

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____ Ігор БОЛБОТ
(підпис)

«__» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ Белла ГОЛУБ
(підпис)

«__» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмне забезпечення інформаційної системи оренди спортивних споруд»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
К.Т.Н., доцент

_____ Ганна Вайганг
(підпис)

Керівник бакалаврської кваліфікаційної роботи

_____ Георгій Бородакін
(підпис)

Виконав

_____ Данило Литвиненко
(підпис)

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

К.Т.Н., доцент _____ Белла ГОЛУБ
(підпис)

«__» _____ 2025 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Литвиненку Данилі Сергійовичу

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення інформаційної системи оренди спортивних споруд»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

методичні рекомендації фахівців щодо утримання спортивних споруд і об'єктів.

Перелік питань, що підлягають дослідженню:

1. Провести аналіз предметної області.
2. Дослідити стан існуючих аналогів.
3. Спроектувати інформаційне забезпечення системи.
4. Реалізувати мобільний застосунок.
5. Провести тестування системи.
6. Підготувати рекомендації щодо впровадження застосунку.

Перелік графічних документів (за необхідності).

Дата видачі завдання «27» грудня 2025 р.

**Керівник бакалаврської
кваліфікаційної роботи**

_____ (підпис)

Георгій Бородкін

**Завдання взяв до
виконання**

_____ (підпис)

Данило Литвиненко

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	5
ВСТУП.....	6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ	8
1.1 Характеристика предметної області оренди спортивних споруд	8
1.2. Аналіз сучасних інформаційних систем і вебсервісів бронювання	10
1.3. Порівняльний аналіз існуючих аналогів	12
1.4. Визначення функціональних та нефункціональних вимог до системи....	13
1.5. Постановка задачі на розроблення інформаційної системи оренди спортивних споруд	14
1.6 Висновки до Розділу 1	15
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	17
2.1. Обґрунтування вибору архітектури програмного забезпечення.....	17
2.2. Моделювання бізнес-процесів системи оренди спортивних споруд.....	18
2.3. Проєктування функціональної структури системи	19
2.4. Розробка UML-діаграм інформаційної системи	23
2.5. Проєктування бази даних системи.....	27
2.6. Проєктування користувацького інтерфейсу вебзастосунку	31
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	35
3.1. Вибір засобів та технологій розроблення	35
3.2. Реалізація серверної частини системи	37
3.3. Реалізація клієнтської частини вебзастосунку	40
3.4. Розроблення модуля реєстрації та авторизації користувачів.....	45
3.5. Розроблення модуля перегляду спортивних споруд та бронювання	47
3.6. Розроблення адміністративного модуля керування системою	51
3.7. Інтеграція програмних модулів з базою даних.....	54
РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ.....	57
4.1. Організація та методика тестування інформаційної системи	57
4.2. Перевірка функціональних можливостей вебзастосунку	59

	4
4.3. Аналіз результатів тестування інформаційної системи	61
4.4. Оцінка зручності використання та практичної ефективності системи	62
4.5. Рекомендації щодо подальшого вдосконалення системи	65
ВИСНОВКИ	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТКИ.....	73

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface, програмний інтерфейс взаємодії веб-сервісів і модулів системи.

CPU - Central Processing Unit, центральний процесор серверного або локального пристрою.

CSV - Comma-Separated Values, формат експорту табличних даних моніторингу.

HTML - HyperText Markup Language, мова розмітки веб-панелі та звітів системи.

HTTP(S) - HyperText Transfer Protocol Secure, протокол захищеного доступу до веб-контролю.

ORM - Object-Relational Mapping, засіб взаємодії програмних об'єктів із реляційною БД.

PDF - Portable Document Format, формат збереження звітів користувача.

RAM - Random Access Memory, оперативна пам'ять пристрою або сервера.

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій важливим напрямом є автоматизація процесів управління різними сферами діяльності, зокрема сферою оренди спортивних споруд. Зростання попиту на фізичну активність, розвиток масового спорту та збільшення кількості спортивних об'єктів обумовлюють необхідність створення ефективних інструментів для організації їх використання.

Традиційні способи бронювання спортивних споруд, що передбачають телефонні дзвінки або особисті звернення, є малоефективними, потребують значних витрат часу та не забезпечують зручного контролю за розкладом зайнятості об'єктів. У зв'язку з цим актуальним є розроблення інформаційних систем, які дозволяють автоматизувати процес пошуку, бронювання та управління орендою спортивних споруд.

Інформаційна система оренди спортивних споруд забезпечує можливість користувачам швидко знаходити необхідні об'єкти, переглядати їх доступність, здійснювати бронювання та керувати власними замовленнями. Для адміністрації така система надає інструменти контролю, обліку та аналізу використання ресурсів.

Актуальність теми бакалаврської роботи полягає у необхідності створення сучасного програмного забезпечення, яке підвищує ефективність управління спортивними об'єктами, покращує зручність для користувачів та сприяє оптимізації процесів оренди.

Об'єктом дослідження є процеси організації оренди спортивних споруд.

Предметом дослідження є методи та засоби розроблення програмного забезпечення інформаційної системи оренди спортивних споруд.

Метою роботи є розроблення програмного забезпечення інформаційної системи оренди спортивних споруд, яка забезпечує зручний доступ до інформації про об'єкти, можливість їх бронювання та ефективне управління процесами оренди.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область оренди спортивних споруд;
- дослідити існуючі інформаційні системи та вебсервіси;
- сформулювати вимоги до інформаційної системи;
- спроектувати структуру та архітектуру системи;
- розробити програмне забезпечення вебзастосунку;
- провести тестування та оцінити ефективність розробленої системи.

Практичне значення отриманих результатів полягає у можливості використання розробленої інформаційної системи для автоматизації процесів оренди спортивних споруд у реальних умовах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика предметної області оренди спортивних споруд

Сфера оренди спортивних споруд є важливою складовою сучасної інфраструктури фізичної культури та спорту. Вона охоплює процеси надання в тимчасове користування різноманітних спортивних об'єктів, зокрема футбольних полів, спортивних залів, тенісних кортів, басейнів та інших спеціалізованих майданчиків.

Основною метою функціонування даної сфери є забезпечення доступу користувачів до спортивних об'єктів для проведення тренувань, змагань або активного відпочинку. Користувачами можуть виступати як окремі особи, так і організації, спортивні команди чи навчальні заклади.

Процес оренди спортивних споруд включає кілька основних етапів: пошук відповідного об'єкта, ознайомлення з його характеристиками, перевірка доступності на певну дату та час, здійснення бронювання та, у разі необхідності, проведення оплати. У свою чергу, власники або адміністратори спортивних споруд здійснюють управління розкладом, контролюють зайнятість об'єктів та забезпечують актуальність інформації.

Особливістю даної предметної області є необхідність врахування часових обмежень, оскільки один і той самий об'єкт може бути орендований різними користувачами у різні часові проміжки. Це потребує точного ведення розкладу та уникнення конфліктів бронювання.

Також важливим аспектом є наявність різних типів користувачів системи, кожен з яких має свої функції та права доступу. Наприклад, звичайні користувачі здійснюють пошук і бронювання, тоді як адміністратори керують об'єктами, редагують інформацію та контролюють процес оренди.

Таким чином, предметна область оренди спортивних споруд характеризується наявністю великої кількості взаємопов'язаних процесів, необхідністю обробки значних обсягів інформації та потребою у зручних і ефективних інструментах управління, що обумовлює доцільність розроблення спеціалізованих інформаційних систем.

Таблиця 1.1

Основні елементи предметної області оренди спортивних споруд

№	Елемент предметної області	Опис
1	Користувач	Особа або організація, яка здійснює пошук, перегляд та бронювання спортивних споруд
2	Адміністратор	Відповідальна особа, яка керує системою, додає та редагує інформацію про спортивні об'єкти, контролює бронювання
3	Спортивна споруда	Об'єкт оренди (спортивний зал, футбольне поле, басейн тощо), який має певні характеристики та доступний для бронювання
4	Бронювання	Процес резервування спортивної споруди на визначений проміжок часу
5	Розклад	Система обліку зайнятості спортивних споруд, яка відображає доступні та зайняті часові інтервали

Наведені у таблиці елементи формують основу предметної області та визначають структуру взаємодії між користувачами і системою. Їх взаємозв'язок забезпечує реалізацію основних функцій інформаційної системи оренди спортивних споруд, зокрема пошуку, бронювання та управління об'єктами.

Таким чином, чітке визначення ключових елементів предметної області дозволяє сформулювати вимоги до системи та є основою для подальшого проєктування програмного забезпечення.

1.2. Аналіз сучасних інформаційних систем і вебсервісів бронювання

У сучасних умовах цифровізації різних сфер діяльності значного поширення набули інформаційні системи та вебсервіси, що забезпечують автоматизацію процесів бронювання. Такі системи застосовуються у різних галузях, зокрема у сфері транспорту, готельного бізнесу, медицини та організації дозвілля, включаючи оренду спортивних споруд.

Основною метою таких систем є спрощення процесу взаємодії між користувачем і постачальником послуг шляхом надання можливості швидкого пошуку, перегляду доступності ресурсів та здійснення бронювання в онлайн-режимі. Використання вебтехнологій дозволяє забезпечити доступ до системи з будь-якого пристрою, що значно підвищує зручність користування.

Серед ключових функцій сучасних систем бронювання можна виділити:

- пошук об'єктів за заданими параметрами;
- перегляд детальної інформації про об'єкти;
- відображення доступності у вигляді розкладу;
- можливість онлайн-бронювання;
- облік замовлень користувачів;
- адміністративне управління системою.

Аналіз сучасних вебсервісів показує, що більшість із них мають схожу функціональну структуру, проте можуть відрізнятися рівнем зручності інтерфейсу, швидкістю роботи, наявністю додаткових функцій та якістю реалізації.

Прикладом таких систем є універсальні платформи бронювання, що дозволяють резервувати різні ресурси, а також спеціалізовані сервіси, орієнтовані на конкретну сферу, зокрема спортивні об'єкти. Вони можуть включати інтеграцію з платіжними системами, календарями, системами сповіщень та іншими сервісами.

Водночас існуючі рішення не завжди повністю задовольняють потреби користувачів, оскільки можуть мати обмежений функціонал, складний інтерфейс або недостатню адаптацію до специфіки оренди спортивних споруд. Це обумовлює необхідність розроблення спеціалізованої інформаційної системи, яка враховує особливості даної предметної області.

Таблиця 1.2

Порівняння сучасних вебсервісів бронювання

№	Назва сервісу	Основні функції	Переваги	Недоліки
1	Універсальні системи бронювання	Пошук об'єктів, онлайн-бронювання, оплата, управління замовленнями	Широкий функціонал, зручність використання, доступність	Не враховують специфіку спортивних споруд
2	Спеціалізовані спортивні сервіси	Бронювання спортивних об'єктів, розклад, управління заняттями	Орієнтація на спорт, наявність тематичних функцій	Обмежений функціонал, не завжди зручний інтерфейс
3	Локальні системи (для клубів)	Ведення розкладу, облік клієнтів, бронювання	Простота, адаптація під конкретний об'єкт	Відсутність масштабованості, обмежений доступ
4	Мобільні додатки бронювання	Швидке бронювання, повідомлення, доступ з телефону	Зручність, мобільність, швидкість доступу	Обмежений функціонал у порівнянні з вебсистемами

Аналіз наведених сервісів свідчить про наявність як універсальних, так і спеціалізованих рішень у сфері бронювання. Проте більшість із них або не враховують особливості оренди спортивних споруд, або мають обмежений функціонал. Це підтверджує доцільність розроблення інформаційної системи, яка поєднує зручність використання, достатній функціонал та орієнтацію на специфіку предметної області.

1.3. Порівняльний аналіз існуючих аналогів

На основі проведеного аналізу сучасних інформаційних систем і вебсервісів бронювання доцільно здійснити їх порівняльний аналіз з метою виявлення переваг і недоліків існуючих рішень, а також визначення напрямів для розроблення власної інформаційної системи.

Існуючі аналоги можна умовно поділити на універсальні системи бронювання, спеціалізовані сервіси для спортивної сфери та локальні рішення, що використовуються окремими організаціями. Кожен із цих типів має свої особливості, які впливають на ефективність їх використання.

Універсальні системи бронювання забезпечують широкий спектр можливостей і можуть застосовуватись у різних сферах діяльності. Вони характеризуються високим рівнем функціональності, зручним інтерфейсом та підтримкою різних типів ресурсів. Однак такі системи, як правило, не враховують специфіку оренди спортивних споруд, що може ускладнювати їх використання в даній сфері.

Спеціалізовані сервіси, орієнтовані на спортивні об'єкти, мають більш адаптований функціонал, який враховує особливості організації тренувального процесу, розкладу занять та бронювання майданчиків. Водночас вони часто мають обмежені можливості масштабування та не завжди забезпечують достатній рівень зручності для користувачів.

Локальні системи, що використовуються окремими спортивними клубами або організаціями, є простими у реалізації та добре адаптованими під конкретні потреби. Проте вони, як правило, не підтримують віддалений доступ, мають обмежений функціонал і не забезпечують інтеграцію з іншими сервісами.

Таким чином, проведений порівняльний аналіз показує, що жоден із розглянутих типів систем не забезпечує повного задоволення вимог користувачів у сфері оренди спортивних споруд. Це обумовлює необхідність створення нової інформаційної системи, яка поєднуватиме переваги існуючих рішень та мінімізуватиме їх недоліки.



Рис. 1.1 – Класифікація інформаційних систем бронювання

Наведена схема відображає основні типи інформаційних систем бронювання, що використовуються в сучасних умовах. Вона демонструє, що кожен тип систем має власні особливості, переваги та обмеження, які необхідно враховувати під час розроблення інформаційної системи оренди спортивних споруд.

