' and a user profile picture. The sidebar menu includes 'Заходи' (Events), 'Шаблони' (Templates), 'Порівняння' (Comparison), 'Пошук' (Search), 'Групи' (Groups), and 'Профіль' (Profile). The main content area shows details for an event titled 'Весілля Євгена та Ганни'. Below the event details is a green box with '3 Нових коментарів' and a rocket icon with 'Всі голосування'. To the right is a table titled 'Організаційні витрати' (Organizational expenses) with columns for item number, name, planned, actual, and accounted amounts.

EventCostsPlan Добрий день, Олександр Рудченко Вийти

Весілля Євгена та Ганни

Тип заходу: Весілля
20.07.2022 19:00 - 21.07.2022 02:00

Опис: Весілля в форматі для друзів в приватному котеджі з полем для гольфу.
Відкритість: інформація не доступна публічно
Стан: відкритий
Доступ команди: неважливий
Місце проведення: Котедж Франківський
Кількість учасників: 30
Стан: відкрита
Теги: гольф, весілля

[Редагувати](#) [Порівняння](#) [Шаблон](#)

3 Нових коментарів
[Всі коментарі](#)

Всі голосування

Організаційні витрати

#	Назва	Заплановано	Фактично	Розраховано
-	Сумарно	36000 грн.	0 грн.	0 грн.
1	Оренда ресторану	10000 грн.	0 грн.	0 грн.
2	Платня ведучого	7000 грн.	0 грн.	0 грн.
3	Створення плану заходу	8000 грн.	0 грн.	0 грн.
4	Музична підтримка	7000 грн.	0 грн.	0 грн.
5	Виступ танцювального балету	4000 грн.	0 грн.	0 грн.

[Додати статтю витрат](#)

Рисунок 4.1 – Відображення бюджету витрат заходу

Пункт меню **Заходи** має частину, яку можна розгорнути, вона включає власні заходи, тобто створені авторизованим працівником, і заходи груп, тобто заходи груп, до яких включено працівника. Кожен з цих варіантів дозволяє відкрити перелік з набором всіх зустрічей, а також кнопкою для створення нової зустрічі. Будь-яку зустріч можна відкрити на окремій сторінці, що і представлена на рис. 4.1. За зустріччю виводиться загальна інформація, нижче інформація про те, яка кількість коментарів до різних статей цього бюджету ще не переглянута працівником. Надаються кнопки для переходу до редагування інформації зустрічі, для створення шаблону на

основі зустрічі, для додавання зустрічі для порівняння (тоді потрібно додати ще одну зустріч після цього, а кнопка на такій сторінці перетвориться на Зберегти шаблон).

На сторінці міститься перелік всіх статей бюджету, розділених за типами. Кожний тип представлений окремою таблицею. Всередині таблиці є кнопка для додавання за ним нової статті витрат. Кожну статтю можна відкрити і перейти до окремої її сторінки (рис. 4.2). Нижче на сторінці є підсумок сумарних витрат за всіма статтями та можливість додавання учасників до заходу.

EventCostsPlan Добрий день, Олександр Рудченко [Вийти](#)

Весілля Івана та Марії Шинкаренів
 Стаття витрат: Оренда автобусів для гостей
 Група статей: Оренда транспорту
 Запланований бюджет: 10000 грн.
 Фактичний бюджет: 9500 грн.
 Обчислений бюджет: 7500 грн.
 Особливості: Необхідно орендувати великогабаритний автобус та мікроавтобус з їх світковим прикрашенням на 4 години офіційної частини

[Редагувати](#) [Назад](#)

Витрати за статтю

#	Назва	Сума	Користувач	Дата
1	Передплата за оренду автобуса ISUZU A0962	2000 грн.	egello	8.06.2022
2	Передплата за оренду мікроавтобуса Mercedes-Benz Sprinter 316	500 грн.	egello	8.06.2022
3	Внесення повної плати за оренду автобуса ISUZU A0962	5000 грн.	egello	9.06.2022

[Додати витрату](#)

Коментарі

Віталій Зоряний 7.06.2022 15:02

Ми не зможемо обійтися одним великим автобусом, потрібен додатково один маленький.

Рисунок 4.2 – Відображення статті витрат

На сторінці статті витрат надається можливість внесення самих витрат за допомогою відповідної кнопки, а також перелік всіх виконаних на поточний момент витрат. Також розташовуються коментарі від всіх працівників. Коментарі виводяться в верхній частині нові, а потім після

розділення з підписом Раніше – старі, тобто ті, які вже переглянув працівник раніше.

Це основні засоби роботи з бюджетами заходів. Окрім того зі сторінки з переліком всіх заходів для кожного заходу доступна кнопка Розподіл, що відкриває сторінку з діаграмами (рис. 4.3). Ця сторінка містить діаграми з сумарним розподілом всіх витрат за типами статей та за групами статей (нижче).

