Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Комп'ютерна система зберігання та обробки файлів

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-4
Анастасія ПРОВОТОРОВА
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Антон СОРОКІН
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Провоторовой Анастасії Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи Комп'ютерна система зберігання та обробки файлів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи
обробка даних
технологія SaaS
зберігання данних

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної області та постановка завдання
Розробка програмних засобів підтримки баз даних на основі технології SAAS
Реалізація програмного комплексу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 12 слайдів

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	26.05.2025–30.05.2025	
2	Вибір технології розробки та інструменталь-	31.05.2025–03.06.2025	
	них засобів	04.06.2025–06.06.2025	
3	Розробка КС	07.06.2025–08.06.2025	
4	Відлагодження КС	09.06.2025–11.06.2025	
5	Аналіз результатів	12.06.2025–13.06.2025	
6	Оформлення ПЗ	14.06.2025–16.06.2025	

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Антон СОРОКІН
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 63 с., 14 рис., 2 дод., 6 джерел.

УПРАВЛІННЯ ЧАСОМ, SAAS, ВЕБ-СИСТЕМА, АНАЛІТИКА АКТИВНОСТІ, УПРАВЛІННЯ ФАЙЛАМИ, ВІЗУАЛІЗАЦІЯ ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС.

Метою кваліфікаційної роботи є розробка комп’ютерної системи для зберігання та обробки файлів статистичних даних, що стосуються управління особистим часом, із використанням технології SaaS.

У ході виконання кваліфікаційної роботи Створено комп’ютерну систему для збереження та аналізу статистичних даних, пов’язаних з управлінням особистим часом на основі технології SaaS. Розроблена платформа характеризується високим рівнем інтеграції функціональних модулів, поєднуючи засоби для ведення активностей, аналітики, керування файлами та адміністрування в єдиному середовищі. Архітектура системи підтримує масштабованість і дозволяє легко додавати нову функціональність без істотних змін у програмному коді. Аналітичний модуль забезпечує візуальне представлення даних у формі кругових діаграм і звітів, що надає користувачеві змогу ефективно аналізувати власну активність та приймати обґрунтовані рішення щодо розподілу часу.

ABSTRACT

Bachelor’s thesis: 63 pages, 14 figures, 2 appendices, 6 sources.

TIME MANAGEMENT, SAAS, WEB SYSTEM, ACTIVITY ANALYTICS, FILE MANAGEMENT, DATA VISUALIZATION, USER INTERFACE.

The major goal of this thesis is the development of a computer system for storing and processing statistical data files related to personal time management, utilizing SaaS technology.

In order to a computer system was developed to facilitate the storage and analysis of statistical data associated with personal time management, based on the Software-as-a-Service (SaaS) model. The implemented platform is characterized by a high level of functional integration, combining modules for activity tracking, data analytics, file management, and administrative control within a unified environment. The system’s architecture supports scalability and allows for seamless extension of functionality without significant modifications to the core codebase. The analytical module provides comprehensive data visualization through dynamically generated pie charts and statistical reports, enabling users to gain clear insights into their activity structure and make informed decisions regarding time allocation.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – програмний інтерфейс додатків

CRUD – створити, прочитати, оновити, видалити

CSS – каскадні таблиці стилів

ERD – діаграма сутність-зв'язок

HTML – мова розмітки гіпертексту

HTTP – протокол передачі гіпертексту

ID – ідентифікатор

JS – мова програмування JavaScript

JSON – нотація об'єктів JavaScript

KB – кілобайт

MB – мегабайт

MySQL – система управління базами даних

PHP – препроцесор гіпертексту

SaaS – програмне забезпечення як послуга

SQL – структурована мова запитів

СУБД – система управління базами даних

SVG – масштабована векторна графіка

TM – управління часом

UML – уніфікована мова моделювання

URL – уніфікований локатор ресурсів

XML – розширювана мова розмітки

ВСТУП

У сучасних умовах інтенсивного інформаційного навантаження, коли обсяг даних постійно зростає, виникає потреба у створенні спеціалізованих сховищ для збереження та систематизації інформації різного функціонального призначення. Актуальні інформаційні системи демонструють високу здатність до накопичення великих масивів даних, мають складну архітектуру та відповідають різноманітним запитам користувачів. Базовим елементом таких систем виступають бази даних, які забезпечують структуроване подання інформації про об’єкти реального світу у контексті конкретної предметної галузі. Вони є центральною ланкою процесу обробки інформації та забезпечують логічне й фізичне зберігання відомостей, доступних для колективного використання.

З позиції концептуального підходу, база даних трактується як централізоване сховище взаємопов’язаних даних, організоване таким чином, щоб забезпечити зручний доступ і ефективне управління інформацією. Засоби управління базами даних дозволяють інтегрувати інформацію з різноманітних джерел, підтримуючи створення єдиної реляційної структури, яка здатна оперативно обслуговувати запити, формувати аналітичні звіти, здійснювати пошук і фільтрацію даних, а також реалізовувати механізми їх візуалізації та друку. Такі інструменти підвищують ефективність аналізу інформації та прийняття рішень у різних сферах діяльності.

Кваліфікаційна робота спрямована на проєктування інформаційної системи, що орієнтована на обробку статистичних відомостей, пов’язаних з організацією та моніторингом використання часу, із застосуванням моделі надання програмного забезпечення як послуги (SaaS). Основна функціональність системи полягає у можливості планування вільного часу користувачем, а також у наданні засобів для рефлексивного аналізу минулих дій з метою оцінки доцільності витраченого часу. Реалізація бази даних

виконана з використанням технологій MySQL, тоді як клієнтська частина системи розроблена з використанням мови PHP, що забезпечує гнучкість у розгортанні та масштабуванні програмного рішення.

Мета цієї роботи полягає у розробці комп'ютерної системи для обробки статистичних даних, що стосуються управління особистим часом, із використанням технології SaaS. Така система має забезпечити зручне планування вільного часу користувачем та дозволити здійснювати ретроспективний аналіз ефективності його використання.

Завдання, які випливають з поставленої мети, включають:

- розробку архітектури інформаційної системи з підтримкою SaaS-моделі;
- проєктування та реалізацію реляційної бази даних для зберігання статистичних відомостей;
- створення клієнтської частини програми на основі PHP для забезпечення інтерактивної взаємодії з користувачем;
- реалізацію функціоналу, що дозволяє вносити, аналізувати та візуалізувати дані про використання часу;
- забезпечення зручного доступу до сервісу для кінцевих користувачів через веб-інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Модель програмного забезпечення як послуги

У сучасному інформаційному середовищі модель програмного забезпечення як послуги (SaaS) [1] часто розглядається не лише як технічна концепція, а й як окрема бізнес-модель. Проте її суть значно глибша, ніж просто періодизована оренда цифрового продукту. SaaS передбачає принципово новий підхід до споживання програмного забезпечення, за якого користувач взаємодіє з функціоналом не через локальну інсталяцію, а шляхом віддаленого доступу до централізованого ресурсу[2].

Рисунок 1.1 – Технологія SaaS

На рисунку 1.1 представлено технологію SaaS.

У цьому контексті сервіс Gmail є найпоширенішим прикладом такого підходу, оскільки він надає поштову функціональність без потреби самостійного налаштування серверів, інсталяції поштових агентів чи підтримки технічної інфраструктури. У той час як корпоративні структури досі часто покладаються на локальні рішення з підтримкою системного адміністратора, більшість кінцевих користувачів вже віддали перевагу хмарним платформам, які забезпечують надійність, масштабованість, безпеку та простоту використання [3,4].

Суттєве поширення веб-інтерфейсів замість традиційних десктопних поштових клієнтів стало ілюстрацією зсуву у парадигмі використання програмних продуктів. Така тенденція стосується не лише електронної пошти, а й інших сегментів інформаційних технологій. Популярність хмарних рішень спостерігається також у середовищі корпоративного документообігу, систем управління проєктами, CRM, ERP, а також веб-додатків для колективної роботи, як-от Google Docs. Вони забезпечують ефективну організацію взаємодії між користувачами, спрощують процеси редагування і синхронізації та зменшують витрати на підтримку інфраструктури [4].

Комерційні платформи, зокрема Google, Microsoft, Zoho, а також низка інших постачальників, розширюють функціональність SaaS-продуктів у галузі корпоративної пошти, планування задач, електронної комерції та контент-менеджменту. Серед прикладів можна назвати такі рішення, як Basecamp, PinnacleCart, Magento, SugarCRM, Open-Xchange тощо. Кожен із них реалізує специфічні інструменти у формі веб-сервісу, що спрощує їхню інтеграцію в бізнес-процеси підприємств різного масштабу [4; 6].

Технологія SaaS надає низку переваг, які стосуються не лише зручності використання, а й оптимізації витрат. Відсутність необхідності в інсталяції, технічному обслуговуванні, ліцензуванні, а також спрощення оновлення програмного забезпечення є ключовими факторами її популярності. Більше

того, така модель дозволяє розробникам забезпечити високий рівень контролю над своїми продуктами, водночас мінімізуючи ризики несанкціонованого копіювання та використання [5].

Попри наявні переваги, існують і суттєві обмеження. Серед них – залежність від стабільності мережевого з'єднання, технічна відсталість браузерних додатків порівняно з повнофункціональними десктопними системами, а також побоювання користувачів щодо безпеки обробки конфіденційної інформації у віддалених середовищах. Особливе занепокоєння викликає фрагментарність SaaS-пропозицій, яка часто не дозволяє інтегрувати окремі модулі у єдину інформаційну систему без додаткових витрат на адаптацію та налаштування взаємодії [5].

Крім технічних труднощів, існує також проблема маркетингової невиразності SaaS-продуктів. Часто розробники концентруються на функціональних можливостях окремого компонента, не враховуючи загального контексту бізнес-процесів користувача. Така вузькість призводить до ситуації, коли запропоновані рішення не відповідають реальним потребам, що, у свою чергу, знижує їхню комерційну привабливість. Відсутність можливості кастомізації функціоналу також ускладнює впровадження SaaS у специфічні корпоративні середовища, де вимагається індивідуальна адаптація під конкретну модель діяльності [5].

Водночас, технологія SaaS має значний інноваційний потенціал, особливо у частині автоматизованого обслуговування, масштабованості, доступності з будь-якої точки світу та можливості реалізації інтегрованих платформ на базі єдиної інфраструктури. Поступовий розвиток веб-технологій, розширення функціональних можливостей браузерів та зростання довіри користувачів створюють сприятливі передумови для подальшого зміцнення позицій моделі SaaS як ключового елементу цифрової трансформації [4–6].