1.4. Визначення функціональних та нефункціональних вимог до системи

Визначення вимог до інформаційної системи є одним із ключових етапів її розроблення, оскільки саме вимоги визначають функціональні можливості системи, її структуру та принципи роботи. Для системи оренди спортивних споруд важливо врахувати як функціональні, так і нефункціональні аспекти.

Функціональні вимоги описують дії, які система повинна виконувати, а також взаємодію між користувачем і системою. До основних функціональних вимог інформаційної системи оренди спортивних споруд належать:

- реєстрація та авторизація користувачів у системі;
- перегляд списку доступних спортивних споруд;
- пошук об'єктів за різними параметрами (тип, місцезнаходження, доступність);
- перегляд детальної інформації про спортивну споруду;
- перевірка доступності об'єкта на конкретну дату та час;

- здійснення бронювання спортивної споруди;
- перегляд та керування власними бронюваннями;
- адміністративне керування системою (додавання, редагування та видалення об'єктів);
- управління розкладом спортивних споруд.

Нефункціональні вимоги визначають характеристики якості системи та обмеження, які повинні бути дотримані під час її розроблення та експлуатації. До основних нефункціональних вимог можна віднести:

- зручність використання (інтуїтивно зрозумілий інтерфейс);
- швидкодію системи (оперативна обробка запитів);
- надійність (стабільна робота без збоїв);
- безпека (захист персональних даних користувачів);
- масштабованість (можливість розширення функціоналу);
- доступність (можливість роботи на різних пристроях та у різних браузерах).

Чітке визначення функціональних і нефункціональних вимог дозволяє сформулювати основу для подальшого проєктування інформаційної системи, вибору технологій розроблення та реалізації її програмної частини.

1.5. Постановка задачі на розроблення інформаційної системи оренди спортивних споруд

На основі проведеного аналізу предметної області та існуючих інформаційних систем бронювання можна сформулювати постановку задачі на розроблення інформаційної системи оренди спортивних споруд.

Метою розроблення є створення веборієнтованої інформаційної системи, яка забезпечить автоматизацію процесів пошуку, бронювання та управління спортивними спорудами, а також підвищить зручність взаємодії користувачів із системою.

Основними користувачами системи є зареєстровані користувачі та адміністратори. Користувачі повинні мати можливість переглядати доступні спортивні об'єкти, отримувати інформацію про них, здійснювати бронювання та керувати власними замовленнями. Адміністратори, у свою чергу, повинні мати доступ до функцій керування системою, включаючи додавання нових об'єктів, редагування інформації та контроль за розкладом.

Інформаційна система повинна забезпечувати зберігання даних про користувачів, спортивні споруди, розклад їх використання та здійснені бронювання. Також система повинна підтримувати механізми перевірки доступності об'єктів у реальному часі для запобігання конфліктам бронювання.

До основних задач, які необхідно реалізувати в процесі розроблення системи, належать:

- розроблення структури інформаційної системи;
- проектування бази даних;
- створення зручного користувацького інтерфейсу;
- реалізація функціоналу бронювання спортивних споруд;
- забезпечення авторизації та аутентифікації користувачів;
- реалізація адміністративного модуля керування системою;
- забезпечення надійності та безпеки роботи системи.

У результаті розроблення має бути створений вебзастосунок, який забезпечує ефективну організацію процесу оренди спортивних споруд та відповідає сучасним вимогам до інформаційних систем.

1.6 Висновки до Розділу 1

У першому розділі було проаналізовано предметну область оренди спортивних споруд, визначено її основні елементи та особливості функціонування. Проведено аналіз сучасних інформаційних систем і вебсервісів бронювання, а також виконано їх порівняльний аналіз.

У результаті дослідження встановлено, що існуючі рішення не повністю враховують специфіку оренди спортивних споруд або мають обмежений функціонал. Це обумовлює необхідність розроблення спеціалізованої інформаційної системи.

На основі отриманих результатів було сформульовано функціональні та нефункціональні вимоги до системи, а також здійснено постановку задачі на її розроблення, що є основою для подальшого проектування інформаційної системи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Обґрунтування вибору архітектури програмного забезпечення

Проектування інформаційної системи передбачає вибір оптимальної архітектури програмного забезпечення, яка забезпечить ефективність, надійність та зручність подальшого розвитку системи. Для інформаційної системи оренди спортивних споруд важливо врахувати особливості її функціонування, зокрема необхідність обробки запитів користувачів у реальному часі, зберігання даних та забезпечення доступу до системи через мережу Інтернет.

Серед існуючих архітектурних підходів найбільш поширеними є монолітна архітектура, клієнт-серверна архітектура та мікросервісна архітектура. Кожен із цих підходів має свої переваги та недоліки.

Монолітна архітектура передбачає реалізацію всієї системи у вигляді єдиного програмного продукту. Такий підхід є простим у розробці, однак ускладнює масштабування та супровід системи при її розширенні.

Мікросервісна архітектура, навпаки, забезпечує високий рівень масштабованості та гнучкості, оскільки система складається з незалежних сервісів. Проте її реалізація є складною та потребує значних ресурсів, що є недоцільним для бакалаврського проєкту.

З урахуванням особливостей задачі доцільним є використання клієнт-серверної архітектури. У такій архітектурі система поділяється на дві основні частини: клієнтську та серверну. Клієнтська частина відповідає за взаємодію з користувачем і реалізацію інтерфейсу, тоді як серверна частина забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних.

Клієнт-серверна архітектура дозволяє забезпечити зручність використання системи, централізоване зберігання даних, а також можливість подальшого розширення функціоналу. Крім того, вона добре підходить для реалізації вебзастосунків, що відповідає вимогам даного проєкту.

Таким чином, для розроблення інформаційної системи оренди спортивних споруд обрано клієнт-серверну архітектуру, яка є оптимальним рішенням з точки зору простоти реалізації, ефективності та можливості подальшого розвитку системи.

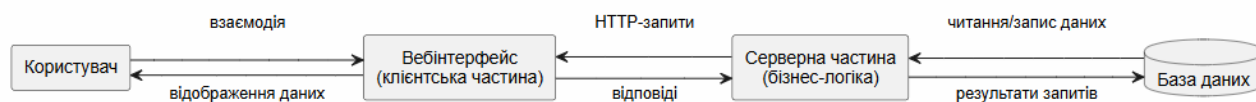


Рис. 2.1 – Клієнт-серверна архітектура інформаційної системи оренди спортивних споруд

На рисунку 2.1 подано загальну структуру клієнт-серверної архітектури інформаційної системи оренди спортивних споруд. Користувач взаємодіє з вебінтерфейсом, який передає запити до серверної частини. Сервер обробляє отримані дані, виконує бізнес-логіку та звертається до бази даних для зберігання або отримання інформації. Такий підхід забезпечує розподіл функцій між компонентами системи та підвищує ефективність її роботи.

2.2. Моделювання бізнес-процесів системи оренди спортивних споруд

Моделювання бізнес-процесів є важливим етапом проєктування інформаційної системи, оскільки дозволяє визначити основні дії користувачів і адміністратора, встановити послідовність виконання операцій та описати логіку функціонування системи. Для інформаційної системи оренди спортивних споруд бізнес-процеси охоплюють взаємодію користувачів із вебзастосунком під час пошуку об'єктів, перегляду інформації, бронювання та адміністрування.

Основним бізнес-процесом системи є процес бронювання спортивної споруди. Він починається з входу користувача до системи або переходу на головну сторінку вебзастосунку. Після цього користувач здійснює пошук потрібної спортивної споруди, переглядає доступні варіанти, обирає відповідний

об'єкт і перевіряє його доступність на необхідну дату та час. У разі наявності вільного часового проміжку користувач оформлює бронювання, після чого інформація зберігається в базі даних і стає доступною для подальшого перегляду.

Окремим бізнес-процесом є керування спортивними спорудами адміністратором. Адміністратор виконує додавання нових об'єктів, редагування їх характеристик, зміну розкладу доступності та перегляд інформації про бронювання. Це дозволяє підтримувати актуальність даних у системі та забезпечувати коректну роботу всіх її функцій.

Також важливим бізнес-процесом є керування обліковими записами користувачів. Система повинна забезпечувати реєстрацію нових користувачів, авторизацію зареєстрованих осіб, а також збереження інформації про їхні бронювання. Такий підхід дозволяє персоналізувати доступ до функцій системи та підвищити рівень безпеки.

Узагальнення бізнес-процесів дає можливість визначити основні сценарії роботи інформаційної системи, встановити взаємозв'язки між її компонентами та створити основу для подальшого функціонального й програмного проєктування.

На рисунку 2.2 зображено послідовність дій користувача під час бронювання спортивної споруди в інформаційній системі. Схема відображає основні етапи процесу: перехід до вебзастосунку, пошук потрібного об'єкта, перегляд його характеристик, вибір дати та часу, перевірку доступності, а також оформлення бронювання або отримання повідомлення про недоступність об'єкта. Такий підхід дозволяє наочно представити логіку функціонування системи та основні сценарії взаємодії користувача з вебзастосунком.

2.3. Проєктування функціональної структури системи

Проєктування функціональної структури інформаційної системи є одним із ключових етапів розроблення, оскільки саме на цьому етапі визначається склад основних компонентів системи, їх призначення, а також взаємозв'язки між ними.

Функціональна структура відображає логічну організацію системи та дозволяє сформулювати чітке уявлення про її внутрішню будову.

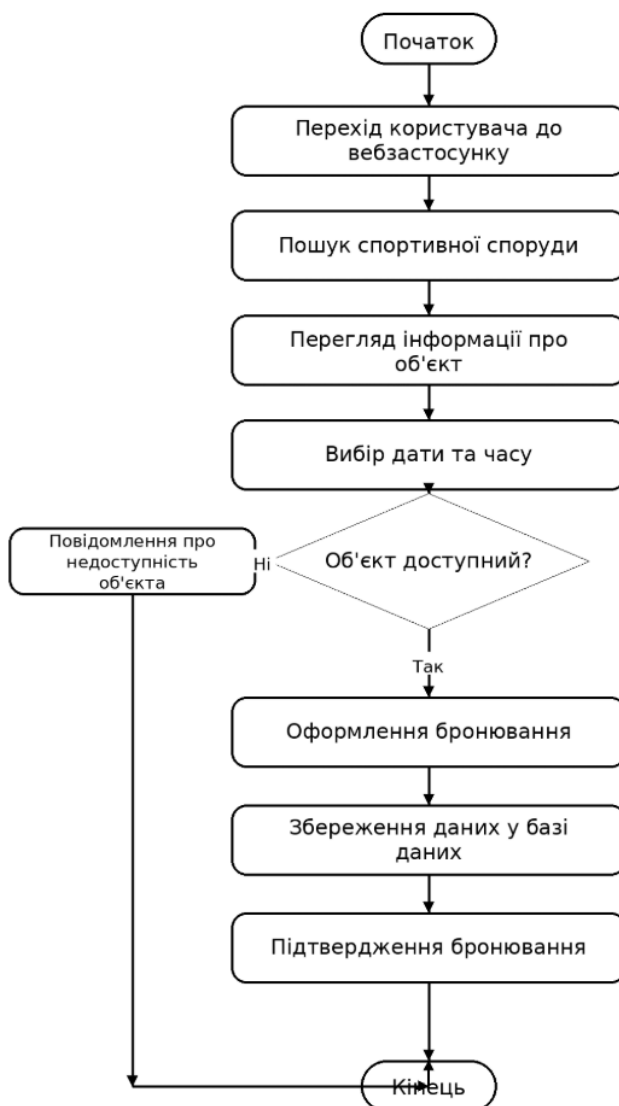


Рис. 2.2 Схема бізнес-процесу бронювання спортивної споруди

Для інформаційної системи оренди спортивних споруд доцільно застосувати модульний підхід до проектування. Такий підхід передбачає поділ системи на окремі функціональні модулі, кожен з яких відповідає за реалізацію певної частини функціоналу. Це забезпечує зручність розроблення, тестування, супроводу та подальшого розширення системи.

Функціональна структура розроблюваної системи включає низку взаємопов'язаних модулів, що забезпечують реалізацію основних бізнес-процесів, визначених на попередньому етапі.

Першим і одним із базових є модуль реєстрації та авторизації користувачів. Його основним призначенням є забезпечення доступу до системи лише для зареєстрованих користувачів, а також ідентифікація особи під час входу. У межах цього модуля реалізується створення облікового запису, перевірка введених даних, а також розмежування прав доступу між звичайними користувачами та адміністраторами. Наявність цього модуля дозволяє забезпечити безпеку системи та персоналізацію взаємодії з користувачем.

Другим важливим компонентом є модуль перегляду спортивних споруд. Він відповідає за відображення списку доступних об'єктів, їх характеристик, опису, фотографій (за наявності), а також іншої інформації, що може бути корисною для користувача. У цьому модулі також може бути реалізований функціонал фільтрації та пошуку, що дозволяє швидко знаходити потрібні об'єкти за заданими параметрами.

Ключовим елементом системи є модуль бронювання спортивних споруд. Саме він реалізує основну функцію системи — оформлення оренди об'єкта. У межах цього модуля користувач обирає дату та час, перевіряє доступність об'єкта та підтверджує бронювання. Модуль взаємодіє з базою даних для збереження інформації та забезпечує уникнення конфліктів при одночасному доступі кількох користувачів.