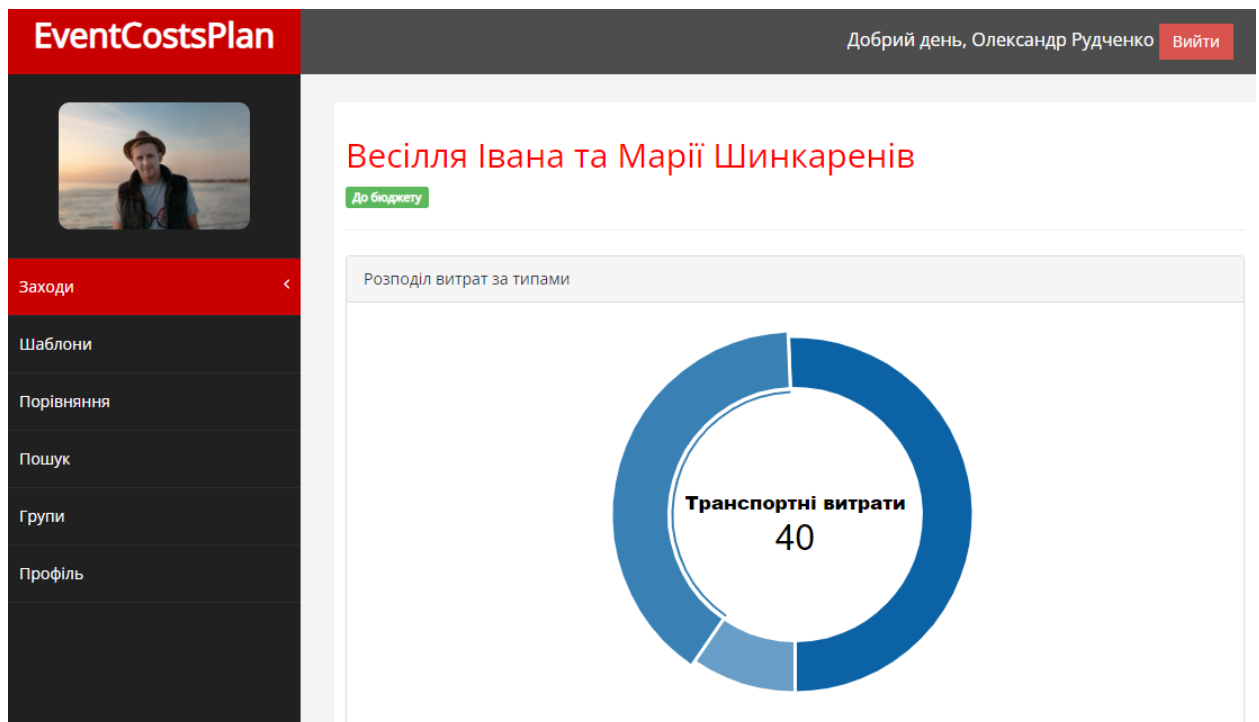


Рисунок 4.3 – Відображення розподілу витрат за типами статей

Переглянути створені шаблони працівник може з пункту меню Шаблони. Звідти для кожного доступна кнопка Вилучити.

Переглянути всі порівняння працівник може з пункту меню Порівняння.

Виконати пошук серед всіх відкритих бюджетів та бюджетів, які доступні працівнику можна з пункту меню Пошук. Параметри пошуку були зазначені в розділі 3 детально.

Створити нові групи, керувати доступом працівників у існуючій групі може з пункту меню Групи.

Адміністратор працює з програмою зі стандартного інтерфейсу Django, керуючи наявними типами, групами статей витрат, майданчиками.

Учасник має спрощений інтерфейс (рис. 4.4), основним засобом якого є перегляд створених бюджетів зустрічей, для яких він віднесений як учасник.

При відкритті пункту меню заходів учаснику відображається набір усіх таких заходів, можна відкрити будь-який з них і переглянути бюджет. Функції з редагування, додавання, описані для працівника, учаснику недоступні.

EventCostsPlan

Добрий день, Іван Агапов [Вийти](#)

Всі ваші заходи

Організаційні витрати

#	Назва	Дати	Витрати	Кількість учасників
1	Весілля Євгена та Ганни	20.07.2022 19:00	79000 грн.	30
2	Літературна конференція	14.06.2022 12:00	144000 грн.	100

Рисунок 4.4 – Відображення заходів учасника

4.4 Висновки за розділом 4

Розглянуто процес роботи працівників служби організації заходу та учасників заходів з програмою. Виконано тестування програми.

Розроблена і представлена на основі інтерфейсів користувача у даному розділі програма може використовуватися для підтримки роботи над плануванням витрат при організації заходів, надаючи для цього не тільки можливості внесення планів і фактичних витрат, але і коментування,

голосування, а також розділення доступу різних працівників до даних процесів, що розширює зручність її застосування для планування витрат за заходами.

ВИСНОВКИ

Організація заходів є дуже популярною сферою. Для її підтримки можуть використовуватися дуже різні програмні засоби, починаючи від найпростіших таблиць і документів, і завершуючи спеціалізованими рішеннями. Проте найпростіші варіанти роботи дозволяють створювати план витрат, вносити такі витрати, але ніяк не підтримують взаємодію між працівниками стосовно визначення конкретних витрат, сум цих витрат, що є важливим і нетривіальним. Тому метою роботи було розробити ПЗ планування витрат на організацію заходів для підтримки спільної роботи працівників над прийняттям спільних рішень і внесенням цих рішень.

При виконанні роботи забезпечено повний цикл розробки програми, починаючи від вивчення предметної області з формулюванням вимог до програми на основі цього вивчення, охоплюючи вибір мови програмування Python, фреймворку веброзробки Django та системи керування БД PostgreSQL для підтримки реалізації програми, проектування і створення БД, визначення структури програми, завершуючи реалізацією вебсистеми і на основі цієї структури та базовим тестування цієї системи.

Створена програма може використовуватися для підтримки роботи над плануванням витрат при організації заходів, надаючи для цього не тільки можливості внесення планів і фактичних витрат, але і коментування, голосування за спірними рішеннями щодо статей витрат, а також розділення доступу різних працівників до даних процесів, в тому числі перегляду бюджетів, редагування даних про захід загалом, внесення бюджету заходу, що в результаті розширює зручність її застосування для планування витрат за заходами. Враховуючи реалізацію програми як вебсистеми, вона не висуває додаткових вимог, як наприклад, встановлення окремого застосунку, а потребує тільки розгортання на вебсервері для забезпечення доступу усіх користувачів до неї.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Events Industry's Top Marketing Statistics, Trends, and Data [Electronic resource]. – Access mode : <https://www.bizzabo.com/blog/event-marketing-statistics/>.
2. 5 Future Event Planning Industry Trends You Need to Know About [Electronic resource]. – Access mode : <https://www.socialtables.com/blog/event-planning/future-industry-trends/>.
3. 80 Event Statistics You Need to Know for 2022 [Electronic resource]. – Access mode : <https://visme.co/blog/event-statistics/>.
4. Creating an Event Budget in 2022 [Electronic resource]. – Access mode : <https://whova.com/blog/creating-event-budget-template/>.
5. The Complete Guide to Creating an Event Budget [Electronic resource]. – Access mode : <https://www.aventri.com/strategy/event-budget-strategy>.
6. Pro Party Planner [Electronic resource]. – Access mode : <https://propartyplanner.com/>.
7. Teague, K. Best party-planning apps for 2022 [Electronic resource] / K. Teague. – Access mode : <https://www.cnet.com/tech/services-and-software/best-party-planning-apps/>.
8. Trello. Керуйте проєктами своєї команди звідусіль. Trello [Електрон. ресурс]. – Режим доступу : <https://trello.com/home>.
9. Trello For Event Planning and Event Management [Electronic resource]. – Access mode : <https://www.bluecatreports.com/blog/trello-for-event-planning/>.
10. Django documentation. Django documentation. Django [Electronic resource]. – Access mode : <https://docs.djangoproject.com/en/4.0/>.
11. Binary Admin Free Website Template | Free CSS Templates | Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page204/binary-admin>.

ДОДАТОК А
Технічне завдання

Вступ

Планування офіційних або неофіційних заходів є достатньо важливим процесом, від якого залежить, чи принесе цей захід задоволення учасникам, чи ні. Планування може виконуватися і у звичайному нотатнику, проте спеціалізований інструмент має надавати набір можливостей, які додадуть зручності роботі в групі над заходами, які не можна організувати наодинці.