1.2 Технології створення web-застосунків

Серед базових технологій, що використовуються для реалізації динамічних веб-застосунків, важливе місце займає Common Gateway Interface (CGI) – інтерфейс, який забезпечує можливість запуску серверних програм у відповідь на HTTP-запити, що надходять через URL. Ці програми здатні обробляти вхідні дані, що містяться у заголовках запиту або в його тілі, залежно від специфіки протоколу, після чого генерують HTML-код, який передається на сторону клієнта. Історично поняття «CGI-скрипт» асоціювалося з використанням скриптових мов, зокрема Perl, проте такі програми не виконуються у браузері, а на сервері, під контролем відповідної операційної системи. Варто зазначити, що розробка CGI-програм може здійснюватися з використанням будь-якої технології, яка дозволяє створювати консольні додатки для серверного середовища.

Іншою ключовою технологією є HTML (HyperText Markup Language), яка виступає стандартом розмітки для документів у глобальному мережевому середовищі. Веб-сторінки, що створюються за допомогою HTML або XHTML, обробляються браузерами та виводяться у формі, зручній для сприйняття користувачем. За допомогою HTML визначається структура документа, включно з заголовками, абзацами, таблицями, списками, а також вставляються інтерактивні елементи, мультимедійні об'єкти та гіперпосилання. Синтаксис мови базується на теговій розмітці, де кожен елемент обмежується початковим і, за потреби, завершальним тегом. Деякі елементи є самодостатніми та не потребують завершального маркера. Додаткові властивості задаються за допомогою атрибутів, які розміщуються у відкриваючому тезі. На відміну від XHTML, HTML є нечутливим до регістру символів у назвах тегів і атрибутів. Структура HTML-документа також дозволяє вкладеність елементів і підтримує використання сутностей – спеціальних символів, представлених у вигляді кодів Unicode, які починаються із символу амперсанда.

У структурі мови SQL ключовою операцією є формування запитів, що реалізується за допомогою конструкції SELECT-FROM-WHERE, яка відображає процес вилучення даних із бази. Ця конструкція відповідає двом базовим операціям реляційної алгебри – селекції та проекції. На першому етапі виконується горизонтальна фільтрація записів відповідно до заданого предиката, що дозволяє обрати лише ті рядки, які задовольняють зазначені умови. Після цього здійснюється вертикальне обмеження вибірки, тобто відбір окремих атрибутів, які підлягають подальшому опрацюванню чи виводу.

Синтаксична форма простого запиту до однієї таблиці передбачає використання інструкції SELECT, яка може бути доповнена ключовими словами DISTINCT або ALL для керування унікальністю результатів. Після цього вказується перелік цільових полів, які підлягають вилученню, назва таблиці, з якої здійснюється вибірка, та умова фільтрації через конструкцію WHERE. Порядок полів у запиті визначає послідовність виводу результатів.

Для введення нових записів до таблиці використовується команда INSERT, яка має дві основні форми. Перша з них дозволяє вставити один рядок даних до вказаної таблиці. Синтаксис цієї форми передбачає зазначення назви таблиці, списку стовпців, до яких вставляються значення, та відповідного набору даних у секції VALUES. Кількість та порядок значень мають повністю відповідати заданому списку полів.

Операція оновлення наявних даних реалізується за допомогою команди UPDATE, яка дозволяє змінювати значення у вибраних рядках відповідно до заданих умов. Видалення записів з таблиці здійснюється через команду DELETE, яка також може бути доповнена предикатом для обмеження дії операції лише певними умовами. Ці механізми забезпечують базовий функціонал маніпулювання даними в межах систем управління реляційними базами даних.

1.3 Проєктування системи керування базою даних

У сучасних умовах економічного розвитку спостерігається стрімке зростання обсягів інформації, що потребують зберігання та обробки. Традиційні методи, зокрема ручне опрацювання, виявляються непридатними для таких задач, що обумовлює необхідність використання автоматизованих систем зберігання даних у вигляді баз даних. Прикладне програмне забезпечення, як правило, репрезентує формалізований опис фрагментів реального світу, при цьому обробка великих обсягів даних реалізується поза межами логіки самої програми. Зазвичай дані розміщуються автономно та структуруються у вигляді спеціалізованих сховищ – баз даних.

Починаючи з другої половини XX століття, для підтримки роботи з даними стали широко застосовуватись системи керування базами даних (СКБД), що виконують функції фізичного розміщення інформації, її опису, пошуку, актуалізації, захисту від несанкціонованого доступу та організації одночасної роботи з кількома запитами. Крім того, сучасні СКБД мають підтримувати мовні засоби визначення даних, а також інтерпретатори запитів на модифікацію, збереження та вилучення інформації. Такі запити можуть мати як попередньо визначений, так і непередбачуваний характер, що вимагає від СКБД адаптивності до різних сценаріїв доступу до даних.

База даних репрезентує структуру, яка забезпечує організоване зберігання несуперечливої, цілісної та мінімально надлишкової інформації, пов’язаної з певною предметною галуззю. Обсяг деталізації даних у базі визначається як вимогами до подальшої обробки, так і складністю процесів у межах відповідної сфери діяльності. Архітектура БД може реалізовуватись у різних формах, включаючи локальні рішення, модель файл-сервер, класичну клієнт-серверну організацію, а також багаторівневі (n-tier) системи.

У межах даної інформаційної системи реалізовано трирівневу архітектуру, яка передбачає відокремлення прикладної логіки від рівня доступу до даних. Такий підхід дозволяє делегувати функції обробки

проміжному програмному рівню, який може виконуватись на спеціалізованих або звичайних веб-серверах. Керування даними здійснюється на стороні окремого серверу бази даних. Взаємодія користувача з системою реалізується через веб-браузер, що дозволяє уникнути потреби в інсталяції додаткового ПЗ. Водночас користувачеві надається інтерфейс у вигляді форм, які завантажуються разом із вебсторінками та ініціюють запити до серверної частини системи.

Уся архітектура включає клієнтську частину (термінал), сервер додатків, що реалізує прикладну логіку, та сервер БД, що забезпечує збереження й доступ до даних. Термінал виконує роль інтерфейсного рівня, не має прямого доступу до БД і відповідає за базову логіку, включаючи авторизацію, валідацію введених даних, а також просту обробку на кшталт сортування. Сервер додатків, як проміжний рівень, забезпечує основну логіку бізнес-процесів, а сервер бази даних зосереджує функції фізичного зберігання та обробки запитів до реляційної чи об'єктно-орієнтованої СКБД.

У мінімальній конфігурації сервер додатків і сервер бази даних можуть функціонувати на одному хост-комп’ютері, тоді як повноцінна реалізація вимагає розділення на фізично незалежні вузли, що забезпечує підвищений рівень безпеки, відмовостійкості та масштабованості. Серед переваг такої архітектури варто відзначити її масштабованість, гнучкість конфігурації, високий рівень безпеки, ефективність при обмежених характеристиках клієнтських пристроїв і низьку залежність від швидкості каналу між терміналами та сервером додатків. Водночас вона ускладнює процес розгортання, потребує серйозного адміністрування, а також високої продуктивності серверного обладнання.

Процес проєктування бази даних, у свою чергу, розгортається у кілька етапів. Першим є концептуальне проєктування, яке передбачає побудову логічної моделі предметної області на основі аналізу вимог користувача. У результаті формується абстрактна модель, що описує сутності, їхні властивості та взаємозв’язки, які мають ключове значення для досягнення

цілей системного моделювання.

1.4 Моделювання предметної області

Процес моделювання предметної області є фундаментальним етапом у розробці програмних систем, оскільки саме на цьому етапі відбувається формалізація знань про реальні об'єкти та процеси, які підлягають автоматизації. Для реалізації цього процесу сучасна практика програмної інженерії пропонує широкий спектр спеціалізованих інструментальних засобів, зокрема CASE-систем, таких як Rational Rose, CA BPwin, Silverrun, Sybase тощо. У межах даної роботи моделювання предметної області здійснювалося із застосуванням уніфікованої мови моделювання UML, яка є галузевим стандартом для візуального опису систем на різних етапах її життєвого циклу.

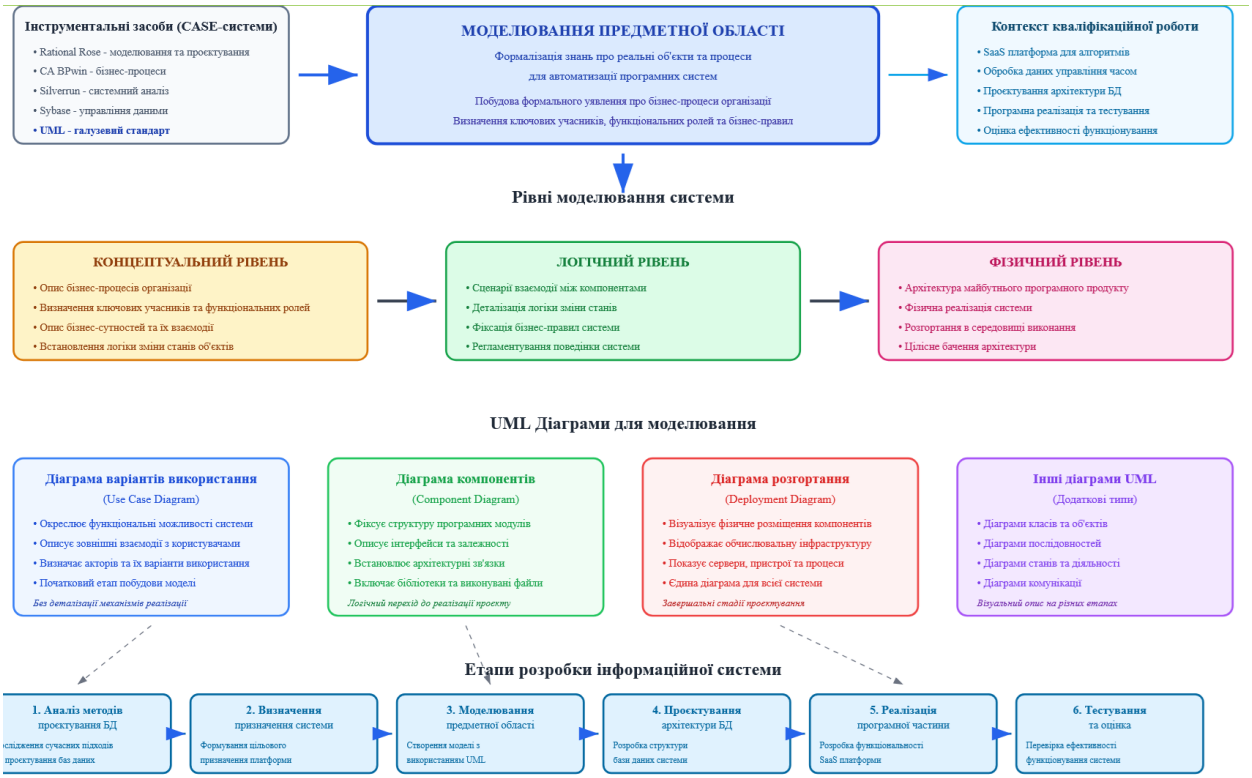

Рисунок 1.2 – Процес моделювання предметної області

На рисунку 1.2 представлен процес моделювання предметної області. Початковим етапом побудови моделі є створення діаграми варіантів

використання, яка окреслює функціональні можливості системи, описуючи зовнішні взаємодії між нею та користувачами або іншими суб'єктами – так званими акторами. Актором вважається будь-яка зовнішня сутність, що ініціює або отримує функціональні сервіси від системи. Варіанти використання, у свою чергу, описують окремі функції, які система виконує у відповідь на запити акторів, без деталізації механізмів реалізації таких функцій.