Тісно пов'язаним із модулем бронювання є модуль керування розкладом. Він відповідає за облік зайнятості спортивних споруд та відображення доступних часових інтервалів. Завдяки цьому модулю система забезпечує актуальність інформації та дозволяє користувачам бачити вільні слоти для бронювання в реальному часі.

Окрему роль у системі відіграє адміністративний модуль. Він призначений для керування всіма елементами системи з боку адміністратора. У межах цього модуля здійснюється додавання нових спортивних споруд, редагування їх характеристик, видалення об'єктів, а також контроль за бронюваннями користувачів. Адміністративний модуль забезпечує підтримку актуальності даних та стабільну роботу системи.

Крім основних модулів, система може включати допоміжні компоненти, такі як модуль повідомлень або сповіщень, який інформує користувачів про успішне бронювання або зміну статусу замовлення, а також модуль обробки помилок, що забезпечує коректну реакцію системи на некоректні дії користувачів.

Важливою особливістю функціональної структури є взаємодія між модулями. Наприклад, модуль бронювання використовує дані з модуля перегляду об'єктів та модуля розкладу, а також передає інформацію до бази даних. У свою чергу, адміністративний модуль має доступ до всіх основних компонентів системи, що дозволяє здійснювати повний контроль за її функціонуванням.

Таким чином, розроблена функціональна структура інформаційної системи оренди спортивних споруд забезпечує реалізацію всіх необхідних процесів, визначених на етапі аналізу, та створює основу для подальшого детального проєктування і програмної реалізації системи.



Рис. 2.3 Функціональна структура інформаційної системи оренди спортивних споруд

На рисунку 2.3 наведено функціональну структуру інформаційної системи оренди спортивних споруд. Центральним елементом є сама система, у межах якої виділено основні функціональні модулі: реєстрації та авторизації, перегляду спортивних споруд, бронювання, керування розкладом і адміністрування. Така структура відображає логічний поділ системи на окремі компоненти, кожен із яких забезпечує виконання визначених функцій.

2.4. Розробка UML-діаграм інформаційної системи

Під час проєктування інформаційної системи важливе значення має використання UML-діаграм, які дають змогу наочно відобразити структуру системи, взаємодію між її елементами та основні сценарії використання. UML є універсальною мовою моделювання, що широко застосовується у процесі аналізу, проєктування та документування програмного забезпечення.

Для інформаційної системи оренди спортивних споруд UML-діаграми дозволяють формалізувати вимоги до системи, деталізувати її функціональні можливості та представити внутрішню організацію програмного продукту. Використання таких діаграм є доцільним, оскільки вони забезпечують зручність сприйняття складних процесів і полегшують подальшу реалізацію програмного забезпечення.

У межах даної роботи доцільно розробити кілька основних UML-діаграм, серед яких особливе значення мають діаграма варіантів використання, діаграма класів та діаграма послідовностей. Кожна з них виконує окрему функцію в процесі проєктування системи.

Діаграма варіантів використання призначена для відображення взаємодії користувачів із системою. Вона дозволяє визначити основних акторів, їх ролі та перелік дій, які вони можуть виконувати в межах вебзастосунку. Така діаграма є корисною для формування загального уявлення про функціональні можливості системи.

Діаграма класів використовується для моделювання статичної структури системи. Вона відображає основні класи, їх атрибути, методи та зв'язки між ними. Це дає можливість сформувати уявлення про внутрішню побудову програмного забезпечення та проєктування бази даних.

Діаграма послідовностей, у свою чергу, демонструє часову послідовність обміну повідомленнями між об'єктами системи під час реалізації певного сценарію. Для інформаційної системи оренди спортивних споруд така діаграма є

особливо корисною для моделювання процесу бронювання, авторизації або взаємодії користувача із серверною частиною.

Таким чином, розроблення UML-діаграм є необхідним етапом проєктування інформаційної системи оренди спортивних споруд, оскільки дає змогу детальніше представити її логіку, структуру та взаємодію основних компонентів. Це створює основу для подальшої програмної реалізації системи та забезпечує більш якісне документування результатів проєктування.



Рис. 2.4 Діаграма варіантів використання інформаційної системи оренди спортивних споруд

На рисунку 2.4 наведено діаграму варіантів використання інформаційної системи оренди спортивних споруд. Вона відображає основних акторів системи, а саме користувача та адміністратора, а також перелік функцій, які доступні кожному з них. Діаграма демонструє, що користувач орієнтований переважно на пошук, перегляд і бронювання спортивних споруд, тоді як адміністратор виконує функції керування розкладом, спортивними об'єктами та користувачами системи.

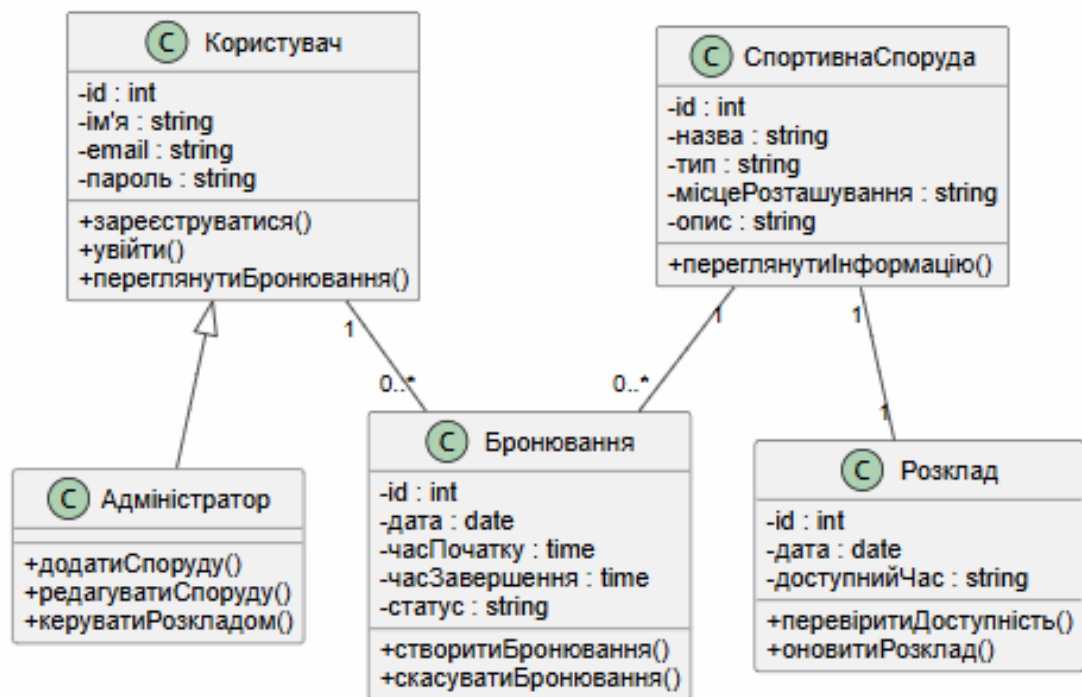


Рис. 2.5 Діаграма класів інформаційної системи оренди спортивних споруд

На рисунку 2.5 наведено діаграму класів інформаційної системи оренди спортивних споруд. Вона відображає основні сутності системи, їх атрибути, методи та зв'язки між ними. Центральними класами є користувач, адміністратор, спортивна споруда, бронювання та розклад. Побудована діаграма дозволяє визначити статичну структуру системи та є основою для подальшої програмної реалізації й проектування бази даних.

Діаграма класів демонструє внутрішню організацію інформаційної системи та показує, які сутності є ключовими для її функціонування. Клас «Користувач» описує зареєстрованих осіб, які взаємодіють із системою та

можуть здійснювати бронювання спортивних споруд. Клас «Адміністратор» є спеціалізованим різновидом користувача та має розширені можливості щодо керування об'єктами та розкладом.

Клас «Спортивна Споруда» містить основну інформацію про об'єкти, які доступні для оренди. Саме з цим класом пов'язаний клас «Бронювання», який зберігає дані про оформлені замовлення, часові межі бронювання та їх статус. Окреме місце займає клас «Розклад», що використовується для перевірки доступності спортивних споруд та запобігання конфліктам бронювання.

Таким чином, діаграма класів дозволяє формалізувати модель даних інформаційної системи та підготувати основу для реалізації як серверної логіки, так і структури бази даних.

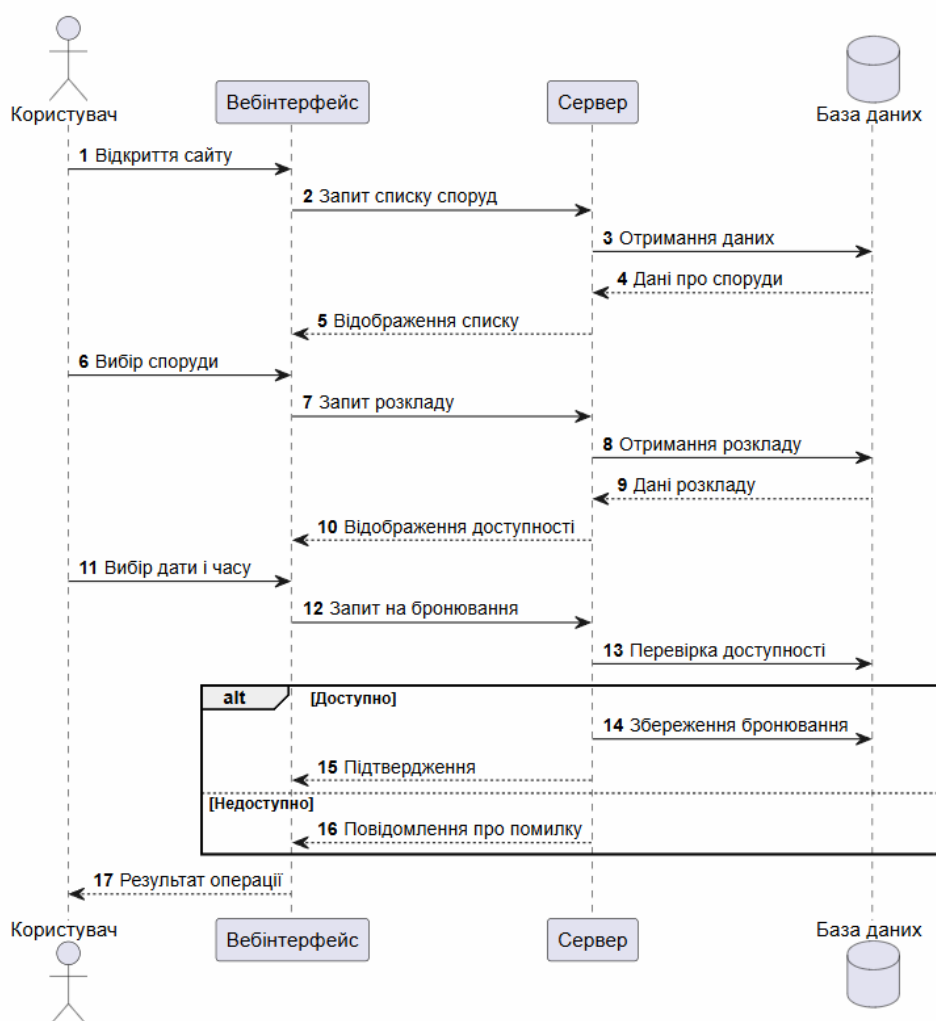


Рис. 2.6 Діаграма послідовностей процесу бронювання спортивної споруди

На рисунку 2.6 наведено діаграму послідовностей, яка відображає процес взаємодії користувача з інформаційною системою під час бронювання спортивної споруди. Діаграма демонструє обмін повідомленнями між основними компонентами системи: користувачем, вебінтерфейсом, сервером та базою даних.

У процесі бронювання користувач спочатку отримує інформацію про доступні спортивні споруди, після чого обирає необхідний об'єкт і перевіряє його доступність. Далі формується запит на бронювання, який обробляється серверною частиною із залученням бази даних. У разі успішної перевірки система зберігає інформацію про бронювання та підтверджує операцію, інакше користувач отримує повідомлення про неможливість виконання дії.

Таким чином, діаграма послідовностей дозволяє детально простежити логіку взаємодії компонентів системи та є важливим елементом проектування програмного забезпечення.

2.5. Проектування бази даних системи

Проектування бази даних є важливим етапом розроблення інформаційної системи, оскільки саме база даних забезпечує зберігання, обробку та доступ до всієї необхідної інформації. Для інформаційної системи оренди спортивних споруд база даних повинна забезпечувати збереження даних про користувачів, спортивні об'єкти, розклад їх використання та здійснені бронювання.

Основною метою проектування бази даних є створення такої структури, яка забезпечить:

- цілісність і узгодженість даних;
- мінімізацію дублювання інформації;
- швидкий доступ до даних;
- зручність подальшого розширення системи.

Аналіз сутностей предметної області

На основі аналізу предметної області було визначено основні сутності, які повинні бути представлені у базі даних:

- користувач;
- адміністратор;
- спортивна споруда;
- бронювання;
- розклад.

Кожна із зазначених сутностей має власні атрибути та взаємозв'язки з іншими сутностями.

Опис структури бази даних

База даних інформаційної системи оренди спортивних споруд складається з кількох взаємопов'язаних таблиць.

Таблиця 2.1

Перелік користувачів (Users)

Поле	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор користувача
name	VARCHAR	Ім'я користувача
email	VARCHAR	Електронна пошта
password	VARCHAR	Пароль
role	VARCHAR	Роль (user/admin)

Дана таблиця містить інформацію про всіх користувачів системи та використовується для авторизації й розмежування прав доступу.