A.1 Підстави для розробки

Розробка в межах дипломної кваліфікаційної роботи реалізується на підставі наказу № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка» за темою «Програмне забезпечення планування витрат на організацію заходів».

A.2 Призначення розробки

Програма призначена для надання працівникам служб з організації заходів засобів планування бюджету цих заходів, командної роботи над цими бюджетами, винесення спільних рішень щодо статей витрат в бюджеті заходу, аналізу існуючих бюджетів, а учасникам подій – надання можливостей ознайомлення з витратами за їхніми заходами.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функції, які виконуватиме програма за результатами розробки:

- створення бюджету заходу;
- редагування бюджету заходу;
- створення бюджету заходу на основі шаблону;

- відображення бюджету заходу;
- відображення розподілу бюджету між типами і групами витрат;
- відображення бюджетів заходів, відкритих для групи;
- додавання статей витрат бюджету заходу;
- редагування статей витрат бюджету заходу;
- внесення витрат за статтями бюджету;
- створення груп працівників;
- керування доступом працівників у групі;
- визначення доступу групи працівників до бюджету;
- проведення голосування за статтею витрат;
- коментування статті витрат;
- відображення повної інформації статті витрат в бюджеті заходу;
- визначення бюджету в якості шаблону;
- створення порівняння бюджетів заходів;
- відображення порівнянь бюджетів групою;
- відображення нових коментарів за всіма статтями з бюджету;
- відображення всіх голосувань за бюджетом заходу;
- визначення тегів бюджету заходу;
- відображення бюджетів заходів за тегом;
- пошук серед бюджетів заходів;
- відображення доступних бюджетів заходів за майданчиком проведення;
- відображення заходів за власною участю;
- визначення учасників заходу;
- внесення типів витрат, груп витрат, майданчиків, типів заходів.

A.3.2 Умови експлуатації

Рекомендовано для забезпечення роботи програми виділити системного адміністратора, до зобов'язань якого віднести підтримку роботи

вебсервера, сервера БД та внесення за потреби базових даних.

А.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби в мінімальній конфігурації, достатній для роботи програми, охоплюють 1 Гб пам'яті жорсткого диску, 4 Гб оперативної пам'яті, 4 ядра процесору (частота 2,8 ГГц).

А.4 Порядок контролю та приймання

Остаточна перевірка отриманих результатів виконання роботи здійснюється під час захисту дипломної роботи.

ДОДАТОК Б
Текст програми

```

import datetime
from django.views.generic.edit import CreateView, DeleteView
from django.views.generic import ListView, UpdateView
from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse_lazy
from .models import EventBudget
from .models import EventType
from .models import Venue
from .models import StaffGroup
from .models import Cost
from .models import CostValue
from .models import CostGroup
from .models import StaffGroup
from .models import CostVote
from .models import CostComment
from .models import StaffCostComment
from .models import EventBudgetComparison
from .models import Staff
from .models import Participant
from .models import EventParticipant
from .models import Tag
from .models import EventTemplate
from .models import StaffGroupParticipant
from datetime import date

```

```

class EventBudgetCreateView (LoginRequiredMixin, CreateView):
    model = EventBudget
    fields = ['event_title',
              'plan_start',
              'plan_end',
              'event_information',
              'planned_budget',
              'actual_budget',
              'notation',
              'participants_quantity',
              'status',
              'opened']

```

```

template_name = 'eventbudget_add.html'

def get_success_url(self):
    return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.pk})

def get_context_data(self, **kwargs):
    event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
    kwargs['event_type'] = event
    venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
    kwargs['venue'] = venue
    staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
    kwargs['staffgroup'] = staffgroup
    return super().get_context_data(**kwargs)

def form_valid(self, form):
    auth_user = self.request.user
    creator = Staff.objects.get(id = auth_user.id)
    if creator:
        self.actual_calculated_budget = 0
        event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
        self.eventtype_id = event
        venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
        self.venue_id = venue
        staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
        self.staffgroup_id = staffgroup
        self.created = date.today()
        self.creator = creator
        return super().form_valid(form)

class EventBudgetTemplateCreateView (LoginRequiredMixin,
CreateView):
    model = EventBudget

```

```

fields = ['event_title',
          'plan_start',
          'plan_end',
          'event_information',
          'planned_budget',
          'actual_budget',
          'notation',
          'participants_quantity',
          'status',
          'opened']

template_name = 'eventbudget_template_add.html'

def get_success_url(self):
    return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.pk})

def get_context_data(self, **kwargs):
    event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
    kwargs['event_type'] = event
    venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
    kwargs['venue'] = venue
    staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
    kwargs['staffgroup'] = staffgroup
    return super().get_context_data(**kwargs)

def form_valid(self, form):
    auth_user = self.request.user
    creator = Staff.objects.get(id = auth_user.id)
    if creator:
        self.actual_calculated_budget = 0
        template = get_object_or_404(EventTemplate,
id=self.kwargs['template'])
        event_template = template.eventbudget_id
        event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
        if event:

```

```

        self.eventtype_id = event
    else:
        self.eventtype_id =
event_template.eventtype_id
        venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
        if venue:
            self.venue_id = venue
        else:
            self.venue_id = event_template.venue_id
        staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
        if venue:
            self.staffgroup_id = staffgroup
        else:
            self.staffgroup_id =
event_template.staffgroup_id
        self.created = date.today()
        self.creator = creator
        if self.kwargs['repeat']:
            costs = event_template.costs
            for cost in costs:
                Cost.objects.create(eventbudget_id =
self.object, costgroup_id = cost.costgroup_id, short_name =
cost.short_name, long_name = cost.long_name, planned_budget =
cost.planned_budget)
        return super().form_valid(form)

```

```

class EventBudgetUplateView (LoginRequiredMixin, UpdateView):
    model = EventBudget
    fields = ['event_title',
              'plan_start',
              'plan_end',
              'event_information',
              'planned_budget',
              'actual_budget',
              'notation',
              'participants_quantity',
              'status',

```

```

        'opened']
    template_name = 'eventbudget_update.html'

    def get_success_url(self):
        return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
        kwargs['event_type'] = event
        venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
        kwargs['venue'] = venue
        staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
        kwargs['staffgroup'] = staffgroup
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        auth_user = self.request.user
        creator = Staff.objects.get(id = auth_user.id)
        if self.creator == creator:
            event = get_object_or_404(EventType,
id=self.kwargs['event_type'])
            self.eventtype_id = event
            venue = get_object_or_404(Venue,
id=self.kwargs['venue'])
            self.venue_id = venue
            staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['staffgroup'])
            self.staffgroup_id = staffgroup
            return super().form_valid(form)

class EventBudgetDetailView (DetailView):
    model = EventBudget
    template_name = 'eventbudget.html'