Основне призначення моделювання предметної області полягає у побудові формального уявлення про бізнес-процеси організації, визначенні їх ключових учасників та функціональних ролей, описі бізнес-сутностей і сценаріїв їх взаємодії, встановленні логіки зміни станів об'єктів і фіксації бізнес-правил, що регламентують поведінку системи. Візуальне моделювання в UML дозволяє послідовно перейти від концептуального опису системи до логічного і далі до фізичного рівня реалізації. Такий підхід забезпечує цілісне бачення архітектури майбутнього програмного продукту.

На завершальних стадіях проєктування використовується діаграма розгортання, яка візуалізує фізичне розміщення програмних компонентів у середовищі виконання. Вона відображає об'єкти обчислювальної інфраструктури, такі як сервери, пристрої та процеси, а також логічні зв’язки між ними. Ця діаграма є єдиною для всієї системи, оскільки репрезентує загальну архітектурну модель її розгортання. Елементи вихідного коду не включаються до такої діаграми, оскільки вона призначена для моделювання лише тих компонентів, що активні на етапі виконання.

У контексті фізичного представлення системи використовуються також діаграми компонентів, які дозволяють зафіксувати структуру програмних модулів, описати їхні інтерфейси та залежності між ними. Такі діаграми дають змогу встановити архітектурні зв’язки між фрагментами коду, а також забезпечують логічний перехід до реалізації проєкту. На діаграмі компонентів можуть бути відображені як програмні бібліотеки, так і

виконувані файли. Вони використовуються для візуалізації структури коду, його повторного використання, формалізації фізичних моделей баз даних, а також для погодження логіки взаємодії компонентів на етапі компіляції чи виконання. Розробка таких діаграм передбачає участь архітекторів, аналітиків і програмістів, що забезпечує узгодженість між логічним моделюванням і фізичною реалізацією системи.

У межах цієї кваліфікаційної роботи передбачено створення інформаційної системи для моделювання алгоритмів обробки даних, що пов’язані з управлінням часом, із використанням принципів технології SaaS. Задля досягнення поставленої мети необхідно було здійснити аналіз сучасних методів проєктування баз даних, визначити цільове призначення системи, сформувати модель предметної області, спроєктувати архітектуру бази даних, реалізувати програмну частину системи та провести її тестування з метою оцінки ефективності функціонування.

2 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ПІДТРИМКИ БАЗ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ SAAS

2.1 Призначення системи розробки

У даній роботі розглянуто функціональні та архітектурні особливості інформаційної системи під назвою «ТМ», яка реалізована з урахуванням принципів технології SaaS. Система створена з метою надати користувачам інструмент для організації особистого часу, а також для аналізу ефективності його використання. В умовах сучасного темпу життя, коли повсякденна активність часто не залишає простору для рефлексії та планування, індивід нерідко не усвідомлює, як саме розподіляє свій час і наскільки раціонально він його витрачає. Саме ці проблеми стали передумовою для розробки системи «ТМ», яка надає можливість відстежувати витрати часу за різними видами діяльності, оцінювати їх доцільність та формувати звички ефективного тайм-менеджменту.

Запропонована система дозволяє користувачу отримувати аналітичну інформацію про структуру власної активності, визначати пріоритети, а також знаходити баланс між роботою та відпочинком. В результаті її використання користувач може не лише планувати майбутні дії, але й обґрунтовано коригувати попередню поведінку, орієнтуючись на дані про минулі витрати часу. У цьому аспекті система «ТМ» виступає як інструмент самоменеджменту та цифрової підтримки прийняття особистих рішень.

2.2 Вимоги до даних та функціональності

Розроблена інформаційна система передбачає функціональну можливість для користувачів фіксувати власні дані, пов’язані з розпорядком дня, а також отримувати результати обробки цієї інформації у візуалізованій

формі, зокрема у вигляді кругових діаграм. Для забезпечення ефективної взаємодії між користувачем та системою було передбачено створення клієнтського застосунку, який виконує роль зручного інтерфейсу доступу до сервісів системи. Такий підхід дозволяє індивідуалізувати досвід користувача, полегшуючи виконання базових операцій і спрощуючи доступ до інструментів керування часом.

Функціональність системи охоплює процеси управління обліковими записами, включаючи реєстрацію нових користувачів, автентифікацію як звичайних користувачів, так і адміністраторів, а також механізми редагування профільної інформації та можливість блокування доступу. Крім того, передбачено можливість управління переліком видів діяльності, до якого користувачі можуть додавати нові категорії. Система здійснює фіксацію даних про витрачений час, реалізує алгоритми обробки зібраної інформації та формує її графічне подання. Усе це забезпечує повноцінну реалізацію функції аналізу часової активності, що є основною метою даної системи.

2.3 Бізнес-правила до системи "ТМ"

Будь-яка база даних підпорядковується низці структурних та функціональних вимог, таких як забезпечення унікальності значень атрибутів або підтримання логічної узгодженості між пов’язаними таблицями. Втім, власне логіка, яка регламентує ці вимоги з точки зору бізнес-логіки, визначається через систему бізнес-правил. У загальному розумінні бізнес-правила становлять собою формалізовані інструкції, що регулюють допустимі стани даних і поведінку інформаційної системи відповідно до специфіки предметної області.

Виділяють три ключові типи бізнес-правил. Перший тип – це правила виведення, які забезпечують перетворення вхідних даних у результат, що може бути використаний у подальших обчисленнях або перевірках. Такі

правила передбачають логіку трансформації значень і повинні бути чітко визначені в процесі проєктування. Другий тип – правила обмеження, які верифікують відповідність операцій або транзакцій установленим умовам. У разі порушення таких умов, система має блокувати виконання відповідної операції. Третій тип – інваріантні правила, які гарантують цілісність і узгодженість даних у результаті змін, забезпечуючи стабільність функціонування системи в цілому.

У контексті розробки системи «ТМ», що реалізована на основі технології SaaS, сформульовано низку специфічних бізнес-правил, які відображають організаційні ролі користувачів, їхні права доступу та допустимі дії в межах інформаційної системи. Зокрема, всі користувачі класифікуються на незареєстрованих відвідувачів, зареєстрованих користувачів і адміністраторів. Незареєстровані користувачі мають обмежений доступ до функціоналу, зокрема, вони можуть ознайомлюватися з інформаційними розділами, але не мають змоги працювати з інструментами статистичного аналізу. Зареєстровані користувачі, після проходження автентифікації, отримують доступ до особистих даних, можуть керувати видами діяльності, переглядати історію активності та аналізувати її у вигляді графічного представлення за обраний період – день, тиждень або місяць.

Адміністративна роль в системі передбачає розширені повноваження, включаючи модифікацію інформаційного наповнення, управління обліковими записами користувачів, блокування або видалення облікових записів, а також редагування категорій і переліку видів діяльності. Кожен вид діяльності структурно характеризується датою, назвою, часом початку і завершення, що дозволяє точно фіксувати часові інтервали.

Додатково передбачена можливість зміни пароля авторизованим користувачем, а також обрання часових проміжків для побудови статистичних звітів. Таким чином, система бізнес-правил у «ТМ» є цілісною логічною моделлю, що регламентує поведінку користувачів, забезпечує узгоджене функціонування підсистем і дозволяє реалізувати обґрунтовану

політику доступу в рамках SaaS-архітектури.

2.4 Проектування структури бази даних

Перед початком розробки структури бази даних важливо сформулювати глосарій термінів, що відображає специфіку предметної області. Глосарій виступає як систематизований словник вузькоспеціалізованих понять, що використовуються у межах конкретного проєкту, і слугує засобом уніфікації термінології, що забезпечує коректну взаємодію між членами проєктної команди, замовником і кінцевим користувачем. У глосарії відображено ключові ролі системи та логічні сутності, які становлять основу моделі варіантів використання.

Основними суб’єктами, що взаємодіють із системою, є користувачі, які залежно від рівня доступу поділяються на незареєстрованих і зареєстрованих. Незареєстровані користувачі можуть ознайомлюватися лише з відкритим вмістом, зокрема статтями та новинами, не маючи доступу до функціональності аналітичної обробки. Натомість зареєстровані користувачі, після проходження автентифікації, отримують змогу використовувати інструменти для аналізу персонального розпорядку часу. Особливу роль у системі відіграє адміністратор, який володіє розширеними повноваженнями з управління користувачами, редагування інформаційного наповнення та структурних елементів системи.

До основних інформаційних об’єктів системи належать статті, що є публікаціями тематичного характеру, та види діяльності, які користувач вносить для подальшої статистичної обробки. Ці терміни формують концептуальний каркас системи, на основі якого виконується подальше моделювання.

Наступним кроком у процесі проєктування є побудова діаграми варіантів використання (Use Case Diagram), яка фіксує функціональне призначення системи, відображаючи очікувану поведінку та сценарії

взаємодії між системою та зовнішніми акторами. Така діаграма виступає концептуальним поданням архітектури системи, що слугує відправною точкою для подальшої деталізації логічної та фізичної моделі.

На основі сформованої функціональної моделі доцільно здійснити побудову діаграми розгортання (Deployment Diagram), що ілюструє фізичну інфраструктуру виконання програмних компонентів системи. Ця діаграма демонструє архітектуру у середовищі виконання, описуючи взаємозв'язки між окремими пристроями, процесами та сервісами. Приклад такої візуалізації наведено на рисунку 2.1, який слугує ілюстрацією взаємодії системних модулів під час роботи системи.