Таблиця 2.2

Перелік спортивних споруд (Facilities)

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор споруди

Поле	Тип даних	Опис
name	VARCHAR	Назва об'єкта
type	VARCHAR	Тип (зал, поле, басейн)
location	VARCHAR	Місцезнаходження
description	TEXT	Опис
price	DECIMAL	Вартість оренди

Ця таблиця містить основну інформацію про об'єкти, доступні для бронювання.

Таблиця 2.3

Перелік об'єктів бронювання (Bookings)

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор бронювання
user_id	INT (FK)	Користувач
facility_id	INT (FK)	Спортивна споруда
date	DATE	Дата бронювання
start_time	TIME	Початок
end_time	TIME	Кінець
status	VARCHAR	Статус (активне/скасоване)

Таблиця забезпечує збереження інформації про всі здійснені бронювання.

Таблиця 2.4

Перелік атрибутів розкладу (Schedule)

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор запису
facility_id	INT (FK)	Споруда
date	DATE	Дата

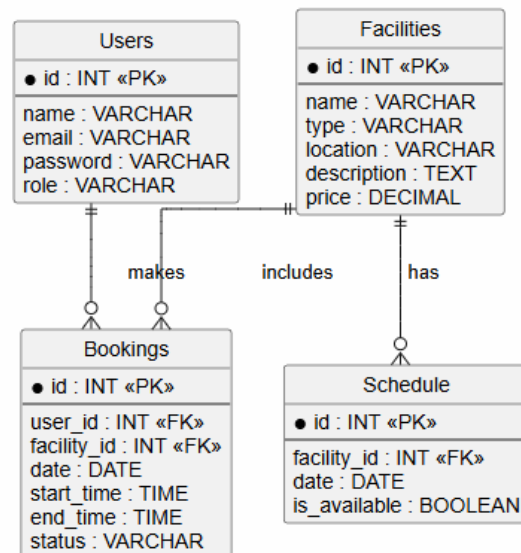
Поле	Тип даних	Опис
is_available	BOOLEAN	Доступність

Таблиця використовується для відображення зайнятості спортивних об'єктів.

Взаємозв'язки між таблицями

У базі даних реалізовано такі основні зв'язки (Рис. 2.7):

- один користувач може мати багато бронювань (зв'язок один-до-багатьох);
- одна спортивна споруда може бути заброньована багато разів;
- кожне бронювання пов'язане з конкретним користувачем і спорудою.



– Рис. 2.7 ER-діаграма бази даних інформаційної системи оренди спортивних споруд

Такий підхід дозволяє забезпечити логічну цілісність даних і уникнути конфліктів під час бронювання.

Нормалізація бази даних

Під час проєктування було застосовано принципи нормалізації, що дозволило:

- уникнути дублювання інформації;

- розділити дані на логічні таблиці;
- забезпечити ефективне оновлення даних.

База даних відповідає третій нормальній формі (3НФ), оскільки всі неключові атрибути залежать лише від первинного ключа.

2.6. Проєктування користувацького інтерфейсу вебзастосунку

Проєктування користувацького інтерфейсу є важливим етапом розроблення інформаційної системи, оскільки саме інтерфейс забезпечує взаємодію користувача із системою. Від зручності, зрозумілості та логічності інтерфейсу залежить ефективність використання вебзастосунку.

Основною метою проєктування інтерфейсу є створення простого, інтуїтивно зрозумілого та зручного середовища для користувача, яке дозволяє швидко виконувати необхідні дії, зокрема пошук, перегляд та бронювання спортивних споруд.

Основні вимоги до інтерфейсу

Під час розроблення інтерфейсу були враховані такі вимоги:

- простота та зрозумілість навігації;
- мінімальна кількість дій для виконання операцій;
- адаптивність до різних пристроїв;
- швидкий доступ до основних функцій;
- візуальна привабливість.

Основні сторінки вебзастосунку

Інтерфейс системи складається з кількох основних сторінок.

Головна сторінка є стартовою точкою (Рис. 2.8) взаємодії користувача із системою. Вона містить:

- список доступних спортивних споруд;
- пошукову форму;

- – фільтри (тип, ціна, місце);
- – навігаційне меню.

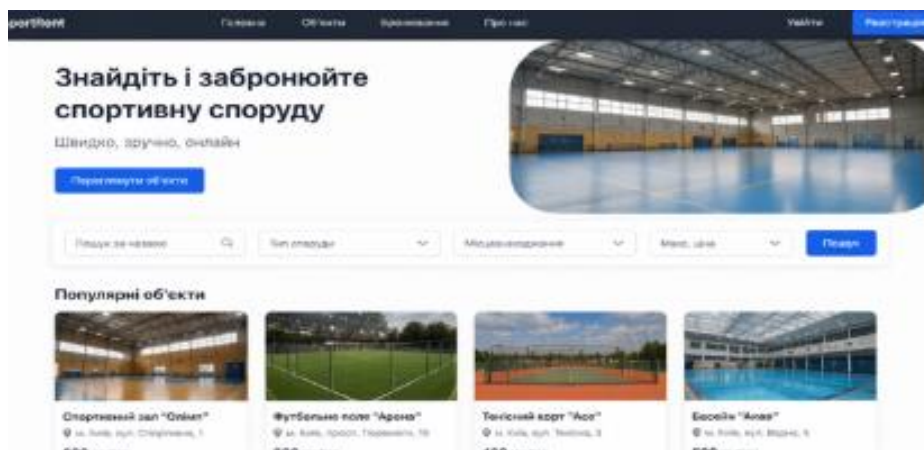


Рис. 2.8 Головна сторінка вебзастосунку

Сторінка авторизації та реєстрації

Ця сторінка (Рис. 2.9) забезпечує можливість створення облікового запису та входу в систему. Вона містить форму введення логіна та пароля.

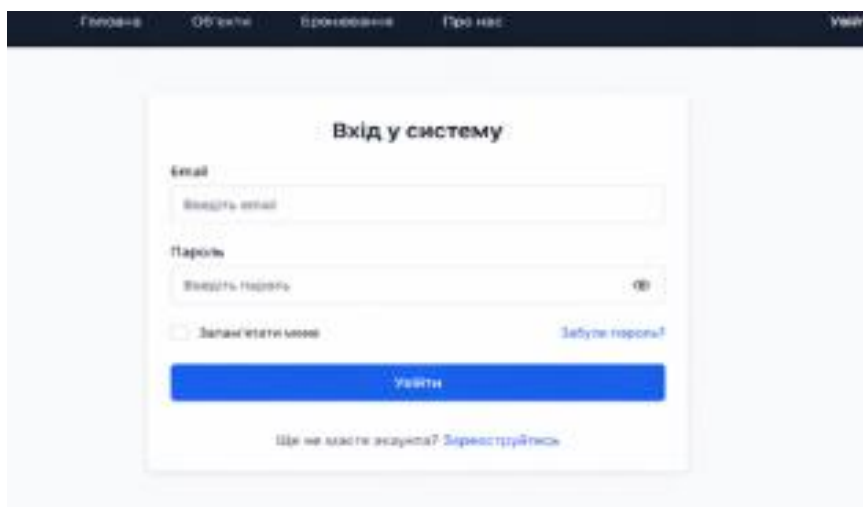


Рис. 2.9 Сторінка авторизації користувача

Сторінка перегляду спортивної споруди

На цій сторінці (Рис. 2.10) відображається детальна інформація про обраний об'єкт:

- – назва;
- – опис;
- – ціна;

- розклад доступності;
- кнопка бронювання.



Рис. 2.10 Сторінка перегляду спортивної споруди

Сторінка бронювання

Сторінка (Рис. 2.11) дозволяє користувачу обрати дату та час і оформити бронювання.

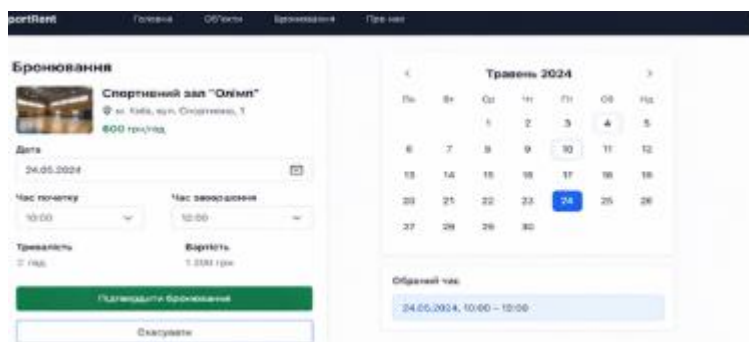


Рис. 2.11 Сторінка бронювання

Адміністративна панель

Адміністративний інтерфейс (Рис. 2.12) призначений для керування системою. Він включає:

- додавання нових споруд;
- редагування інформації;
- перегляд бронювань;
- управління користувачами.

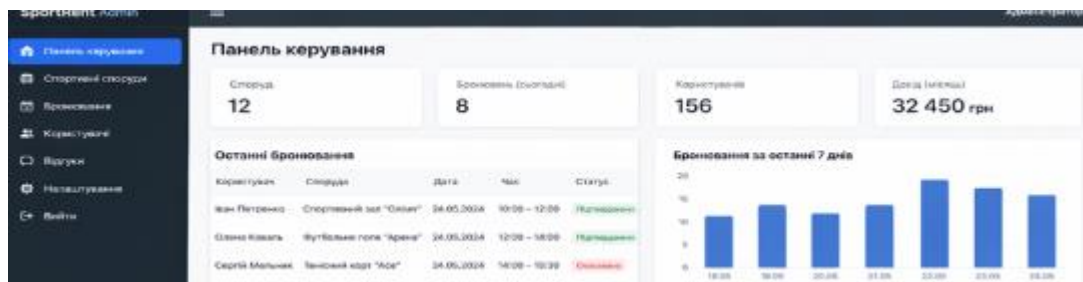


Рис. 2.12 Адміністративна панель системи

Навігація та структура інтерфейсу

Для забезпечення зручності використання було реалізовано навігаційне меню, яке дозволяє швидко переходити між основними сторінками системи. Меню розміщується у верхній частині сторінки та містить основні розділи: головна, об'єкти, бронювання, профіль.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Вибір засобів та технологій розроблення

Розроблення інформаційної системи оренди спортивних споруд потребує вибору сучасних та ефективних технологій, які забезпечують стабільну роботу системи, зручність використання та можливість подальшого розширення функціоналу.

Під час вибору засобів розроблення було враховано такі критерії:

- простота реалізації;
- швидкість розробки;
- наявність документації та підтримки;
- можливість створення вебзастосунку;
- масштабованість.

Вибір архітектурного підходу

Для реалізації системи було обрано клієнт-серверну архітектуру, яка передбачає поділ системи на дві частини:

- клієнтську (інтерфейс користувача);
- серверну (обробка даних і бізнес-логіка).

Такий підхід дозволяє забезпечити ефективну взаємодію між користувачем і системою та централізоване управління даними.

Вибір мови програмування та технологій

Для реалізації серверної частини було обрано мову програмування **Python** та фреймворк **Django**.

Python є однією з найпопулярніших мов програмування завдяки своїй простоті, зрозумілості синтаксису та широким можливостям. Використання Django дозволяє значно прискорити процес розроблення завдяки наявності

вбудованих компонентів для роботи з базами даних, авторизації користувачів та маршрутизації запитів.

Основні переваги Django:

- швидка розробка вебзастосунків;
- вбудована система аутентифікації;
- зручна робота з базами даних (ORM);
- високий рівень безпеки.

Технології клієнтської частини

Для створення інтерфейсу користувача використовуються стандартні вебтехнології:

- HTML – для структури сторінок;
- CSS – для оформлення інтерфейсу;
- JavaScript – для реалізації інтерактивності.

Застосування цих технологій дозволяє створити адаптивний та зручний інтерфейс, який працює у будь-якому сучасному браузері.

Система управління базою даних

У якості системи управління базами даних було обрано **MySQL**.

MySQL є надійною та продуктивною СУБД, яка широко використовується у веброзробці. Вона забезпечує:

- швидку обробку запитів;
- підтримку великих обсягів даних;
- інтеграцію з Django.

Інструменти розроблення

Під час розроблення використовувались такі інструменти:

- Visual Studio Code – середовище розробки;
- Git – система контролю версій;
- браузер Google Chrome – тестування вебзастосунку;
- PlantUML – створення діаграм.

3.2. Реалізація серверної частини системи

Серверна частина інформаційної системи оренди спортивних споруд відповідає за обробку запитів користувачів, реалізацію бізнес-логіки, взаємодію з базою даних та забезпечення коректної роботи всіх функцій системи.

Для реалізації серверної частини було використано фреймворк Django, який дозволяє швидко створювати надійні вебзастосунки та забезпечує зручну структуру проєкту.

Структура серверного застосунку

Серверна частина системи має модульну структуру, що дозволяє розділити функціональність на окремі компоненти.

Основні елементи структури проєкту:

- **models.py** – опис структури бази даних;
- **views.py** – обробка запитів користувача;
- **urls.py** – маршрутизація запитів;
- **templates/** – HTML-шаблони;
- **admin.py** – адміністративна панель;

Такий підхід дозволяє забезпечити зручність розроблення та подальшого супроводу системи.

Реалізація моделей даних

Моделі у Django використовуються для опису структури бази даних. Кожна модель відповідає окремій таблиці.

Приклад моделі користувача

```
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):

    role = models.CharField(max_length=10)
```

Модель спортивної споруди

```
class Facility(models.Model):

    name = models.CharField(max_length=100)

    type = models.CharField(max_length=50)

    location = models.CharField(max_length=150)

    description = models.TextField()

    price = models.DecimalField(max_digits=8, decimal_places=2)
```

Модель бронювання

```
class Booking(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    facility = models.ForeignKey(Facility, on_delete=models.CASCADE)

    date = models.DateField()

    start_time = models.TimeField()

    end_time = models.TimeField()

    status = models.CharField(max_length=20)
```

Реалізація обробки запитів (views)

Файл `views.py` відповідає за обробку HTTP-запитів та взаємодію між користувачем і базою даних.