```

```

def get_context_data(self, **kwargs):
    ebdv_contextdata = super().get_context_data(**kwargs)
    costs = Cost.objects.filter(eventbudget_id__id =
self.object.pk)
    costs = costs.order_by('costgroup_id')
    ebdv_contextdata['costs'] = costs
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    comments =
CostComment.objects.filter(cost_id__eventbudget_id__id =
self.object.pk)
    notseen_comments = comments.exclude(staffs__contains =
staff)
    ebdv_contextdata['notseen_comments'] =
notseen_comments.count()
    return ebdv_contextdata

```

```

class CreatedEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_byme.html'

    def get_queryset(self):
        auth_user = self.request.user
        creator = Staff.objects.get(id = auth_user.id)
        budgets = EventBudget.objects.filter(creator =
creator)

        budgets = budgets.order_by('-created')
        return budgets

```

```

class GroupsEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_bygroups.html'

    def get_queryset(self):
        auth_user = self.request.user
        staff = Staff.objects.get(id = auth_user.id)
        sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
        groups = []

```

```

        for sgroup in sgroups:
            if sgroup.staff_id not in groups:
                groups.append(sgroup.staff_id)
        budgets = EventBudget.objects.filter(staffgroup_id__in
= groups)

        budgets = budgets.order_by('-created')
        return budgets

class GroupEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_bygroup.html'

    def get_queryset(self):
        group = self.request.GET.get('group')
        group = StaffGroup.objects.get(id = group.id)
        budgets = EventBudget.objects.filter(staffgroup_id__in
= group)

        budgets = budgets.order_by('-created')
        return budgets

class ParticipantEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_byparticipant.html'

    def get_queryset(self):
        auth_user = self.request.user
        participant = Participant.objects.get(id =
auth_user.id)

        pevents =
EventParticipant.objects.filter(participant_id = participant)
        events = []
        for event in pevents:
            if event.eventbudget_id not in events:
                events.append(event.eventbudget_id)
        events = events.order_by('-created')
        return events

class TagEventBudgetsView (ListView):
    model = EventBudget

```

```

template_name = 'eventbudgets_bytag.html'
def get_queryset(self):
    tag = self.request.GET.get('tag')
    tag = Tag.objects.get(name = tag)
    auth_user = self.request.user
    participant = Participant.objects.get(id =
auth_user.id)
    if participant:
        pevents =
EventParticipant.objects.filter(participant_id = participant)
        events = []
        for event in pevents:
            if event.eventbudget_id not in events:
                events.append(event.eventbudget_id)
        events = events.order_by('-created')
        events = events.filter(tags__contains = tag)
        return events
    else:
        staff = Staff.objects.get(id = auth_user.id)
        sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
        groups = []
        for sgroup in sgroups:
            if sgroup.staff_id not in groups:
                groups.append(sgroup.staff_id)
        budgets =
EventBudget.objects.filter(staffgroup_id__in = groups)
        budgets = budgets.filter(tags__contains = tag)
        open_budgets = EventBudget.objects.filter(opened
= True, tags__contains = tag)
        budgets = budgets.filter(tags__contains = tag)
        budget_results = []
        for budget in budgets:
            budget_results.append(budget)
        for budget in open_budgets:
            if budget not in budget_results:
                budget_results.append(budget)
        budgets = budgets.order_by('-created')
        return budgets

```

```

class VenueEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_byvenue.html'

    def get_queryset(self):
        venue = self.request.GET.get('venue')
        venue = Venue.objects.get(id = venue)
        auth_user = self.request.user
        search_list = []
        participant = Participant.objects.get(id =
auth_user.id)
        if participant:
            pevents =
EventParticipant.objects.filter(participant_id = participant)
            events = []
            for event in pevents:
                if event.eventbudget_id not in events:
                    events.append(event.eventbudget_id)
            events = events.order_by('-created')
            events = events.filter(venue_id = venue)
            return events
        else:
            staff = Staff.objects.get(id = auth_user.id)
            sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
            groups = []
            for sgroup in sgroups:
                if sgroup.staff_id not in groups:
                    groups.append(sgroup.staff_id)
            budgets =
EventBudget.objects.filter(staffgroup_id__in = groups)
            budgets = budgets.filter(venue_id = venue)
            open_budgets = EventBudget.objects.filter(opened
= True, venue_id = venue)
            budget_results = []
            for budget in budgets:
                budget_results.append(budget)
            for budget in open_budgets:
                if budget not in budget_results:

```

```

        budget_results.append(budget)
    budgets = budgets.order_by('-created')
    return budgets

class FindEventBudgetsView (ListView):
    model = EventBudget
    template_name = 'eventbudgets_find.html'

    def get_queryset(self):
        typeid = self.request.GET.get('type')
        name = self.request.GET.get('name')
        costs_max = self.request.GET.get('costs_max')
        costs_min = self.request.GET.get('costs_min')
        time_start = self.request.GET.get('time_start')
        time_end = self.request.GET.get('time_end')
        costs_max = float(costs_max)
        costs_min = float(costs_min)
        time_start = datetime.datetime.strptime(time_start,
"%d.%m.%Y").date()
        time_end = datetime.datetime.strptime(time_end,
"%d.%m.%Y").date()
        venue = self.request.GET.get('venue')
        venue = Venue.objects.get(id = venue)
        tags = str(self.request.POST.get('tags'))
        tags = tags.split(',')
        tags_parsed = []
        for tag in tags:
            tag = Tag.objects.get(name = tag)
            tags_parsed.append(tag)
        auth_user = self.request.user
        search_list = []
        participant = Participant.objects.get(id =
auth_user.id)
        if participant:
            pevents =
EventParticipant.objects.filter(participant_id = participant)
            events = []
            for event in pevents:
                if event.eventbudget_id not in events:

```

```

        events.append(event.eventbudget_id)
    events = events.order_by('-created')
    search_list = events
else:
    staff = Staff.objects.get(id = auth_user.id)
    sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
    groups = []
    for sgroup in sgroups:
        if sgroup.staff_id not in groups:
            groups.append(sgroup.staff_id)
    budgets =
EventBudget.objects.filter(staffgroup_id__in = groups)
    open_budgets = EventBudget.objects.filter(opened
= True)

    budget_results = []
    for budget in budgets:
        budget_results.append(budget)
    for budget in open_budgets:
        if budget not in budget_results:
            budget_results.append(budget)
    budgets = budgets.order_by('-created')
    search_list = budgets
    if name:
        search_list =
search_list.filter(event_title__icontains = name)
    if typeid:
        search_list = search_list.filter(eventtype_id =
typeid)
    if venue:
        search_list = search_list.filter(venue_id =
venue)
    if costs_max:
        search_list =
search_list.filter(actual_budget__lte = costs_max)
    if costs_min:
        search_list =
search_list.filter(actual_budget__gte = costs_min)
    if time_start:

```

```

        search_list = search_list.filter(plan_start__gte
= time_start)
    if time_end:
        search_list = search_list.filter(plan_end__lte =
time_end)

    search_list = search_list.order_by('-created')
    newlist = []
    for tag in tags:
        with_tags = search_list.filter(tags__contains =
tag)

        for event in with_tags:
            if event not in newlist:
                newlist.append(event)
    return newlist

```

```

class CostCreateView (LoginRequiredMixin, CreateView):
    model = Cost
    fields = ['short_name',
              'long_name',
              'planned_budget',
              'actual_budget',
              'notation']
    template_name = 'cost_add.html'

    def get_success_url(self):
        return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.eventbudget_id.pk})

    def get_context_data(self, **kwargs):
        event = get_object_or_404(EventBudget,
id=self.kwargs['event'])
        kwargs['event'] = event
        costgroup = get_object_or_404(CostGroup,
id=self.kwargs['costgroup'])
        kwargs['costgroup'] = costgroup
        return super().get_context_data(**kwargs)

```

```

def form_valid(self, form):
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
    groups = []
    for sgroup in sgroups:
        if sgroup.staff_id not in groups:
            groups.append(sgroup.staff_id)
    budgets = EventBudget.objects.filter(staffgroup_id__in
= groups)

    if self.object.eventbudget_id in budgets:
        event = get_object_or_404(EventBudget,
id=self.kwargs['event'])
        costgroup = get_object_or_404(CostGroup,
id=self.kwargs['costgroup'])
        self.costgroup_id = costgroup
        self.eventbudget_id = event
        return super().form_valid(form)

class CostUpdateView (LoginRequiredMixin, UpdateView):
    model = Cost
    fields = ['short_name',
              'long_name',
              'planned_budget',
              'actual_budget',
              'notation']
    template_name = 'cost_update.html'

    def get_success_url(self):
        return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.eventbudget_id.pk})

    def get_context_data(self, **kwargs):
        costgroup = get_object_or_404(CostGroup,
id=self.kwargs['costgroup'])
        kwargs['costgroup'] = costgroup
        return super().get_context_data(**kwargs)

```

```

def form_valid(self, form):
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
    groups = []
    for sgroup in sgroups:
        if sgroup.staff_id not in groups:
            groups.append(sgroup.staff_id)
    budgets = EventBudget.objects.filter(staffgroup_id__in
= groups)
    if self.object.eventbudget_id in budgets:
        costgroup = get_object_or_404(CostGroup,
id=self.kwargs['costgroup'])
        self.costgroup_id = costgroup
        return super().form_valid(form)

class CostValueCreateView (LoginRequiredMixin, CreateView):
    model = CostValue
    fields = ['information',
              'costed']
    template_name = 'costvalue_add.html'

    def get_success_url(self):
        return reverse_lazy('eventcostview', kwargs={'pk':
self.object.cost_id.pk})

    def get_context_data(self, **kwargs):
        cost = get_object_or_404(Cost, id=self.kwargs['cost'])
        kwargs['cost'] = cost
        return super().get_context_data(**kwargs)

class CostValueUpdateView (LoginRequiredMixin, UpdateView):
    model = CostValue
    fields = ['costed']
    template_name = 'costvalue_update.html'

    def get_success_url(self):

```

```

        return reverse_lazy('eventcostview', kwargs={'pk':
self.object.cost_id.pk})

class CostCommentCreateView (LoginRequiredMixin, CreateView):
    model = Cost
    fields = ['comment_text']
    template_name = 'comment_add.html'

    def get_success_url(self):
        return reverse_lazy('eventcostview', kwargs={'pk':
self.object.cost_id.pk})

    def get_context_data(self, **kwargs):
        cost = get_object_or_404(Cost, id=self.kwargs['cost'])
        kwargs['cost'] = cost
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        auth_user = self.request.user
        staff = Staff.objects.get(id = auth_user.id)
        sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
        groups = []
        for sgroup in sgroups:
            if sgroup.staff_id not in groups:
                groups.append(sgroup.staff_id)
        budgets = EventBudget.objects.filter(staffgroup_id__in
= groups)

        if self.object.eventbudget_id in budgets:
            cost = get_object_or_404(Cost,
id=self.kwargs['cost'])
            self.cost_id = cost
            self.staff_id = staff
            self.created = date.today()
            return super().form_valid(form)

class CostDetailView (DetailView):
    model = Cost
    template_name = 'cost.html'

```

```

def get_context_data(self, **kwargs):
    cdv_contextdata = super().get_context_data(**kwargs)
    costs = CostValue.objects.filter(cost_id__id =
self.object.pk)
    costs = costs.order_by('created')
    cdv_contextdata ['costs'] = costs
    votes = CostVote.objects.filter(cost_id__id =
self.object.pk)
    cdv_contextdata ['votes'] = votes
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    comments =
CostComment.objects.filter(cost_id__eventbudget_id__id =
self.object.pk)
    notseen_comments = comments.exclude(staffs__contains =
staff)
    notseen_comments = notseen_comments.order_by('-
created')
    cdv_contextdata ['notseen_comments'] =
notseen_comments
    seen_comments = comments.filter(staffs__contains =
staff)
    cdv_contextdata ['seen_comments'] = seen_comments
    return cdv_contextdata

```

```

class EventTemplateCreateView (LoginRequiredMixin, CreateView):
    model = EventTemplate
    template_name = 'template_add.html'

    def get_success_url(self):
        return reverse_lazy('templatesview', kwargs={'pk':
self.object.staff_id.pk})

    def get_context_data(self, **kwargs):
        event = get_object_or_404(EventBudget,
id=self.kwargs['event'])
        kwargs['event'] = event
        return super().get_context_data(**kwargs)