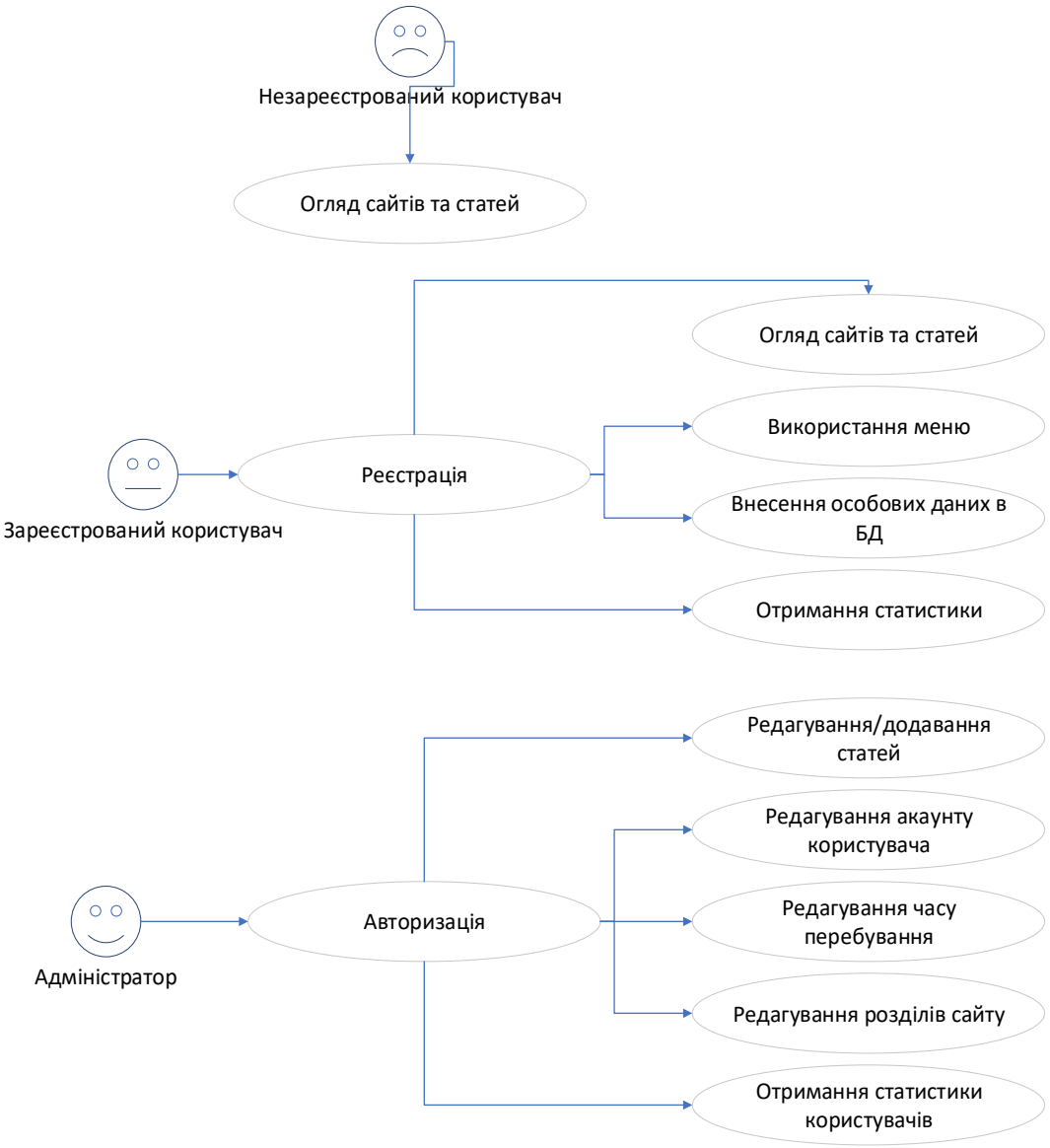

Рисунок 2.1 – Діаграма варіантів використання предметної області

База даних, яка забезпечує функціонування розробленої інформаційної системи, має складну структуру та включає 39 сутностей. Такий обсяг пояснюється необхідністю підтримки широкого функціонального спектру системи, що охоплює як адміністративні модулі, так і сервіси взаємодії з кінцевими користувачами. У подальшому аналізі розглядатимуться лише окремі сутності, які є ключовими для розуміння основної логіки системи.

До загальної структури бази даних входять такі сутності: jos_banner, jos_bannerclient, jos_bannertrack, jos_categories, jos_components, jos_contact_details, jos_content, jos_content_frontpage, jos_content_rating, jos_core_acl_aro_map, jos_core_log_searches, jos_groups, jos_jce_groups, jos_jce_plugins, jos_menu, jos_menu_types, jos_messages, jos_messages_cfg, jos_ssssss, jos_core_acl_aro, jos_core_acl_aro_groups, jos_core_acl_aro_sections, jos_core_acl_groups_aro_map, jos_core_log_items та інші.

Ці сутності відображають структурну розгалуженість інформаційної системи та її здатність обробляти різноманітні типи даних, включаючи контент, повідомлення, елементи меню, елементи керування доступом і логування дій користувачів. Надалі буде здійснено детальний аналіз лише окремих сутностей, що безпосередньо беруть участь у реалізації ключових функцій системи.

З метою формування цілісного уявлення про структуру та взаємозв’язки між сутностями, що складають базу даних, доцільним є побудова концептуальної моделі типу «сутність–зв’язок». Візуальне представлення цієї моделі наведено на рисунку 2.4. Модель дозволяє відобразити об’єкти предметної області та логічні відносини між ними, що у сукупності формують основу структури бази даних.

Кожна сутність у моделі має набір властивостей, які характеризують її суттєві параметри. Об’єкти одного типу, як правило, володіють однаковим набором атрибутів. Залежно від природи інформації, що зберігається, властивості можуть набувати різних типів. Зокрема, розрізняють прості властивості, які мають єдине значення, та складові, що складаються з кількох

підвластивостей. Деякі з них можуть виконувати роль ключових, тобто забезпечувати унікальність записів у межах певного контексту. Також властивості можуть бути однозначними або допускати множинні значення, залежно від моделі відображення реального світу.

У процесі побудови моделей можливою є ситуація, коли значення певної властивості відсутнє – через невизначеність або недоречність її застосування до конкретного екземпляра сутності. Крім того, важливо враховувати розрізнення між базовими властивостями, які зберігаються у базі даних безпосередньо, та похідними, що обчислюються на основі інших атрибутів. Такий підхід дозволяє забезпечити цілісність і гнучкість структури даних при її реалізації та експлуатації.

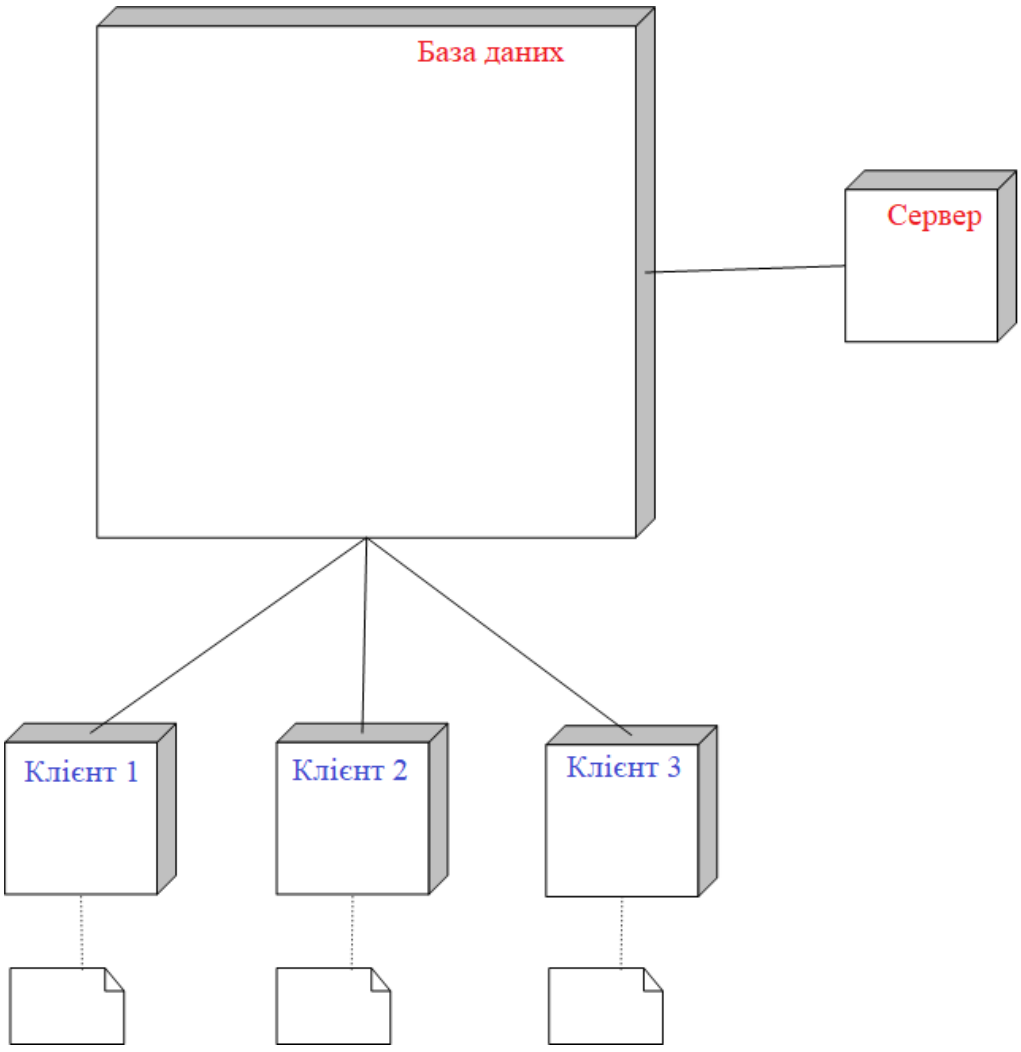

Рисунок 2.2 – Діаграма розгортання

На рисунку 2.2 візуалізовано типову клієнт-серверну архітектуру доступу до бази даних у межах інформаційної системи. Центральним елементом є база даних, яка з’єднана із сервером, що відповідає за обробку запитів, забезпечення бізнес-логіки та управління транзакціями. До бази даних підключаються кілька клієнтських пристроїв, кожен з яких взаємодіє з центральним сховищем через відповідний інтерфейс.