Приклад отримання списку споруд

```
from django.shortcuts import render

from .models import Facility

def facility_list(request):
```

```

facilities = Facility.objects.all()

return render(request, 'facilities.html', {'facilities': facilities})

```

Приклад створення бронювання

```

from django.shortcuts import redirect

from .models import Booking

def create_booking(request):

    if request.method == "POST":

        Booking.objects.create(

            user=request.user,

            facility_id=request.POST.get('facility_id'),

            date=request.POST.get('date'),

            start_time=request.POST.get('start_time'),

            end_time=request.POST.get('end_time'),

            status="active"

        )

    return redirect('home')

```

Маршрутизація запитів

Файл `urls.py` визначає, які функції обробляють запити користувачів.

```

from django.urls import path

from . import views

urlpatterns = [

```

```

    path("", views.facility_list, name='home'),

    path('booking/', views.create_booking, name='booking'),

]

```

Адміністративна панель

Django має вбудовану адміністративну панель, яка дозволяє керувати даними без додаткової розробки.

```

from django.contrib import admin

from .models import Facility, Booking

```

```
admin.site.register(Facility)
```

```
admin.site.register(Booking)
```

Адміністративна панель використовується для:

- додавання спортивних споруд;
- редагування даних;
- перегляду бронювань.

Забезпечення безпеки

Під час реалізації серверної частини було враховано питання безпеки:

- використання вбудованої системи аутентифікації Django;
- захист від SQL-ін'єкцій завдяки ORM;
- перевірка даних, введених користувачем;
- розмежування прав доступу.

3.3. Реалізація клієнтської частини вебзастосунку

Клієнтська частина інформаційної системи оренди спортивних споруд відповідає за взаємодію користувача із системою та забезпечує відображення даних, отриманих із серверної частини.

Для реалізації клієнтської частини було використано стандартні вебтехнології: HTML, CSS та JavaScript. Такий підхід дозволяє створити адаптивний, зручний та функціональний інтерфейс, який працює у будь-якому сучасному браузері.

Структура клієнтської частини

Клієнтська частина складається з HTML-шаблонів, які формують сторінки вебзастосунку, та стилів, що відповідають за зовнішній вигляд.

Основні компоненти клієнтської частини:

- HTML-шаблони (templates);
- CSS-стилі;
- JavaScript для обробки подій;
- інтеграція з серверною частиною через HTTP-запити.

Реалізація головної сторінки

Головна сторінка (Рис. 3.1) є основним елементом інтерфейсу, оскільки саме з неї починається взаємодія користувача із системою.

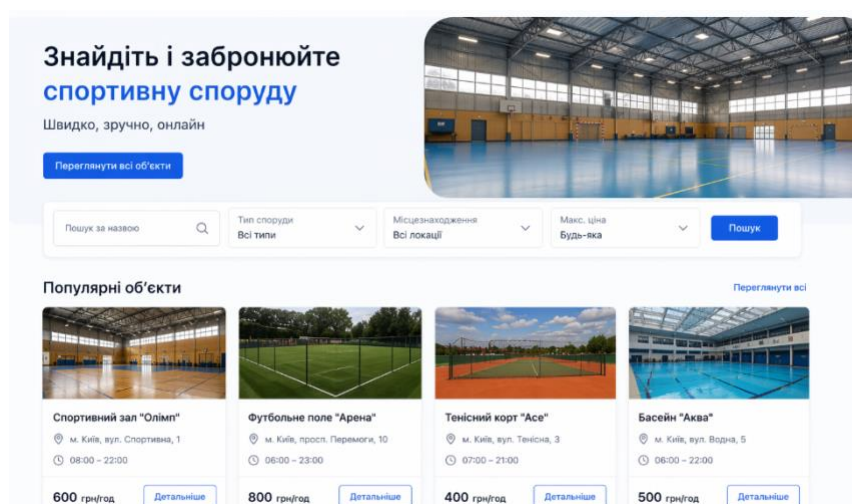


Рис. 3.1 Головна сторінка вебзастосунку

Вона містить:

- список доступних спортивних споруд;
- форму пошуку;
- навігаційне меню;

- кнопки переходу до інших сторінок.

Реалізація сторінки авторизації

Сторінка авторизації (Рис. 3.2) забезпечує доступ користувача до персоналізованих функцій системи.

Вона містить:

- форму введення логіна та пароля;
- кнопку входу;
- посилання на реєстрацію.

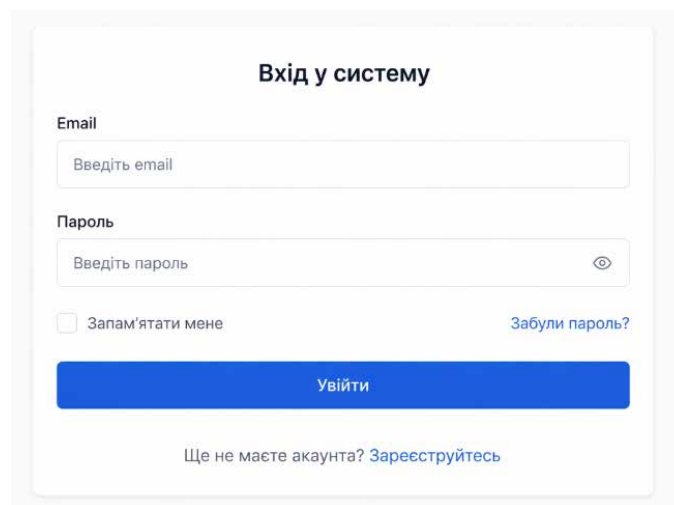
The image shows a login form titled "Вхід у систему" (Login to the system). It contains two input fields: "Email" with the placeholder text "Введіть email" and "Пароль" (Password) with the placeholder text "Введіть пароль" and a toggle icon for password visibility. Below the password field is a checkbox labeled "Запам'ятати мене" (Remember me) and a link "Забули пароль?" (Forgot password?). A large blue button labeled "Увійти" (Login) is positioned below the inputs. At the bottom, there is a link "Ще не маєте акаунта? Зареєструйтесь" (Don't have an account? Register).

Рис. 3.2 Сторінка авторизації користувача

Опис:

Сторінка авторизації реалізує механізм входу до системи та дозволяє обмежити доступ до функціоналу лише для зареєстрованих користувачів.

Реалізація сторінки перегляду споруди

Ця сторінка (Рис. 3.3) надає детальну інформацію про обраний об'єкт.

Вона включає:

- назву та опис споруди;
- фотографії;
- ціну;
- доступність;
- кнопку бронювання.

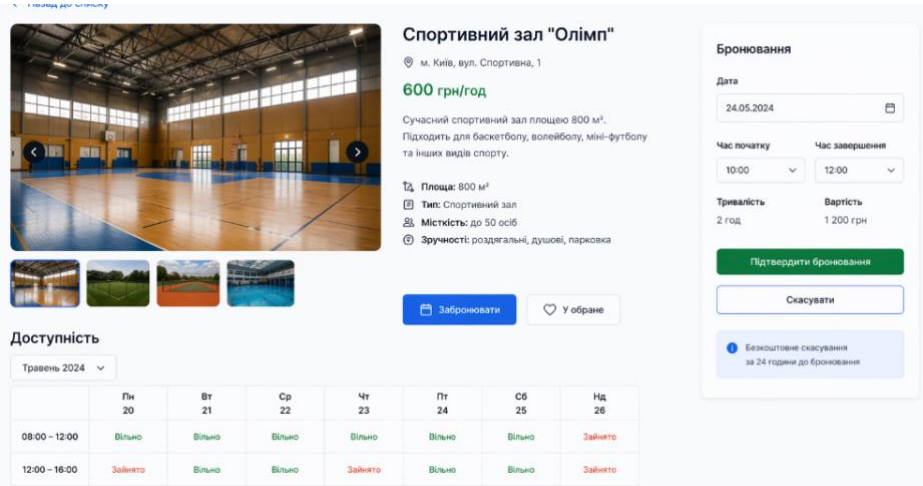


Рис. 3.3 Сторінка перегляду спортивної споруди

Опис:

Сторінка дозволяє користувачу ознайомитися з характеристиками об’єкта перед оформленням бронювання.

Реалізація сторінки бронювання

Сторінка (Рис. 3.4) бронювання є ключовою для системи, оскільки саме вона забезпечує основний функціонал.

Вона містить:

- вибір дати;
- вибір часу;
- підтвердження бронювання.

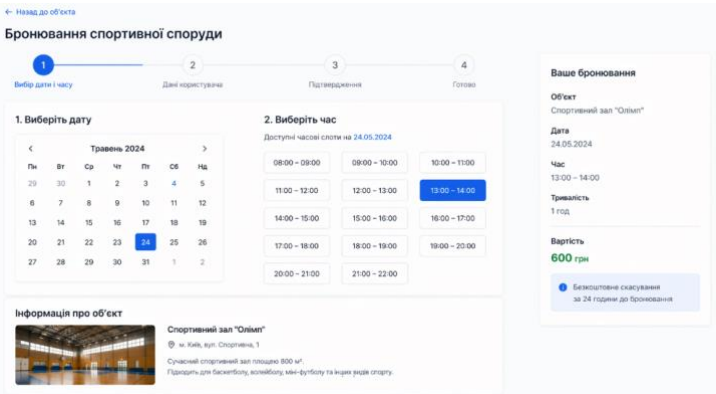


Рис. 3.4 Процес бронювання спортивної споруди

Опис:

Сторінка дозволяє користувачу обрати необхідний час та здійснити бронювання спортивної споруди.

Реалізація адміністративної панелі

Адміністративна панель (Рис. 3.5) забезпечує керування системою.

Функції:

- додавання споруд;
- редагування інформації;
- перегляд бронювань;
- управління користувачами.

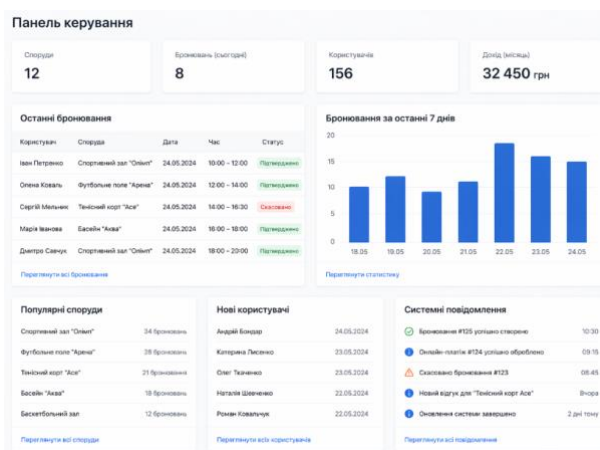


Рис. 3.5 Адміністративна панель системи

Опис:

Адміністративна панель дозволяє здійснювати повний контроль над системою та підтримувати актуальність даних.

Інтерактивність інтерфейсу

Для покращення взаємодії з користувачем було використано JavaScript, який забезпечує:

- обробку форм;
- перевірку введених даних;
- динамічне оновлення сторінок;
- взаємодію із сервером без перезавантаження сторінки.

Адаптивність інтерфейсу

Інтерфейс вебзастосунку адаптований для роботи на різних пристроях:

- комп'ютерах;
- планшетах;
- смартфонах.

Це забезпечує зручність використання системи незалежно від типу пристрою.

3.4. Розроблення модуля реєстрації та авторизації користувачів

Модуль реєстрації та авторизації є важливою складовою інформаційної системи оренди спортивних споруд, оскільки він забезпечує ідентифікацію користувачів та контроль доступу до функціоналу системи.

Основною метою даного модуля є надання можливості користувачам створювати облікові записи, входити до системи та використовувати персоналізовані функції.

Функціональні можливості модуля

Модуль забезпечує:

- реєстрацію нових користувачів;
- авторизацію (вхід у систему);
- збереження облікових даних;
- розмежування прав доступу;
- вихід із системи.

Реалізація реєстрації користувача

Реєстрація передбачає введення користувачем таких даних:

- ім'я;
- електронна пошта;
- пароль.

Після заповнення форми дані передаються на сервер, де відбувається їх перевірка та збереження у базі даних.

Приклад реалізації реєстрації (Django)

```
from django.contrib.auth.models import User

from django.shortcuts import render, redirect

def register(request):

    if request.method == "POST":

        User.objects.create_user(

            username=request.POST.get('username'),

            email=request.POST.get('email'),

            password=request.POST.get('password')

        )

        return redirect('login')

    return render(request, 'register.html')
```

Реалізація авторизації користувача

Авторизація дозволяє користувачу увійти до системи за допомогою логіна та пароля.

Приклад реалізації входу

```
from django.contrib.auth import authenticate, login

def user_login(request):

    if request.method == "POST":
```

```

user = authenticate(

    username=request.POST.get('username'),

    password=request.POST.get('password')

)

if user:

    login(request, user)

    return redirect('home')

return render(request, 'login.html')

```

Захист даних користувачів

Для забезпечення безпеки було реалізовано:

- шифрування паролів;
- перевірку введених даних;
- захист від несанкціонованого доступу;
- використання сесій для збереження стану користувача.

Розмежування прав доступу

У системі передбачено два типи користувачів:

- звичайний користувач;
- адміністратор.

Адміністратор має розширені права доступу до керування системою, тоді як звичайний користувач може лише здійснювати бронювання.