```

```

def form_valid(self, form):
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    self.staff_id = staff
    eb = get_object_or_404(EventBudget,
id=self.kwargs['event'])
    self.eventbudget_id = eb
    return super().form_valid(form)

class EventTemplateDeleteView (DeleteView):
    model = EventTemplate

class TemplatesView (ListView):
    model = EventTemplate
    template_name = 'templates.html'

    def get_queryset(self):
        auth_user = self.request.user
        staff = Staff.objects.filter(id = auth_user.id)
        templates = EventTemplate.objects.filter(staff_id =
staff)

        templates = templates.order_by('-id')
        return templates

class EventBudgetComparisonCreateView (LoginRequiredMixin,
CreateView):
    model = EventBudgetComparison
    template_name = 'comparison_add.html'

    def get_success_url(self):
        return reverse_lazy('personcomparisonsview',
kwargs={'pk': self.object.staff_id.pk})

    def get_context_data(self, **kwargs):
        event = get_object_or_404(EventBudget,
id=self.kwargs['event1'])
        kwargs['event1'] = event
        event = get_object_or_404(EventBudget,
id=self.kwargs['event2'])

```

```

        kwargs['event2'] = event
        staffgroup = get_object_or_404(StaffGroup,
id=self.kwargs['group'])
        kwargs['group'] = staffgroup
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        auth_user = self.request.user
        staff = Staff.objects.get(id = auth_user.id)
        self.staff_id = staff
        eb = get_object_or_404(EventBudget,
id=self.kwargs['event1'])
        self.eventbudget1_id = eb
        eb = get_object_or_404(EventBudget,
id=self.kwargs['event2'])
        self.eventbudget2_id = eb
        sg = get_object_or_404(StaffGroup,
id=self.kwargs['group'])
        self.staffgroup_id = sg
        return super().form_valid(form)

class MyEventBudgetComparisonsView (ListView):
    model = EventBudgetComparison
    template_name = 'mycomparisons.html'

    def get_queryset(self):
        auth_user = self.request.user
        staff = Staff.objects.filter(id = auth_user.id)
        comparisons =
EventBudgetComparison.objects.filter(staff_id = staff)
        comparisons = templates.order_by('-id')
        return comparisons

class EventBudgetComparisonDeleteView (DeleteView):
    model = EventBudgetComparison

class EventBudgetComparisonsView (ListView):
    model = EventBudgetComparison

```

```

template_name = 'comparisonss.html'

def get_queryset(self):
    auth_user = self.request.user
    staff = Staff.objects.get(id = auth_user.id)
    sgroups =
StaffGroupParticipant.objects.filter(staff_id = staff)
    groups = []
    for sgroup in sgroups:
        if sgroup.staff_id not in groups:
            groups.append(sgroup.staff_id)
    comparisons =
EventBudgetComparison.objects.filter(staffgroup_id__in = groups)
    comparisons = comparisons.order_by('-id')
    return comparisons

class EventTagCreateView (LoginRequiredMixin, CreateView):
    model = EventTag
    template_name = 'tag_add.html'

    def get_success_url(self):
        return reverse_lazy('eventbudgetview', kwargs={'pk':
self.object.eventbudget_id.pk})

    def get_context_data(self, **kwargs):
        event = get_object_or_404(EventBudget,
id=self.kwargs['event'])
        kwargs['event'] = event
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        eb = get_object_or_404(EventBudget,
id=self.kwargs['event'])
        self.eventbudget_id = eb
        tag = self.kwargs['tag']
        tagin = Tag.objects.get(name = tag)
        if tagin:
            self.tag_id = tagin
        else:

```

```

        Tag.objects.create(name = tag)
        tagin = Tag.objects.get(name = tag)
        self.tag_id = tagin
    return super().form_valid(form)

class EventBudgetNewCommentsView (ListView):
    model = CostComment
    template_name = 'budget_newcomments.html'

    def get_queryset(self):
        event = self.kwargs['event']
        comments =
CostComment.objects.get(cost_id__eventbudget_id = event)
        notseen_comments = comments.exclude(staffs__contains =
staff)

        notseen_comments = notseen_comments.order_by('-
created')

        cdv_contextdata ['notseen_comments'] =
notseen_comments
        seen_comments = comments.filter(staffs__contains =
staff)

        cdv_contextdata ['seen_comments'] = seen_comments
        return cdv_contextdata

class EventBudgetNewCommentsView (DetailView):
    model = CostComment
    template_name = 'budget_newcomments.html'

    def get_context_data(self, **kwargs):
        ebnc_contextdata = super().get_context_data(**kwargs)
        event = self.kwargs['event']
        comments =
CostComment.objects.get(cost_id__eventbudget_id = event)
        auth_user = self.request.user
        staff = Staff.objects.get(id = auth_user.id)
        notseen_comments = comments.exclude(staffs__contains =
staff)

        notseen_comments = notseen_comments.order_by('-
created')