Архітектура передбачає централізоване зберігання інформації, що дозволяє клієнтам отримувати, змінювати та передавати дані за допомогою серверного посередника. Така модель сприяє зменшенню надмірності даних, забезпечує узгодженість інформації та централізоване управління доступом. Усі дії клієнтів опосередковуються сервером, який забезпечує підключення, автентифікацію та подальшу обробку запитів до бази даних, забезпечуючи високий рівень безпеки, масштабованості та контролю.

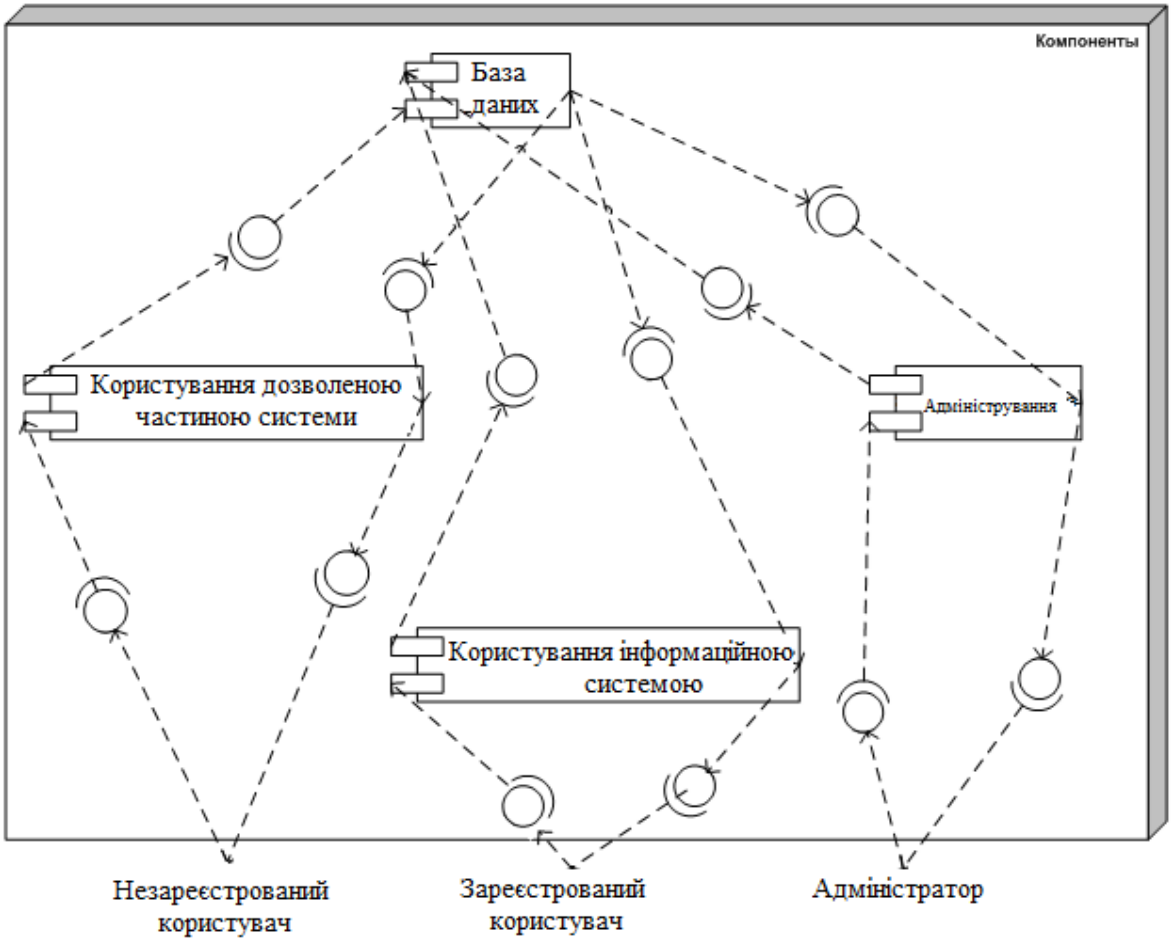

Рисунок 2.3 – Діаграма компонентів

На рисунку 3.3 подано діаграму компонентів, яка відображає логіку функціонального поділу системи на окремі модулі залежно від ролі користувача. У межах архітектури розглядаються три типи суб’єктів: незареєстрований користувач, зареєстрований користувач та адміністратор. Кожен із них взаємодіє з відповідними компонентами інформаційної системи через визначений інтерфейс доступу.

Компонент "Користування дозволеною частиною системи" обслуговує незареєстрованих користувачів, забезпечуючи обмежений набір функцій. Компонент "Користування інформаційною системою" відкриває розширені можливості для зареєстрованих користувачів, які взаємодіють із системою в рамках персоналізованого інтерфейсу. Компонент "Адміністрування" реалізує функціональність, призначену для адміністраторів, надаючи можливість керування вмістом, доступами та іншими елементами системи. Взаємозв’язок усіх компонентів з базою даних забезпечує цілісність, централізоване зберігання даних та синхронізацію дій користувачів відповідно до їхніх повноважень.

Тип сутності	Тип зв’язку	Тип сутності	Кардинальність
Користувач	Зв’язана	Заняття	N:M
Користувач	Зв’язана	Сесія	1:01
Користувач	Зв’язана	Контакт	1:01
Користувач	Зв’язана	Контент	1:01

Рисунок 2.4 – Структурований опис даних

На рисунку 2.4 представлено структурований опис даних, що використовуються для обчислення статистичних показників, пов’язаних із результатами змагань. Наведена інформація відображає формат зберігання

ключових атрибутів, необхідних для подальшої аналітичної обробки. Такий опис дозволяє однозначно ідентифікувати кожен параметр, а також забезпечує формальну основу для реалізації процедур статистичного аналізу в межах інформаційної системи.

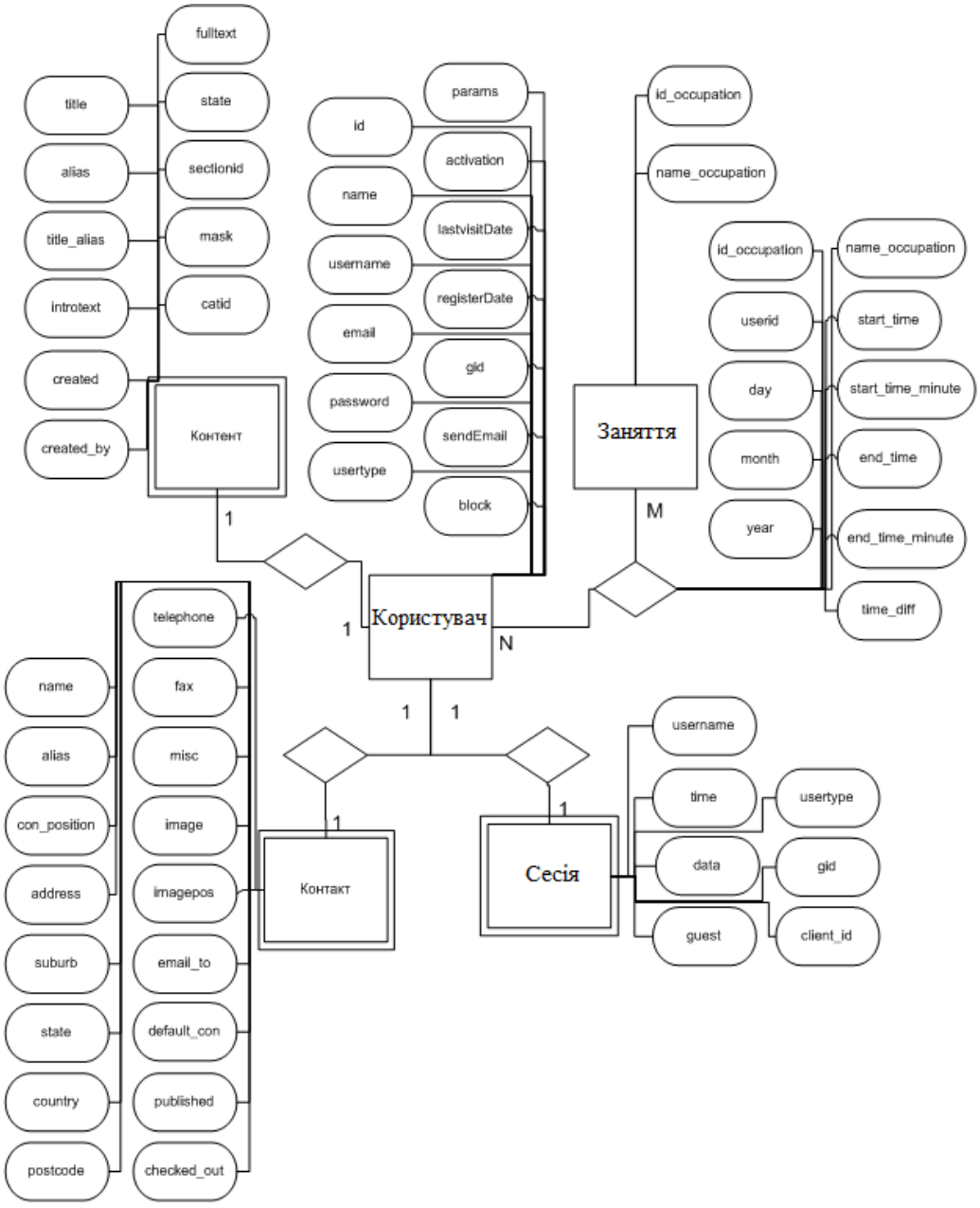

Рисунок 2.5 – Концептуальна модель даних. Діаграма сутність-зв’язок в термінах Чена

На рисунку 2.5 представлено концептуальну модель даних типу «сутність-зв’язок», яка описує основну структуру інформаційної системи

управління часом із використанням технології SaaS. Модель візуалізує ключові сутності системи, їх атрибути, а також логічні зв’язки між ними.

Центральною сутністю є користувач, що пов’язаний із декількома іншими сутностями, які забезпечують реалізацію функціональності системи. Зокрема, встановлено взаємозв’язки з контактними даними користувача, інформацією про відвідування системи, збереженим контентом і реєстром виконаних видів діяльності. Сутність, що відповідає за облік занять, містить інформацію про часові характеристики подій, пов’язаних із конкретними користувачами, включаючи дату, час початку та завершення дії, а також обчислену тривалість.

Кожна сутність детально описується через відповідні атрибути, які забезпечують зберігання повної інформації, необхідної для обробки даних. Зв’язки між сутностями мають кардинальність, що визначає кількісні обмеження взаємодії, зокрема зв’язок один до багатьох між користувачами та заняттями. Така структура забезпечує гнучкість у представленні даних, дозволяє виконувати аналітичну обробку і підтримує масштабованість у межах багатокористувацького середовища.