3.5. Розроблення модуля перегляду спортивних споруд та бронювання

Модуль перегляду спортивних споруд та бронювання є ключовим компонентом інформаційної системи, оскільки саме він забезпечує основний функціонал взаємодії користувача із системою.

Основною метою модуля є надання користувачу можливості переглядати доступні спортивні споруди, отримувати детальну інформацію про них та здійснювати бронювання у зручний час.

Функціональні можливості модуля

Модуль забезпечує:

- відображення списку спортивних споруд;
- перегляд детальної інформації про об'єкт;
- перевірку доступності споруди;
- вибір дати та часу;
- створення бронювання;
- відображення статусу бронювання.

Реалізація перегляду спортивних споруд

Перегляд спортивних споруд здійснюється через отримання даних із бази даних та їх відображення у вигляді списку на головній сторінці.

Приклад реалізації

```
def facility_list(request):
```

```
    facilities = Facility.objects.all()
```

```
    return render(request, 'facilities.html', {'facilities': facilities})
```

У результаті користувач отримує список об'єктів із короткою інформацією: назва, місцезнаходження, ціна та доступність.

Реалізація перегляду детальної інформації

Після вибору конкретної споруди користувач переходить на сторінку з детальним описом.

Приклад реалізації

```
def facility_detail(request, id):
```

```
    facility = Facility.objects.get(id=id)
```



```
return render(request, 'facility_detail.html', {'facility': facility})
```

На цій сторінці відображаються:

- опис споруди;
- характеристики;
- розклад доступності;
- кнопка бронювання.

Перевірка доступності об'єкта

Перед створенням бронювання система перевіряє, чи вільний обраний часовий інтервал.

```
from .models import Booking
```

```
def check_availability(facility, date, start_time, end_time):
```

```
    bookings = Booking.objects.filter(
```

```
        facility=facility,
```

```
        date=date,
```

```
        start_time__lt=end_time,
```

```
        end_time__gt=start_time
```

```
    )
```

```
    return not bookings.exists()
```

Такий підхід дозволяє уникнути конфліктів бронювання.

Реалізація процесу бронювання

Після перевірки доступності користувач може здійснити бронювання.

```
def create_booking(request):
```

```
    if request.method == "POST":
```

```
        facility_id = request.POST.get('facility_id')
```

```

date = request.POST.get('date')

start_time = request.POST.get('start_time')

end_time = request.POST.get('end_time')


facility = Facility.objects.get(id=facility_id)


if check_availability(facility, date, start_time, end_time):

    Booking.objects.create(

        user=request.user,

        facility=facility,

        date=date,

        start_time=start_time,

        end_time=end_time,

        status="active"

    )

```

Відображення бронювання користувача

Після створення бронювання інформація зберігається в базі даних та стає доступною користувачу.

Користувач може:

- переглядати свої бронювання;
- скасовувати їх;
- відстежувати статус.

Зручність використання модуля

Для підвищення зручності користувача було реалізовано:

- календар вибору дати;
- список доступних часових слотів;
- миттєве підтвердження бронювання;
- інформування про помилки.

3.6. Розроблення адміністративного модуля керування системою

Адміністративний модуль є важливою складовою інформаційної системи оренди спортивних споруд, оскільки він забезпечує можливість керування всіма елементами системи та підтримання актуальності даних.

Основною метою адміністративного модуля є надання адміністратору інструментів для контролю за роботою системи, управління спортивними спорудами, бронюваннями та користувачами.

Функціональні можливості адміністративного модуля

Адміністративний модуль забезпечує:

- додавання нових спортивних споруд;
- редагування інформації про об'єкти;
- видалення споруд;
- перегляд та керування бронюваннями;
- управління користувачами;
- перегляд статистики системи.

Реалізація управління спортивними спорудами

Адміністратор має можливість створювати нові об'єкти та редагувати існуючі.

Приклад створення об'єкта

```
def add_facility(request):
```

```
    if request.method == "POST":
```

```

Facility.objects.create(

    name=request.POST.get('name'),

    type=request.POST.get('type'),

    location=request.POST.get('location'),

    description=request.POST.get('description'),

    price=request.POST.get('price')

)

```

Це дозволяє підтримувати актуальний перелік спортивних споруд.

Управління бронюваннями

Адміністратор має доступ до всіх бронювань у системі та може:

- переглядати їх;
- змінювати статус;
- скасовувати бронювання.

```

def manage_booking(request, id):

    booking = Booking.objects.get(id=id)

    booking.status = "cancelled"

    booking.save()

```

Управління користувачами

Адміністратор може переглядати список користувачів та виконувати дії:

- блокування користувача;
- редагування даних;
- видалення облікового запису.

Використання адміністративної панелі Django

Для спрощення реалізації адміністративного модуля було використано вбудовану адміністративну панель Django.

```
from django.contrib import admin  
  
from .models import Facility, Booking, User
```

```
admin.site.register(Facility)
```

```
admin.site.register(Booking)
```

```
admin.site.register(User)
```

Це дозволяє:

- швидко керувати даними;
- не створювати окремий інтерфейс з нуля;
- зекономити час розробки.

Забезпечення безпеки адміністративного доступу

Для захисту адміністративного модуля було реалізовано:

- доступ лише для користувачів із роллю адміністратора;
- перевірку прав доступу;
- використання авторизації.

Зручність використання адміністративної панелі

Інтерфейс адміністративної панелі забезпечує:

- зручну навігацію;
- швидкий доступ до даних;
- відображення статистики;
- можливість пошуку та фільтрації.

3.7. Інтеграція програмних модулів з базою даних

Інтеграція програмних модулів з базою даних є завершальним етапом реалізації інформаційної системи, оскільки саме вона забезпечує взаємодію між серверною частиною та збереженими даними.

Основною метою інтеграції є забезпечення коректного зчитування, збереження та оновлення інформації в базі даних під час роботи системи.

Механізм взаємодії з базою даних

У розробленій системі взаємодія з базою даних здійснюється за допомогою ORM (Object-Relational Mapping), який надає фреймворк Django.

ORM дозволяє працювати з базою даних у вигляді об'єктів Python без необхідності написання складних SQL-запитів.

Основні операції:

- створення записів;
- зчитування даних;
- оновлення інформації;
- видалення записів.

Приклади взаємодії з базою даних

Створення запису

```
Booking.objects.create(
```

```
    user=request.user,
```

```
    facility=facility,
```

```
    date=date,
```

```
    start_time=start_time,
```

```
    end_time=end_time
```

```
)
```

Отримання даних

```
facilities = Facility.objects.all()
```

Оновлення запису

```
booking.status = "cancelled"
```

```
booking.save()
```

Видалення запису

```
booking.delete()
```

Зв'язок між модулями системи

У системі всі модулі взаємодіють між собою через базу даних:

- модуль авторизації працює з таблицею користувачів;
- модуль бронювання взаємодіє з таблицею бронювань;
- модуль перегляду використовує дані про спортивні споруди;
- адміністративний модуль має доступ до всіх таблиць.

Такий підхід (Рис. 3.6) забезпечує централізоване зберігання інформації та узгоджену роботу системи.

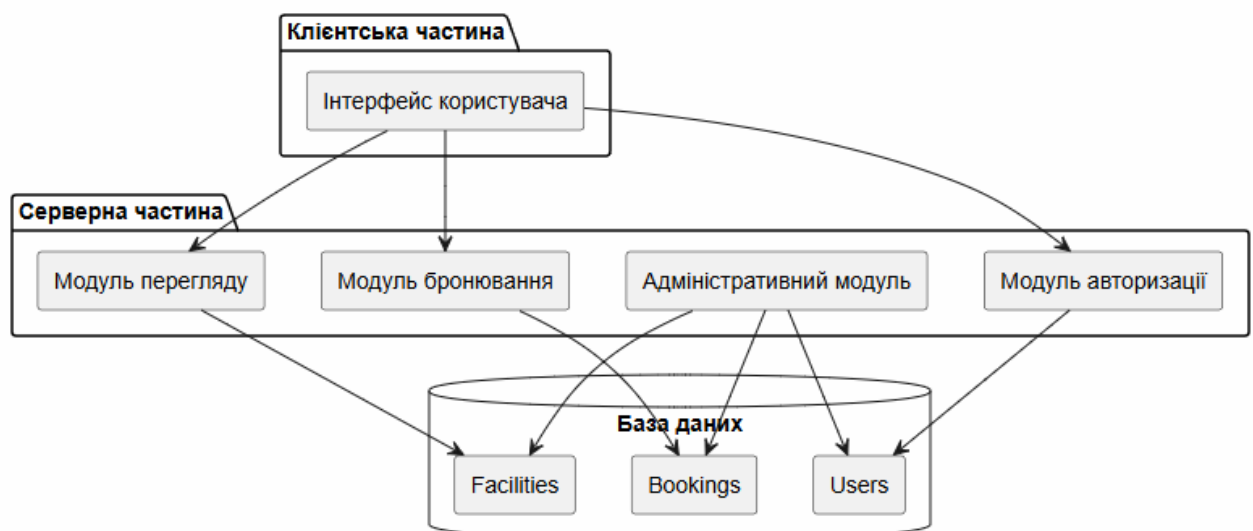


Рис. 3.6 Схема взаємодії програмних модулів із базою даних

На рисунку 3.6 наведено схему взаємодії основних програмних модулів інформаційної системи з базою даних. Клієнтська частина передає запити до

серверної, де відповідні модулі обробляють інформацію та взаємодіють із базою даних. Такий підхід забезпечує централізоване управління даними та узгоджену роботу всієї системи.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Організація та методика тестування інформаційної системи

Тестування є завершальним етапом розроблення інформаційної системи, який дозволяє перевірити коректність її роботи, виявити можливі помилки та оцінити відповідність реалізованого функціоналу поставленим вимогам.

Основною метою тестування є забезпечення стабільної та надійної роботи інформаційної системи оренди спортивних споруд, а також перевірка правильності реалізації основних функцій.

Завдання тестування

У процесі тестування було поставлено такі завдання:

- перевірка функціональності системи;
- виявлення помилок у роботі;
- перевірка коректності обробки даних;
- оцінка зручності використання;
- перевірка взаємодії між модулями.

Види тестування

Під час перевірки системи було використано кілька видів тестування.

Функціональне тестування

Функціональне тестування спрямоване на перевірку відповідності роботи системи її функціональним вимогам. У межах цього тестування перевірялися:

- реєстрація користувача;
- авторизація;
- перегляд спортивних споруд;
- процес бронювання;

- робота адміністративного модуля.

Тестування інтерфейсу користувача

Цей вид тестування дозволяє оцінити зручність використання системи та логічність інтерфейсу.

Було перевірено:

- зрозумілість навігації;
- доступність основних функцій;
- коректність відображення сторінок;
- адаптивність інтерфейсу.

Тестування продуктивності

Тестування продуктивності дозволяє оцінити швидкість роботи системи та її реакцію на дії користувача.

Було перевірено:

- швидкість завантаження сторінок;
- час обробки запитів;
- стабільність роботи при повторних діях.

Тестування безпеки

Тестування безпеки спрямоване на перевірку захищеності системи від несанкціонованого доступу.

Було перевірено:

- коректність авторизації;
- захист персональних даних;
- обмеження доступу до адміністративних функцій.

Методика проведення тестування

Тестування проводилося шляхом виконання типових сценаріїв використання системи.

Основні сценарії:

1. Реєстрація нового користувача

2. Вхід у систему
3. Пошук спортивної споруди
4. Перегляд інформації про об'єкт
5. Бронювання споруди
6. Скасування бронювання
7. Дії адміністратора

Кожен сценарій перевірявся на коректність виконання та відповідність очікуваному результату.

Інструменти тестування

Для тестування використовувались:

- веббраузер Google Chrome;
- вбудовані інструменти розробника;
- тестові облікові записи;
- ручне тестування сценаріїв.

4.2. Перевірка функціональних можливостей вебзастосунку

Перевірка функціональних можливостей інформаційної системи оренди спортивних споруд здійснювалася шляхом виконання основних сценаріїв роботи користувача та адміністратора.

Метою даного етапу тестування є підтвердження коректної роботи реалізованого функціоналу відповідно до вимог, визначених на етапі проєктування системи.

Опис процесу тестування

Тестування проводилося вручну шляхом послідовного виконання дій у вебзастосунку. Для цього використовувалися тестові облікові записи та різні варіанти вхідних даних.

У процесі тестування перевірялися:

- – правильність обробки даних;
- – коректність переходів між сторінками;
- – відповідність результатів очікуваням;
- – стабільність роботи системи.

Результати тестування

Результати перевірки функціональних можливостей наведено у таблиці 4.1.

Таблиця 4.1

Результати функціонального тестування системи

№	Функція	Вхідні дані	Очікуваний результат	Фактичний результат
1	Реєстрація користувача	Ім'я, email, пароль	Створення нового акаунта	Виконано
2	Авторизація	Логін, пароль	Успішний вхід у систему	Виконано
3	Перегляд споруд	Запит списку	Відображення списку	Виконано
4	Перегляд деталей	Вибір об'єкта	Відображення інформації	Виконано
5	Бронювання	Дата, час	Створення бронювання	Виконано
6	Перевірка доступності	Перетин часу	Відмова у бронюванні	Виконано
7	Скасування бронювання	ID бронювання	Видалення запису	Виконано
8	Додавання споруди (адмін)	Дані об'єкта	Створення запису	Виконано

Аналіз результатів тестування

Результати тестування показали, що всі основні функції системи працюють коректно. Під час виконання тестових сценаріїв не було виявлено критичних помилок, які могли б вплинути на роботу системи.

Було підтверджено:

- стабільність роботи вебзастосунку;
- коректність взаємодії з базою даних;
- правильність реалізації бізнес-логіки;
- відповідність функціоналу поставленим вимогам.