```

```
        ebnc_contextdata ['notseen_comments'] =  
notseen_comments  
    return ebnc_contextdata
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАНУВАННЯ ВИТРАТ НА
ОРГАНІЗАЦІЮ ЗАХОДІВ

Виконав
студент групи КНТ-128

І.Р. Хаустов

Керівник

Т.В. Голуб

2022

Рисунок В.1 – Слайд № 1

Об'єкт, предмет, мета роботи

Об'єктом роботи є процес планування організації заходів.

Предметом роботи є програмне забезпечення планування витрат на організацію заходів.

Мета роботи – розробити програмне забезпечення планування витрат на організацію заходів для підтримки спільної роботи працівників над прийняттям спільних рішень і внесенням цих рішень.

Рисунок В.2 – Слайд № 2

Завдання роботи

- ✓ визначення вимог до розробки;
- ✓ проєктування бази даних;
- ✓ проєктування програмного забезпечення
планування витрат на організацію заходів;
- ✓ програмна реалізація вебзастосунку
планування витрат на організацію заходів.

3

Рисунок В.3 – Слайд № 3

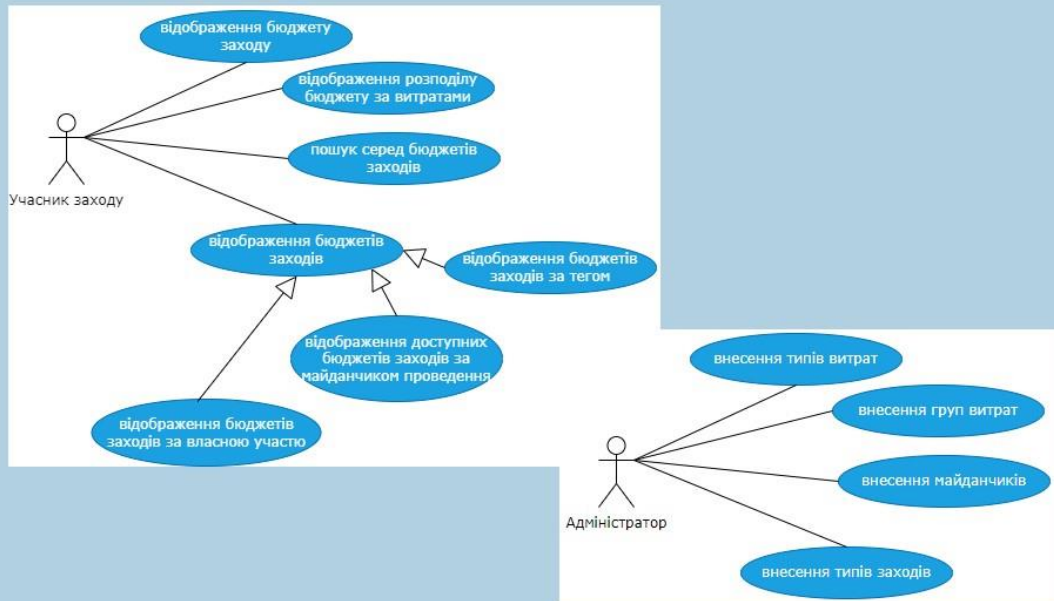
Вибір підходу до веброзробки

Критерій	Веброзробка на основі фреймворків	Стандартна веброзробка
Повторне використання коду	Активне	Доволі низьке, але можливе за рахунок виділення функцій і бібліотек
Гнучкість до змін	Висока	Низька
Час розробки	Менший, але потребує часу на вивчення фреймворку	Більший
Наявні вбудовані засоби	Підключення бібліотек мови програмування, зазвичай мають засоби адміністрування, шаблонізації, спрощення роботи з даними, але залежить від фреймворку	Можливе підключення бібліотек у відповідність з мовою програмування

4

Рисунок В.4 – Слайд № 4

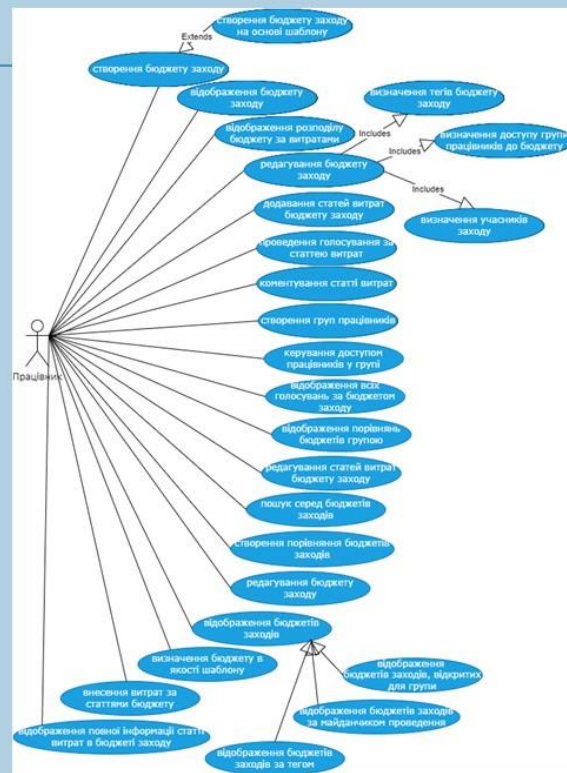
Діаграма прецедентів учасника заходу і адміністратора



5

Рисунок В.5 – Слайд № 5

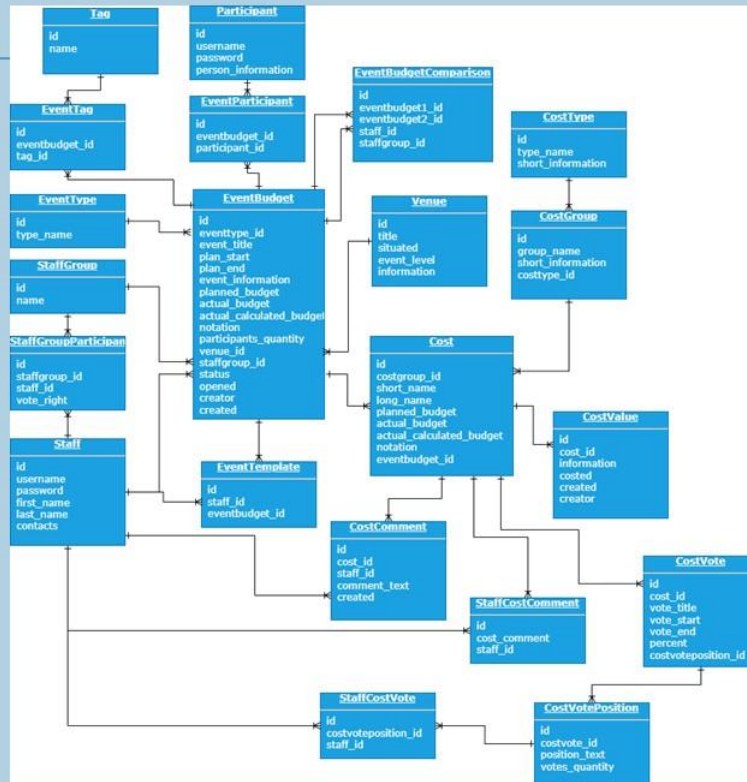
Діаграма прецедентів працівника



6

Рисунок В.6 – Слайд № 6

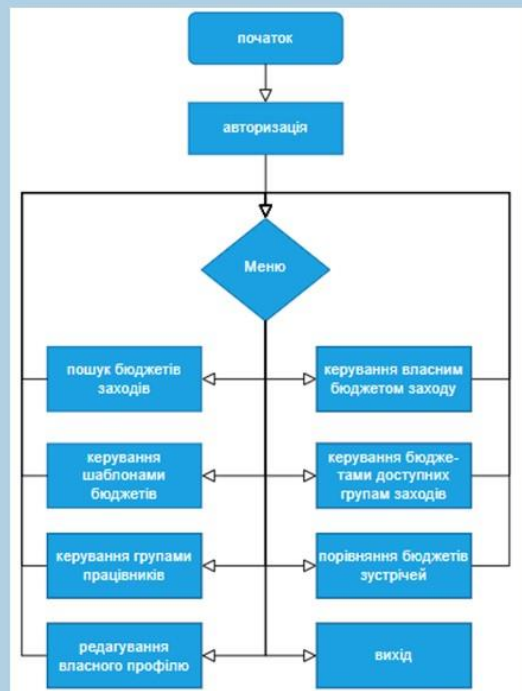
Схема бази даних



7

Рисунок В.7 – Слайд № 7

Функціональна схема програми для працівника



8

Рисунок В.8 – Слайд № 8

Відображення бюджету витрат заходу

EventCostsPlan

Добрий день, Олександр Рудченко [Вийти](#)

[Заходи](#)
[Шаблони](#)
[Порівняння](#)
[Пошук](#)
[Групи](#)
[Профіль](#)

Весілля Євгена та Ганни

Тип заходу: Весілля
20.07.2022 19:00 - 21.07.2022 02:00

Опис: Весілля в форматі для друзів в приватному котеджі з полем для гольфу.
Відкритість: інформація не доступна публічно
Стан: відкритий
Доступ команд: неважливий
Місце проведення: Котедж Франківський
Кількість учасників: 30
Стан: відкритий
Теги: гольф, весілля

[Редагувати](#) [Порівняти](#) [Шаблон](#)