Після видалення зв'язків типу M:N отримаємо логічну модель.

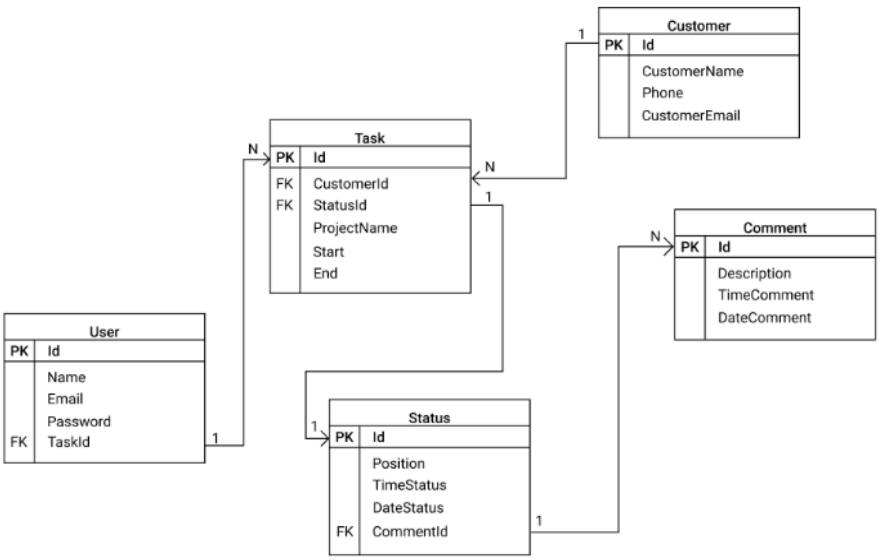

Рисунок 2.6 – Схема БД

На рисунку 2.6 представлено логічну модель бази даних у вигляді діаграми «сутність-зв’язок», яка описує структуру системи керування завданнями з урахуванням користувачів, клієнтів, коментарів і статусів проєктів. Діаграма побудована з використанням нотації ERD і включає основні таблиці з первинними та зовнішніми ключами, що забезпечують узгодженість даних і підтримку цілісності зв’язків між таблицями.

Центральною сутністю є таблиця Task, яка зберігає дані про конкретні завдання, включаючи їхній початок, завершення, назву проєкту та посилання на відповідного клієнта і статус. Завдання пов’язано з таблицею Customer, яка містить контактну інформацію замовника, а також з таблицею Status, яка описує поточний стан завдання в часовому вимірі.

Таблиця User зберігає інформацію про користувачів системи, які можуть бути пов’язані з певним завданням. Таблиця Comment забезпечує можливість зберігання коментарів до завдань, із зазначенням часу та дати публікації, що дозволяє відстежувати історію обговорень і змін у межах кожного проєкту.

Модель підтримує зв’язки типу один до багатьох і багато до одного, що дозволяє ефективно реалізовувати функціональність відстеження проєктів, їхнього статусу та взаємодії між користувачами і клієнтами. Така структура бази даних сприяє масштабованості, аналітичному аналізу й адаптації до складних бізнес-процесів.

Логічна модель даних, побудована за синтаксисом UML та розроблена у середовищі DBDesigner, представлена на рисунку 2.7. Ця модель відображає структуру інформаційної системи, зокрема основні сутності, їх атрибути, а також типи та напрями зв’язків між ними. Вона є ключовим етапом у процесі проєктування бази даних, оскільки дозволяє формалізувати інформаційні потреби системи у вигляді об’єктно-реляційної схеми.

Кожна сутність в UML-нотації представлена прямокутником, який містить назву таблиці та перелік атрибутів, серед яких виділено первинні ключі та зовнішні ключі, що забезпечують логічні зв’язки між таблицями.

Кардинальності між об’єктами визначаються числовими позначеннями на кінцях асоціацій і вказують на допустиму кількість екземплярів, що беруть участь у зв’язку. Таким чином, модель охоплює структуру управління завданнями, взаємодію з клієнтами, контроль статусів і коментування.

Застосування UML-нотації у поєднанні з функціоналом DBDesigner дає змогу створити формалізовану, логічно зв'язану структуру, яка забезпечує узгодженість даних, ефективне керування інформаційними потоками та подальшу реалізацію фізичної моделі бази даних.

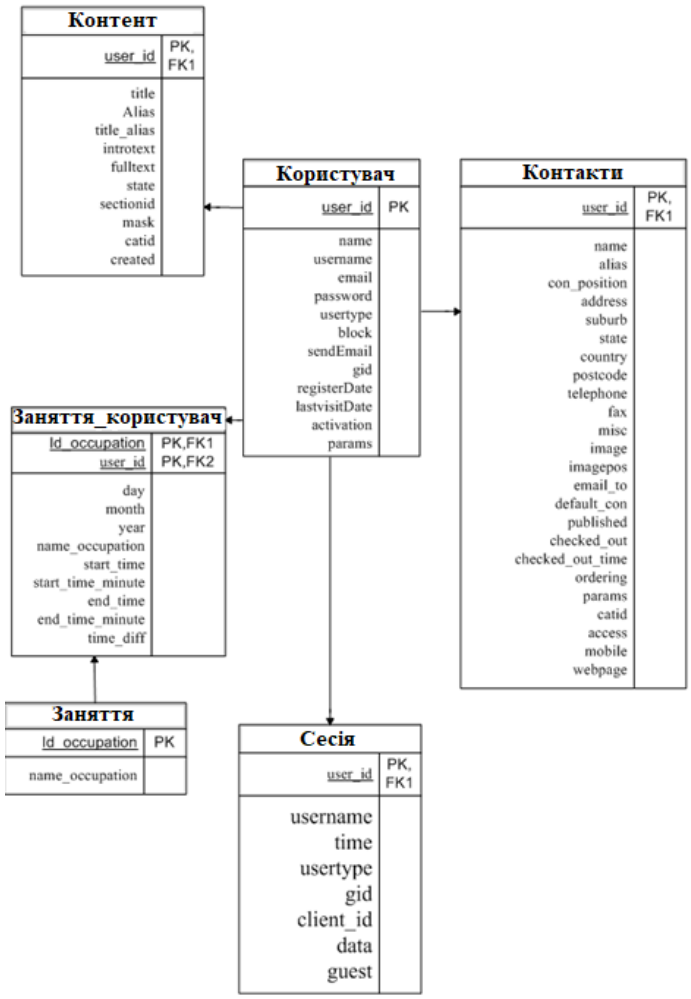

Рисунок 2.7 – Логічна модель даних з синтаксисом UML

Логічна модель даних за синтаксисом UML, розроблена в DBDesigner, представлена на рисунку 2.7.

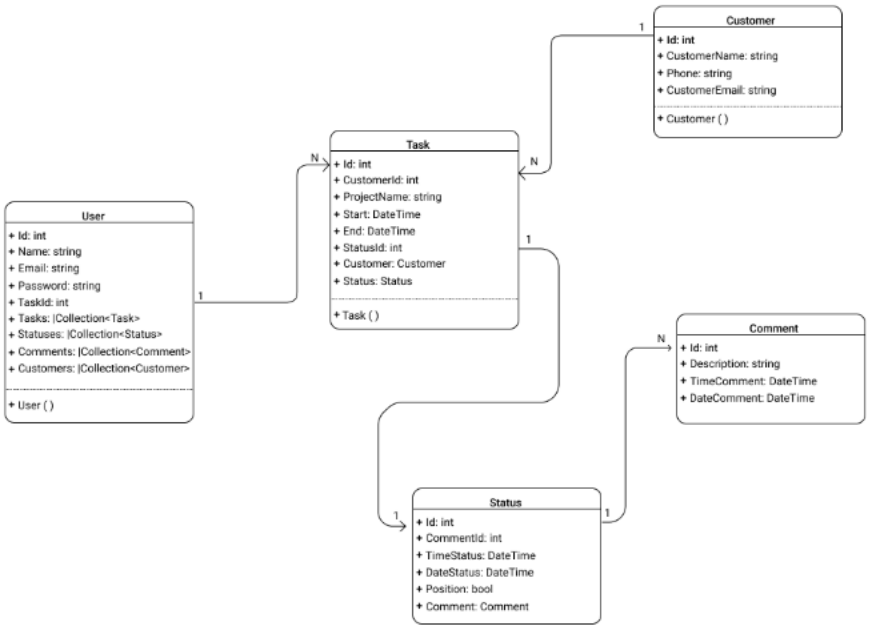

Рисунок 2.7 – Фізична модель даних

Фізична модель бази даних для заданої предметної області, представлена на рисунку 2.8, відображає реалізацію логічної структури у вигляді конкретних таблиць, полів, типів даних і зв’язків, які фізично зберігаються у системі управління базами даних. На відміну від логічної моделі, що зосереджується на структурі даних та їх взаємозв’язках, фізична модель деталізує технічні аспекти зберігання – визначає типи стовпців, обмеження цілісності, індекси та правила доступу до даних.

У цій моделі кожна сутність реалізується у вигляді таблиці з конкретними типами атрибутів, що відповідають характеристикам реальних даних. Первинні ключі встановлюються для забезпечення унікальності записів, а зовнішні ключі – для підтримки цілісності зв’язків між таблицями. Окрім цього, можуть бути вказані обмеження на нульові значення, значення за замовчуванням, автоінкрементні поля тощо.

Фізична модель слугує основою для створення SQL-скриптів, що використовуються для генерації структури бази даних у реальному середовищі, наприклад, у MySQL або PostgreSQL. Саме вона визначає, як

дані фактично зберігатимуться, оброблятимуться й оновлюватимуться, і є ключовим елементом на етапі імплементації інформаційної системи.

3 РЕАЛІЗАЦІЯ КОМП’ЮТЕРНОЇ СИСТЕМИ

Розроблена комп'ютерна система управління часом "TM" представляє собою комплексну інформаційну платформу, побудовану на основі технології SaaS, що забезпечує користувачам можливість ефективного відстеження, аналізу та оптимізації використання особистого часу. Система характеризується багаторівневою архітектурою, яка включає презентаційний рівень, бізнес-логіку та рівень даних, що забезпечує масштабованість та надійність функціонування в умовах багатокористувацького середовища.

Презентаційний рівень системи реалізовано у вигляді веб-орієнтованого клієнтського застосунку, який надає інтуїтивно зрозумілий інтерфейс для взаємодії з функціональними можливостями платформи. Інтерфейс користувача побудовано за принципами адаптивного дизайну, що забезпечує коректне відображення на різних типах пристроїв та роздільностях екрану. Навігаційна структура системи організована у вигляді бокової панелі з основними функціональними розділами, включаючи панель управління, модуль активностей, аналітичну підсистему та адміністративний інтерфейс.