Виявлені недоліки

У процесі тестування було виявлено незначні недоліки, які не впливають на основну функціональність:

- можливі затримки при завантаженні сторінок;
- відсутність розширеної системи фільтрації;
- обмежена кількість повідомлень користувачу.

4.3. Аналіз результатів тестування інформаційної системи

Аналіз результатів тестування є важливим етапом оцінювання якості розробленої інформаційної системи, оскільки дозволяє узагальнити отримані результати, виявити сильні сторони системи та визначити можливі напрями її вдосконалення.

У процесі тестування було перевірено основні функціональні можливості системи, взаємодію між модулями, а також стабільність її роботи.

Оцінка результатів тестування

На основі отриманих результатів можна зробити висновок, що система:

- коректно реалізує всі основні функції;
- забезпечує стабільну роботу без критичних збоїв;
- правильно обробляє введені дані;

- забезпечує взаємодію між усіма модулями.

Також було підтверджено, що система відповідає функціональним вимогам, визначеним на етапі проєктування.

Аналіз продуктивності системи

У ході тестування було встановлено, що вебзастосунок демонструє достатню швидкість роботи:

- сторінки завантажуються без значних затримок;
- запити до бази даних обробляються швидко;
- система стабільно працює при повторних діях користувача.

Аналіз зручності використання

Інтерфейс системи було оцінено як:

- зрозумілий для користувача;
- логічно структурований;
- простий у використанні;
- доступний на різних пристроях.

Користувач може легко виконувати основні дії без додаткових інструкцій.

На Рисунку 4.1 наведено загальну схему процесу тестування інформаційної системи. Вона відображає послідовність дій під час перевірки функціоналу, включаючи виконання тестових сценаріїв, аналіз результатів та виправлення помилок у разі їх виявлення.

4.4. Оцінка зручності використання та практичної ефективності системи

Оцінка зручності використання та практичної ефективності інформаційної системи оренди спортивних споруд є важливим етапом, який дозволяє визначити, наскільки розроблений вебзастосунок відповідає потребам користувачів та може бути використаний у реальних умовах.

Оцінка зручності використання

Зручність використання системи визначається простотою взаємодії користувача з інтерфейсом, швидкістю виконання дій та доступністю основних функцій.

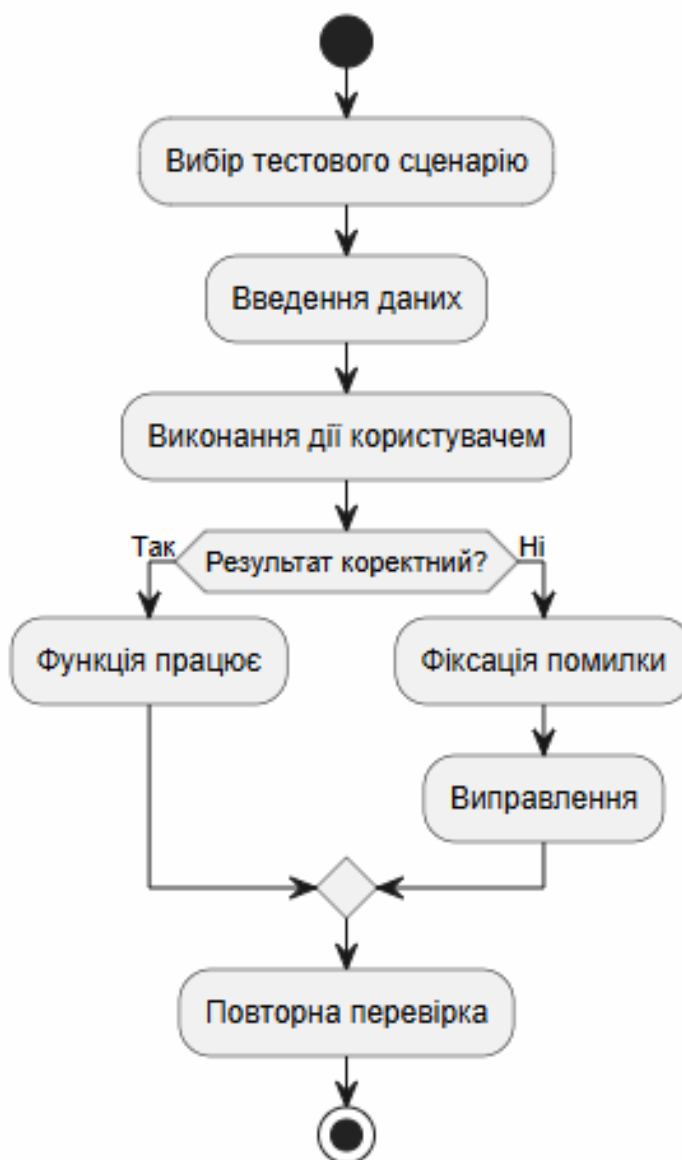


Рис. 4.1 Схема процесу тестування інформаційної системи

У ході аналізу було встановлено, що система має такі переваги:

- інтуїтивно зрозумілий інтерфейс;
- логічну структуру сторінок;
- швидкий доступ до основних функцій;
- мінімальну кількість дій для виконання операцій;

- зручну навігацію між сторінками.

Користувач може без додаткового навчання:

- – знайти спортивну споруду;
- – переглянути інформацію;
- – здійснити бронювання;
- – керувати своїми замовленнями.

Оцінка ефективності системи

Практична ефективність системи визначається її здатністю спрощувати та автоматизувати процеси оренди спортивних споруд.

У результаті аналізу було встановлено, що система:

- значно скорочує час пошуку об'єктів;
- автоматизує процес бронювання;
- зменшує ймовірність помилок;
- забезпечує зручний контроль за розкладом;
- дозволяє ефективно керувати даними.

Порівняння з традиційними методами

У порівнянні з традиційними способами організації оренди (телефонні дзвінки, особисті звернення) розроблена система має суттєві переваги:

- доступність у будь-який час;
- швидкість обробки інформації;
- відсутність людського фактору;
- автоматичне оновлення розкладу;
- зручність для користувача.
- На рисунку 4.2 наведено схему оцінки ефективності інформаційної системи, яка враховує основні критерії — зручність використання та швидкість роботи. Такий підхід дозволяє визначити загальний рівень ефективності розробленого вебзастосунку.

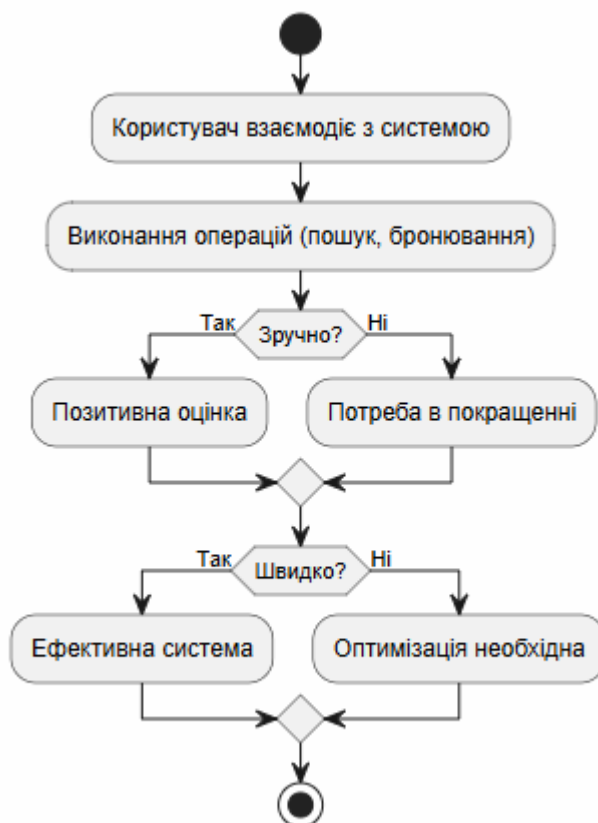


Рис. 4.2 Схема оцінки ефективності інформаційної системи

4.5. Рекомендації щодо подальшого вдосконалення системи

Незважаючи на те, що розроблена інформаційна система оренди спортивних споруд успішно реалізує основний функціонал і відповідає поставленим вимогам, існують можливості для її подальшого вдосконалення та розширення.

Розвиток системи дозволить підвищити її ефективність, зручність використання та конкурентоспроможність (Рис. 4.3).

Розширення функціональних можливостей

Одним із напрямів удосконалення є розширення функціоналу системи, зокрема:

- додавання онлайн-оплати за бронювання;
- інтеграція з платіжними системами;
- реалізація системи сповіщень (email, SMS);

- додавання рейтингу та відгуків користувачів;
- можливість фільтрації за більшою кількістю параметрів.



Рис. 4.3 Основні напрями вдосконалення інформаційної системи

Покращення інтерфейсу користувача

Для підвищення зручності використання доцільно:

- покращити дизайн сторінок;
- додати анімації та інтерактивні елементи;
- оптимізувати навігацію;
- реалізувати персоналізований кабінет користувача.

Підвищення продуктивності системи

Для покращення швидкодії системи можна:

- оптимізувати запити до бази даних;
- використовувати кешування;
- зменшити час завантаження сторінок;
- впровадити асинхронну обробку запитів.

Покращення безпеки

Удосконалення безпеки може включати:

- впровадження двофакторної аутентифікації;

- додатковий захист даних користувачів;
- контроль підозрілої активності;
- регулярне оновлення системи.

Масштабування системи

У майбутньому система може бути розширена до:

- мобільного додатку;
- підтримки великої кількості користувачів;
- інтеграції з іншими сервісами;
- використання хмарних технологій.

ВИСНОВКИ

У результаті виконання бакалаврської роботи було розроблено програмне забезпечення інформаційної системи оренди спортивних споруд, яке забезпечує автоматизацію процесів пошуку, перегляду та бронювання спортивних об'єктів.

У процесі виконання роботи було проаналізовано предметну область оренди спортивних споруд, визначено її основні особливості та проблеми, пов'язані з використанням традиційних методів організації оренди. Проведений аналіз сучасних інформаційних систем дозволив виявити їхні переваги та недоліки, що стало основою для формування вимог до розроблюваної системи.

У ході проєктування було обґрунтовано вибір архітектури програмного забезпечення, визначено функціональну структуру системи та розроблено UML-діаграми, які відображають її логіку та структуру. Також було спроектовано базу даних, що забезпечує зберігання та обробку інформації про користувачів, спортивні споруди та бронювання.

У межах програмної реалізації було створено серверну та клієнтську частини вебзастосунку, реалізовано модулі авторизації, перегляду спортивних споруд, бронювання та адміністративного керування. Використання сучасних технологій дозволило забезпечити ефективну взаємодію між компонентами системи та стабільну її роботу.

Під час тестування було перевірено основні функціональні можливості системи, що підтвердило її працездатність та відповідність поставленим вимогам. Результати тестування показали, що система працює стабільно, коректно обробляє дані та забезпечує зручну взаємодію користувача з інтерфейсом.

У результаті оцінки ефективності встановлено, що розроблена система значно спрощує процес оренди спортивних споруд, підвищує швидкість виконання операцій та зменшує ймовірність помилок. Запропоновані рекомендації щодо вдосконалення відкривають можливості для подальшого розвитку системи.

Таким чином, поставлена мета бакалаврської роботи досягнута, а всі визначені завдання виконані. Розроблена інформаційна система може бути використана на практиці для автоматизації процесів оренди спортивних споруд та має потенціал для подальшого розвитку і вдосконалення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering. – 10th ed. – Boston: Pearson, 2016. – 816 p.
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. – 8th ed. – New York: McGraw-Hill, 2015. – 970 p.
3. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Addison-Wesley, 2003. – 208 p.
4. Date C. J. An Introduction to Database Systems. – 8th ed. – Pearson, 2003. – 1024 p.
5. Elmasri R., Navathe S. Fundamentals of Database Systems. – 7th ed. – Pearson, 2016. – 1272 p.
6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley, 1994. – 395 p.
7. Django Software Foundation. Django Documentation [Електронний ресурс]. – Режим доступу: <https://docs.djangoproject.com>
8. MySQL Documentation [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc>
9. Mozilla Developer Network. Web Technologies Documentation [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org>
10. W3Schools. HTML, CSS and JavaScript Tutorial [Електронний ресурс]. – Режим доступу: <https://www.w3schools.com>
11. Кучук Г. А., Ковальчук В. М. Інформаційні системи і технології: навчальний посібник. – Київ: КНЕУ, 2018. – 312 с.
12. Бондаренко О. В. Проектування інформаційних систем: навчальний посібник. – Харків: ХНУРЕ, 2019. – 256 с.

13. Гужва В. М. Інформаційні системи в економіці: навчальний посібник. – Київ: КНЕУ, 2017. – 400 с.
14. Коваленко І. І. Базы даних: теорія та практика: навчальний посібник. – Київ: Центр навчальної літератури, 2020. – 320 с.
15. Методичні рекомендації до виконання бакалаврських робіт для студентів спеціальності «Інженерія програмного забезпечення». – Харків: ХАІ, 2024. – 40 с.
16. Larman C. Applying UML and Patterns. – 3rd ed. – Prentice Hall, 2004. – 736 p.
17. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. – 7th ed. – McGraw-Hill, 2019. – 1376 p.
18. Freeman E., Robson E. Head First Design Patterns. – O'Reilly Media, 2020. – 694 p.
19. Bootstrap Documentation [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/docs/>
20. Python Software Foundation. Python Documentation [Електронний ресурс]. – Режим доступу: <https://docs.python.org/3/>
21. React Documentation [Електронний ресурс]. – Режим доступу: <https://react.dev/>
22. Oracle. MySQL Workbench Manual [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc/workbench/en/>
23. Web-технології та Web-дизайн: Застосування мови HTML для створення електронних ресурсів // Навчальний посібник. (видання друге) / – Київ: Видавництво Ліра-К, 2020. - 212 с.
24. Бородкіна І.Л., Бородкін Г.О. Теорія алгоритмів: Навчальний посібник. – Київ : Центр учбової літератури, 2018. – 184 с.