З Нових коментарів

Відповісти

Відповісти

Організаційні витрати

#	Назва	Заплановано	Фактично	Розраховано
-	Сумарно	36000 грн.	0 грн.	0 грн.
1	Оренда ресторану	10000 грн.	0 грн.	0 грн.
2	Платня ведучого	7000 грн.	0 грн.	0 грн.
3	Створення плану заходу	8000 грн.	0 грн.	0 грн.
4	Музична підтримка	7000 грн.	0 грн.	0 грн.
5	Виступ танцювального балету	4000 грн.	0 грн.	0 грн.

[Додати статтю витрат](#)

9

Рисунок В.9 – Слайд № 9

Відображення статті витрат

EventCostsPlan

Добрий день, Олександр Рудченко [Вийти](#)

[Заходи](#)
[Шаблони](#)
[Порівняння](#)
[Пошук](#)
[Групи](#)
[Профіль](#)

Весілля Івана та Марії Шинкаренів

Стаття витрат: Оренда автобусів для гостей
Група статей: Оренда транспорту
Запланований бюджет: 10000 грн.
Фактичний бюджет: 9500 грн.
Обчислений бюджет: 7500 грн.
Особливості: Необхідно орендувати великогабаритний автобус та мікроавтобус з їх світловим прикрашенням на 4 години офіційної частини.

[Редагувати](#) [Нова](#)

Відповісти

Витрати за статтею

#	Назва	Сума	Користувач	Дата
1	Передплата за оренду автобуса ISUZU A0962	2000 грн.	egello	8.06.2022
2	Передплата за оренду мікроавтобуса Mercedes-Benz Sprinter 316	500 грн.	egello	8.06.2022
3	Внесення повної плати за оренду автобуса ISUZU A0962	5000 грн.	egello	9.06.2022

[Додати витрату](#)

Коментарі

Віталій Зоряний

07.06.2022 15:02

Ми не зможемо обійтися одним великим автобусом, потрібен додатково один маленький.

10

Рисунок В.10 – Слайд № 10

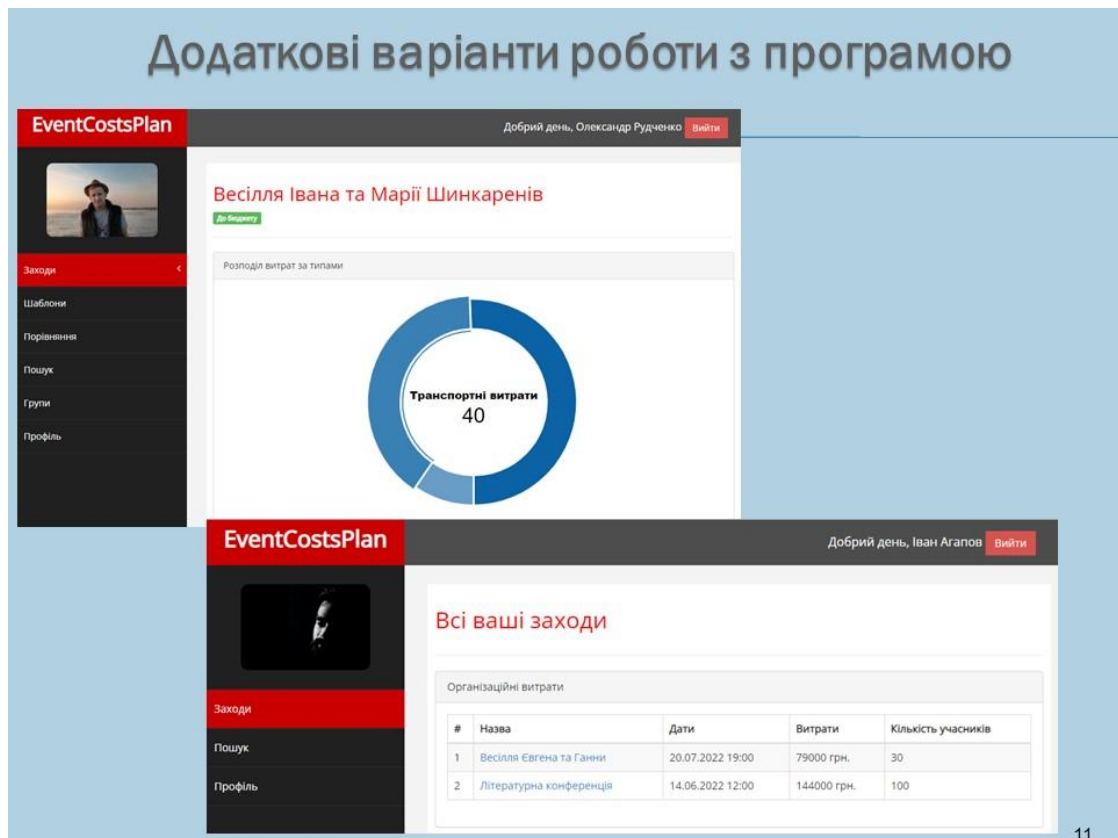


Рисунок В.11 – Слайд № 11

Висновки

При виконанні роботи забезпечено повний цикл розробки програми, починаючи від вивчення предметної області з формулюванням вимог до програми на основі цього вивчення, охоплюючи вибір мови програмування Python, фреймворку веброботки Django та системи керування БД PostgreSQL для підтримки реалізації програми, проєктування і створення БД, визначення структури програми, завершуючи реалізацією вебсистеми і на основі цієї структури та базовим тестування цієї системи.

Створена програма може використовуватися для підтримки роботи над плануванням витрат при організації заходів, надаючи для цього не тільки можливості внесення планів і фактичних витрат, але і коментування, голосування за спірними рішеннями щодо статей витрат, а також розділення доступу різних працівників до даних процесів, в тому числі перегляду бюджетів, редагування даних про захід загалом, внесення бюджету заходу, що в результаті розширює зручність її застосування для планування витрат за заходами.

Рисунок В.12 – Слайд № 12