Модуль автентифікації та авторизації становить критично важливий компонент системи, що забезпечує диференційований доступ до функціональних можливостей залежно від ролі користувача. Система підтримує три основні ролі: незареєстровані відвідувачі з обмеженим доступом до інформаційних ресурсів, зареєстровані користувачі з повним доступом до персональних інструментів управління часом, та адміністратори з розширеними повноваженнями щодо управління системою та користувачами. Механізм авторизації реалізовано з урахуванням принципів безпеки та забезпечує захист персональних даних користувачів.

Підсистема управління активностями являє собою центральний функціональний блок, що дозволяє користувачам фіксувати інформацію про

власні заняття з детальним зазначенням часових характеристик. Кожна активність характеризується набором атрибутів, включаючи назву, дату проведення, час початку та завершення, а також автоматично обчислену тривалість. Система забезпечує можливість додавання нових активностей через інтерактивну форму з валідацією введених даних, що гарантує цілісність та коректність інформації.

Аналітичний модуль реалізує комплексну обробку зібраних даних з генерацією візуальних представлень у вигляді кругових діаграм та статистичних звітів. Алгоритми аналізу забезпечують розрахунок ключових показників ефективності, включаючи загальний час активності, розподіл часу за категоріями діяльності, середню тривалість занять та процентний розподіл активностей. Візуалізація даних реалізована з використанням векторних графічних елементів, що забезпечує високу якість відображення та інтерактивність діаграм.

Адміністративна підсистема надає розширені можливості для управління системою, включаючи модерацію користувачів, редагування контенту та конфігурацію системних параметрів. Цей компонент забезпечує централізоване управління інформаційними ресурсами, підтримку цілісності даних та моніторинг функціонування системи. Адміністративний інтерфейс структуровано за функціональними блоками, що забезпечує ефективну навігацію та виконання управлінських завдань.

Система зберігання даних побудована на основі реляційної моделі з чітко визначеною структурою сутностей та їх взаємозв'язків. Архітектура бази даних включає основні таблиці для зберігання інформації про користувачів, активності, категорії діяльності та системні налаштування. Забезпечення цілісності даних досягається через систему обмежень, первинних та зовнішніх ключів, що гарантує надійність зберігання та узгодженість інформації.

Компонент бізнес-логіки реалізує набір правил та алгоритмів, що регламентують поведінку системи відповідно до специфіки предметної

області. Цей рівень забезпечує валідацію даних, обробку бізнес-процесів, розрахунок аналітичних показників та координацію взаємодії між різними модулями системи. Архітектура бізнес-логіки побудована за принципами модульності та масштабованості, що дозволяє ефективно адаптувати систему до змінних вимог користувачів.

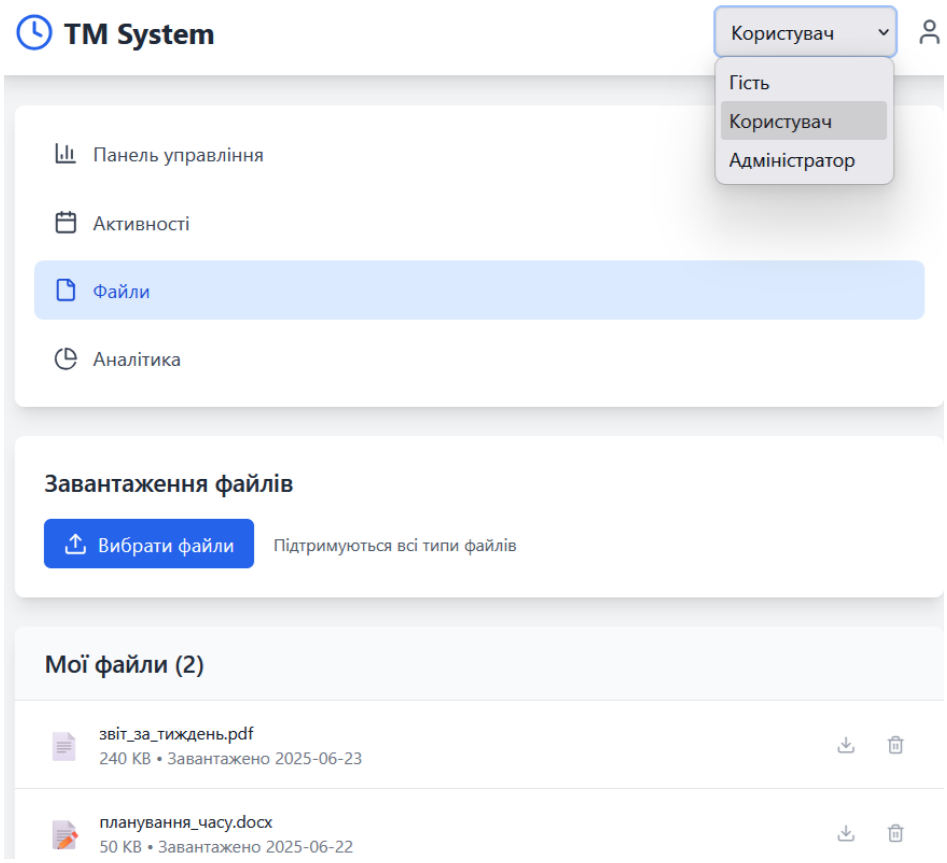

Рисунок 3.1 – Інтерфейс

На рисунку 3.1 представлено фрагмент інтерфейсу користувача системи управління часом "TM", що демонструє навігаційну структуру та функціональні елементи програмного застосунку. Зображення відображає верхню частину веб-інтерфейсу з логотипом системи, розташованим у лівому верхньому куті, та випадаючим меню вибору ролі користувача у правій частині заголовка, яке містить опції для гостя, звичайного користувача та адміністратора.

Основну частину інтерфейсу займає бічна навігаційна панель з трьома

основними розділами системи, серед яких активним є розділ "Активності", що візуально виділено синім кольором. Нижче навігаційної панелі розміщено функціональний блок для додавання нової активності з відповідним полем введення назви активності, що демонструє інтерактивний характер інтерфейсу та можливості для введення даних користувачем.

Графічний дизайн інтерфейсу виконано в мінімалістичному стилі з використанням світлої колірної схеми та чіткої типографіки, що забезпечує високу читабельність та зручність використання. Структура меню та розташування елементів відповідає сучасним принципам проектування користувацьких інтерфейсів, орієнтованим на інтуїтивну навігацію та ефективну взаємодію з системою.

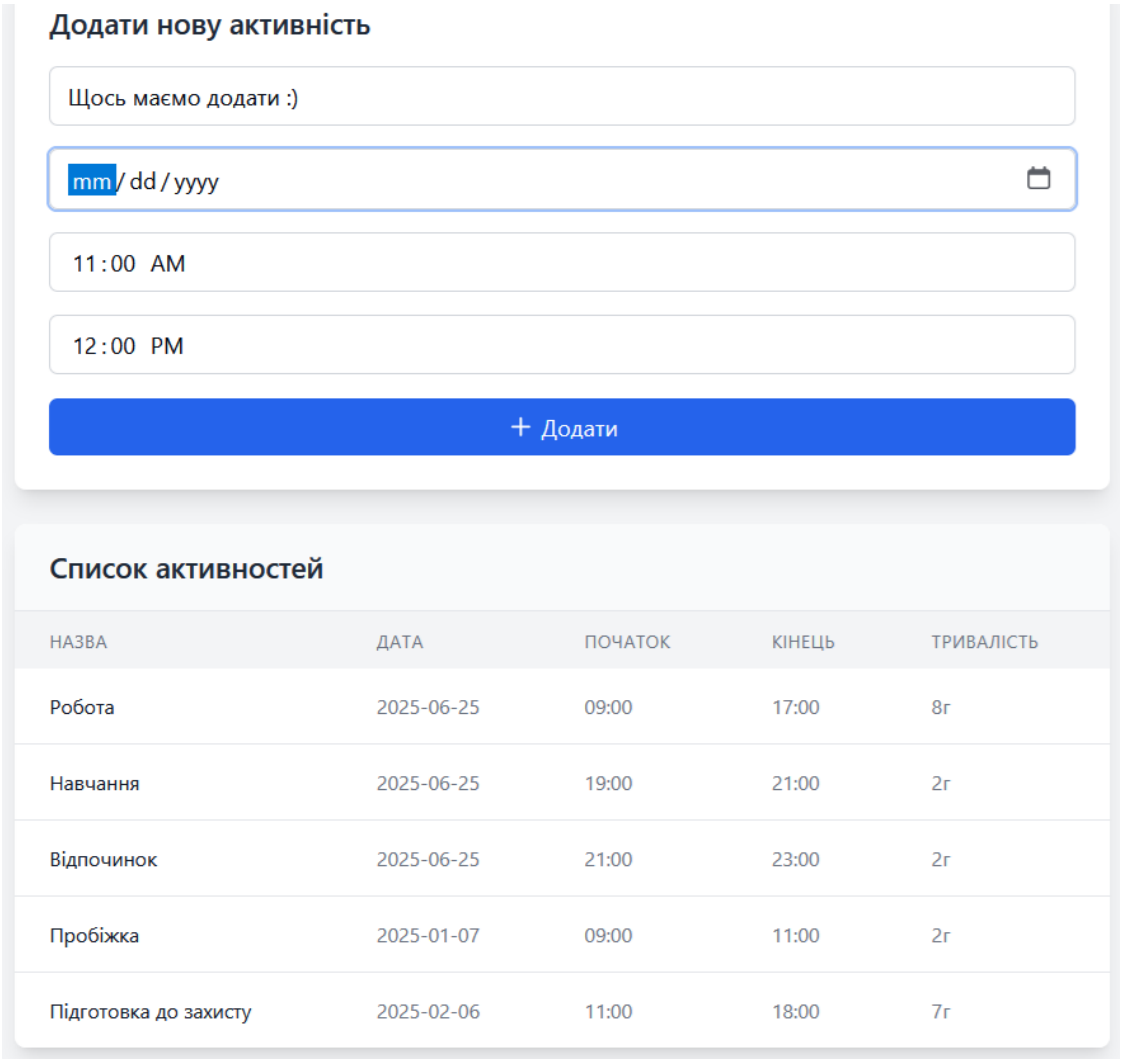

Рисунок 3.2 – Інтерфейс

На рисунку 3.2 представлено інтерфейс модуля управління активностями системи "ТМ", що складається з двох основних функціональних блоків. Верхня частина містить форму для введення нової активності з полями для назви заняття, дати, часу початку та завершення, завершену синьою кнопкою додавання. Нижня частина демонструє табличне представлення списку зареєстрованих активностей з колонками для назви, дати, часових меж та обчисленої тривалості, що ілюструює можливості системи щодо зберігання та відображення персональних даних користувача про розподіл часу.