- 25.Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення:
Навчальний посібник. – Київ : Центр учбової літератури, 2018. - 204 с.

ДОДАТОК А**ЛІСТИНГ ПРОГРАМНОГО КОДУ**

У даному додатку наведено лістинг основних програмних модулів інформаційної системи оренди спортивних споруд.

В.1. Файл models.py

```
from django.db import models

from django.contrib.auth.models import User


class Facility(models.Model):

    name = models.CharField(max_length=120, verbose_name='Назва')

    location = models.CharField(max_length=220,
verbose_name='Локація')

    price = models.IntegerField(default=0, verbose_name='Ціна за
годину')

    type = models.CharField(max_length=100, default='Спортивний
зал', verbose_name='Тип споруди')

    image = models.ImageField(upload_to='facilities/', blank=True,
null=True, verbose_name='Головне фото')

    work_time = models.CharField(max_length=40, default='08:00 -
22:00', verbose_name='Час роботи')

    area = models.CharField(max_length=50, default='800 м²',
verbose_name='Площа')

    capacity = models.CharField(max_length=50, default='до 50
осіб', verbose_name='Місткість')

    description = models.TextField(default='Сучасна спортивна
споруда для тренувань, змагань та активного відпочинку.',
verbose_name='Опис')

    conveniences = models.CharField(max_length=220,
default='роздягальні, душові, парковка', verbose_name='Зручності')


class Meta:

    verbose_name = 'Споруда'

    verbose_name_plural = 'Споруди'


def __str__(self):

    return self.name
```

```
class FacilityPhoto(models.Model):

    facility = models.ForeignKey(Facility,
on_delete=models.CASCADE, related_name='photos',
verbose_name='Споруда')

    image = models.ImageField(upload_to='facilities/gallery/',
verbose_name='Фото')

    title = models.CharField(max_length=120, blank=True,
verbose_name='Назва фото')

    is_main = models.BooleanField(default=False,
verbose_name='Зробити головним фото')

    order = models.PositiveIntegerField(default=0,
verbose_name='Порядок показу')

    class Meta:

        ordering = ['order', 'id']

        verbose_name = 'Фото споруди'

        verbose_name_plural = 'Фото споруд'

    def __str__(self):

        return self.title or f'Фото для {self.facility.name}'

class Schedule(models.Model):

    facility = models.ForeignKey(Facility,
on_delete=models.CASCADE, related_name='schedule_items',
verbose_name='Споруда')

    date = models.DateField(verbose_name='Дата')

    start = models.TimeField(default='08:00',
verbose_name='Початок')

    end = models.TimeField(default='22:00', verbose_name='Кінець')

    is_available = models.BooleanField(default=True,
verbose_name='Доступно')

    note = models.CharField(max_length=220, blank=True,
verbose_name='Примітка')
```

```
class Meta:
    ordering = ['date', 'start']
    verbose_name = 'Запис розкладу'
    verbose_name_plural = 'Розклад'

    def __str__(self):
        status = 'доступно' if self.is_available else 'недоступно'
        return f'{self.facility.name} - {self.date} {self.start}-{self.end} ({status})'

class Booking(models.Model):
    client = models.CharField(max_length=100)
    user = models.ForeignKey(User, on_delete=models.SET_NULL,
                             null=True, blank=True)
    facility = models.ForeignKey(Facility,
                                on_delete=models.CASCADE)
    date = models.DateField()
    start = models.TimeField()
    end = models.TimeField()
    status = models.CharField(max_length=30,
                              default='Підтверджено')
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'{self.client} - {self.facility}'

class Review(models.Model):
    STATUS_CHOICES = (
        ('Новий', 'Новий'),
        ('Опубліковано', 'Опубліковано'),
        ('Відхилено', 'Відхилено'),
    )
```

```

    facility = models.ForeignKey(Facility,
on_delete=models.CASCADE, related_name='reviews',
verbose_name='Споруда')

    user = models.ForeignKey(User, on_delete=models.SET_NULL,
null=True, blank=True, verbose_name='Користувач')

    author = models.CharField(max_length=120,
verbose_name='Автор')

    rating = models.PositiveSmallIntegerField(default=5,
verbose_name='Оцінка')

    text = models.TextField(verbose_name='Текст відгуку')

    status = models.CharField(max_length=30,
choices=STATUS_CHOICES, default='Новий', verbose_name='Статус')

    created_at = models.DateTimeField(auto_now_add=True,
verbose_name='Дата створення')

class Meta:

    ordering = ['-created_at']

    verbose_name = 'Відгук'

    verbose_name_plural = 'Відгуки'

def __str__(self):

    return f'{self.author} - {self.facility.name}'

```

В.2. Файл views.py

```

from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from .models import Facility, Booking, User

def facility_list(request):
    facilities = Facility.objects.all()
    return render(request, 'facilities.html', {'facilities':
facilities})

```

```
def facility_detail(request, facility_id):
    facility = get_object_or_404(Facility, id=facility_id)
    return render(request, 'facility_detail.html', {'facility':
facility})

def check_availability(facility, date, start_time, end_time):
    bookings = Booking.objects.filter(
        facility=facility,
        date=date,
        start_time__lt=end_time,
        end_time__gt=start_time,
        status='active'
    )
    return not bookings.exists()

@login_required
def create_booking(request):
    if request.method == "POST":
        facility_id = request.POST.get('facility_id')
        date = request.POST.get('date')
        start_time = request.POST.get('start_time')
        end_time = request.POST.get('end_time')

        facility = get_object_or_404(Facility, id=facility_id)

        if check_availability(facility, date, start_time,
end_time):
            Booking.objects.create(
                user=request.user,
                facility=facility,
                date=date,
                start_time=start_time,
                end_time=end_time,
                status='active'
            )
            return redirect('home')

    return redirect('home')

def register(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
```

```
        User.objects.create_user(
            username=username,
            email=email,
            password=password
        )
        return redirect('login')

    return render(request, 'register.html')

def user_login(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username,
            password=password)
        if user is not None:
            login(request, user)
            return redirect('home')

    return render(request, 'login.html')

def user_logout(request):
    logout(request)
    return redirect('home')

@login_required
def my_bookings(request):
    bookings = Booking.objects.filter(user=request.user)
    return render(request, 'my_bookings.html', {'bookings':
bookings})

@login_required
def cancel_booking(request, booking_id):
    booking = get_object_or_404(Booking, id=booking_id,
user=request.user)
    booking.status = 'cancelled'
    booking.save()
    return redirect('my_bookings')
```

В.3. Файл urls.py

```
from django.urls import path
```

```
from . import views
urlpatterns = [
    path('', views.home, name='home'),
    path('login/', views.login_page, name='login'),
    path('logout/', views.logout_page, name='logout'),
    path('register/', views.register_page, name='register'),
    path('objects/', views.objects_page, name='objects'),
    path('object/', views.first_detail, name='detail_old'),
        path('object/<int:facility_id>/', views.detail,
name='detail'),
    path('booking/', views.booking_page, name='booking_old'),
        path('booking/<int:facility_id>/', views.booking_page,
name='booking'),
        path('booking/confirm/<int:booking_id>/',
views.booking_done, name='booking_done'),
        path('my-bookings/', views.my_bookings,
name='my_bookings'),
        path('booking/cancel/<int:booking_id>/',
views.cancel_booking, name='cancel_booking'),
    path('profile/', views.profile, name='profile'),
    path('about/', views.about, name='about'),
    path('admin-panel/', views.panel, name='panel'),
        path('admin-panel/facilities/', views.panel_facilities,
name='panel_facilities'),
        path('admin-panel/facilities/add/',
views.panel_facility_create, name='panel_facility_create'),
        path('admin-panel/facilities/<int:facility_id>/edit/',
views.panel_facility_edit, name='panel_facility_edit'),
        path('admin-panel/facilities/<int:facility_id>/delete/',
views.panel_facility_delete, name='panel_facility_delete'),
        path('admin-panel/schedule/', views.panel_schedule,
name='panel_schedule'),
        path('admin-panel/schedule/<int:item_id>/delete/',
views.panel_schedule_delete, name='panel_schedule_delete'),
```



```

        path('admin-panel/users/<int:user_id>/toggle-staff/',
views.panel_user_toggle_staff, name='panel_user_toggle_staff'),
        path('admin-panel/users/<int:user_id>/delete/',
views.panel_user_delete, name='panel_user_delete'),
        path('admin-panel/bookings/', views.panel_bookings,
name='panel_bookings'),
        path('admin-panel/users/', views.panel_users,
name='panel_users'),
        path('admin-panel/reviews/', views.panel_reviews,
name='panel_reviews'),
        path('admin-panel/reviews/<int:review_id>/<str:status>/',
views.panel_review_status, name='panel_review_status'),
    ]

```

В.4. Файл admin.py

```

from django.contrib import admin
from django.utils.html import format_html
from .models import Facility, FacilityPhoto, Schedule, Booking,
Review

class FacilityPhotoInline(admin.TabularInline):
    model = FacilityPhoto
    extra = 1
    fields = ('preview', 'image', 'title', 'is_main', 'order')
    readonly_fields = ('preview',)

    def preview(self, obj):
        if obj.image:
            return format_html('',
obj.image.url)
        return 'Немає фото'
    preview.short_description = 'Перегляд'

@admin.register(Facility)

```

```

class FacilityAdmin(admin.ModelAdmin):
    list_display = ('name', 'type', 'location', 'price',
'main_preview')
    list_filter = ('type', 'location')
    search_fields = ('name', 'type', 'location')
    inlines = [FacilityPhotoInline]

    def main_preview(self, obj):
        photo = obj.photos.filter(is_main=True).first() or
obj.photos.first()
        image = photo.image if photo else obj.image
        if image:
            return format_html('', image.url)
        return 'Немає фото'
    main_preview.short_description = 'Фото'

@admin.register(FacilityPhoto)
class FacilityPhotoAdmin(admin.ModelAdmin):
    list_display = ('facility', 'title', 'is_main', 'order',
'preview')
    list_filter = ('facility', 'is_main')
    search_fields = ('facility__name', 'title')

    def preview(self, obj):
        if obj.image:
            return format_html('',
obj.image.url)
        return 'Немає фото'
    preview.short_description = 'Перегляд'

@admin.register(Schedule)

```

```

class ScheduleAdmin(admin.ModelAdmin):
    list_display = ('facility', 'date', 'start', 'end',
'is_available', 'note')
    list_filter = ('facility', 'date', 'is_available')
    search_fields = ('facility__name', 'note')

@admin.register(Booking)
class BookingAdmin(admin.ModelAdmin):
    list_display = ('client', 'facility', 'date', 'start',
'end', 'status')
    list_filter = ('facility', 'date', 'status')
    search_fields = ('client', 'facility__name')

@admin.register(Review)
class ReviewAdmin(admin.ModelAdmin):
    list_display = ('author', 'facility', 'rating', 'status',
'created_at')
    list_filter = ('facility', 'rating', 'status')
    search_fields = ('author', 'facility__name', 'text')
    list_editable = ('status',)

```

В.5. Фрагмент шаблону facilities.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Спортивні споруди</title>
</head>
<body>
    <h1>Список спортивних споруд</h1>

    {% for facility in facilities %}
        <div>
            <h2>{{ facility.name }}</h2>
            <p>Тип: {{ facility.type }}</p>
            <p>Місцезнаходження: {{ facility.location }}</p>
            <p>Вартість: {{ facility.price }} грн</p>
            <a href="{% url 'facility_detail' facility.id

```

```

    %}">Детальніше</a>
    </div>
    <hr>
    {% endfor %}
</body>
</html>

```

В.6. Фрагмент шаблону facility_detail.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>{{ facility.name }}</title>
</head>
<body>
    <h1>{{ facility.name }}</h1>
    <p>Тип: {{ facility.type }}</p>
    <p>Місцезнаходження: {{ facility.location }}</p>
    <p>Опис: {{ facility.description }}</p>
    <p>Вартість: {{ facility.price }} грн</p>

    <form method="post" action="{% url 'create_booking' %}">
        {% csrf_token %}
        <input type="hidden" name="facility_id" value="{{
facility.id }}">
        <label>Дата:</label>
        <input type="date" name="date" required><br>
        <label>Час початку:</label>
        <input type="time" name="start_time" required><br>
        <label>Час завершення:</label>
        <input type="time" name="end_time" required><br>
        <button type="submit">Забронювати</button>
    </form>
</body>
</html>

```

В.7. Фрагмент шаблону login.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Авторизація</title>
</head>
<body>
    <h1>Вхід до системи</h1>

```

```
<form method="post">
  {% csrf_token %}
  <label>Логін:</label>
  <input type="text" name="username" required><br>
  <label>Пароль:</label>
  <input type="password" name="password" required><br>
  <button type="submit">Увійти</button>
</form>
</body>
</html>
```

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Литвиненко Данило Сергійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення інформаційної системи оренди спортивних споруд» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - ☐ інструменти генеративного ШІ не використовувалися.
 - ☐ інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - ☐ перекладу іншомовних джерел;
 - ☐ стилістичного редагування та перевірки граматики;
 - ☐ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«__» _____ 2026 р. _____ Данило ЛИТВИНЕНКО
(підпис)