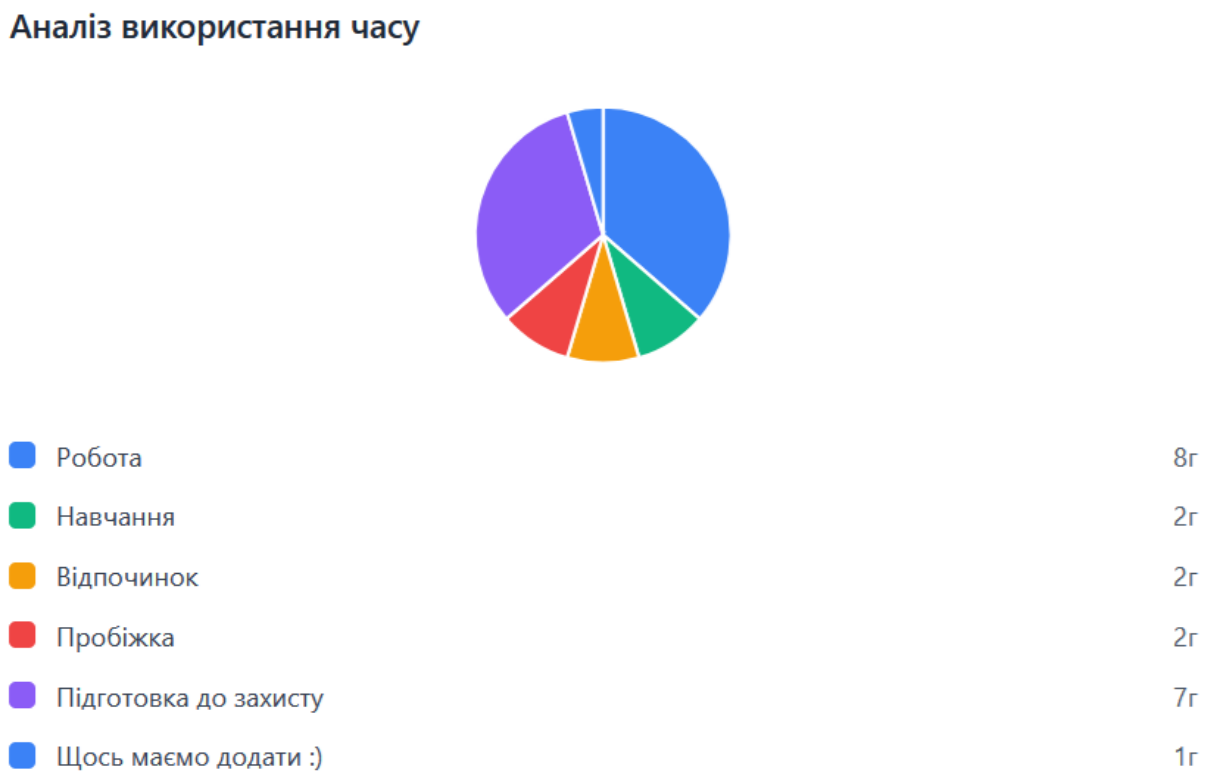

Рисунок 3.3 – Інтерфейс

На рисунку 3.3 представлено інтерфейс аналітичного модуля системи "ТМ", що демонструє візуалізацію розподілу часу користувача у вигляді кругової діаграми з кольоровим кодуванням різних видів діяльності. Діаграма супроводжується легендою, яка відображає відповідність кольорових сегментів конкретним активностям та їх тривалість у годинах, що

забезпечує наочне представлення структури використання часу та дозволяє користувачу аналізувати ефективність розподілу власної активності протягом певного періоду.

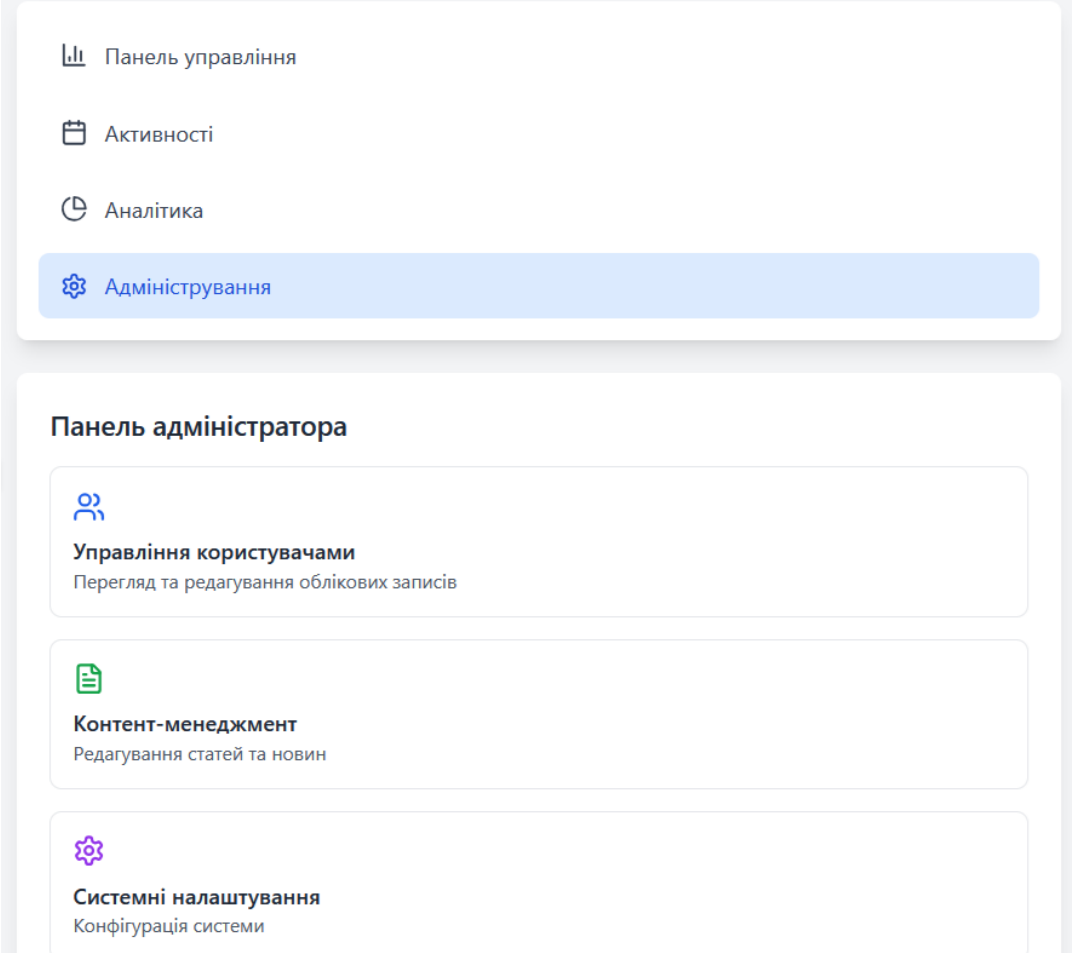

Рисунок 3.4 – Адміністративний інтерфейс системи "ТМ"

На рисунку 3.4 представлено адміністративний інтерфейс системи "ТМ", що включає навігаційну панель з виділеним розділом "Адміністрування" та основну робочу область з панеллю адміністратора. Функціональний блок містить три категорії управлінських інструментів: управління користувачами для редагування облікових записів, контент-менеджмент для роботи зі статтями та новинами, а також системні налаштування для конфігурації платформи, що демонструє розширені можливості адміністративного рівня доступу в архітектурі системи.

ВИСНОВКИ

Розроблено комп’ютерну систему для зберігання та обробки файлів статистичних даних, що стосуються управління особистим з використанням SaaS технологій. Розроблена система демонструє високий рівень функціональної інтеграції, об'єднуючи модулі управління активностями, аналітичної обробки даних, управління файлами та адміністративного керування в єдину цілісну платформу.

Реалізована архітектура забезпечує масштабованість системи та можливість подальшого розширення функціональності без суттєвих змін у базовій структурі програмного коду. Аналітичний компонент системи реалізує комплексну візуалізацію даних через динамічне генерування кругових діаграм та статистичних звітів, що дозволяє користувачам отримувати наочне представлення про структуру власної активності та приймати обґрунтовані рішення щодо оптимізації розподілу часу. Алгоритми обчислення ключових показників ефективності забезпечують точність аналітичних розрахунків та релевантність генерованих рекомендацій.

Модуль управління файлами розширює функціональні можливості системи, надаючи користувачам інструменти для централізованого зберігання та організації документів, пов'язаних із процесами планування та контролю часу. Інтеграція файлового менеджера з основною аналітичною системою створює синергетичний ефект, підвищуючи загальну ефективність робочих процесів користувачів. Система безпеки та розмежування доступу реалізована через багаторівневу модель авторизації, що забезпечує захист персональних даних користувачів та адміністративних функцій системи. Диференційований підхід до надання привілеїв користувачам різних ролей гарантує відповідність принципам інформаційної безпеки та мінімізує ризики несанкціонованого доступу до критично важливих компонентів системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Избачков Ю. С. Інформаційні системи: підручник [Текст]: 2-е вид. СПб: Пітер, 2008. 656 с.

2. Гайдамакін Н. А. Автоматизовані системи, бази і банки даних. Вступний курс [Текст]: Навчальний посібник. М.: Геліос АРВ, 2002. 368с.

3. Матюхін В. О., Огірко І. В. «Інформаційні системи і технології у сфері сільського туризму». Навчальна програма дисципліни для бакалаврів, спеціалістів. К.: МАУП, 2004. 16 с.

4. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. Х.: ХНАМГ, 2010. 222 с. Бібо, Б. JQuery. Докладне керівництво з просунутого JavaScript [Текст] /Б.Бібо. СПБ: Наука, 2010. 327с.

5. Колисниченко, Д.А. Програмування на PHP [Текст] / Д.А. Колисниченко. Санкт - Петербург: БВХ Петербург, 2011. 399с.

6. Дейт, К. Вступ до систем баз даних [Текст]/ К. Дейт. – М.: Наука, 1980. – 464 с. Мартін, Г. SQL: Довідкове керівництво [Текст]/ Г. Мартін. – М.: Лорі, 2000. – 291 с.