Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ПО АДМІНІСТРУВАННЮ ФУТБОЛЬНИХ ПОЛІВ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-1

Мальцев М.Є.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мальцеву Миколі Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного засобу по адмініструванню футбольних полів

затверджена наказом по університету від «16» травня 2022 року № 541 Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2022 р.

3. Вихідні дані до роботи: фреймворк Vaadin 14+, фреймворк Spring, Maven, СУБД PostgreSQL, мова програмування Java, середовище розробки програмних систем IntelliJ IDEA

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд наявних програмних рішень та знаходження їх недоліків
2. Вимоги до застосунку з адміністрування
3. Розробка інформаційної системи для адміністрування полів
4. Розробка серверної частини застосунку
5. Розробка клієнтської частини застосунку
6. Реалізація безпеки застосунку
7. Ролі користувачів у застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Ілюстрації макетів сторінок застосунку, ілюстрація схеми бази даних, скріншоти роботи застосунку, скріншоти класів, ілюстрації архітектури фреймворків.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на до атестаційної роботи	18.04.2022	
2	Аналіз завдання, аналіз літератури	18.04.22-22.04.22	
3	Аналіз літератури	23.04.22-24.04.22	
4	Огляд існуючих програмних рішень	25.04.22-27.04.22	
5	Розробка інформаційної системи	28.04.22-06.05.22	
6	Програмна реалізація	07.05.22-21.05.22	
7	Оформлення пояснювальної записки	22.05.22-26.05.22	
8	Перевірка на плагіат	31.05.22	
9	Рецензування	31.05.22	
10	Підготовка презентації та доповіді	30.05.22-31.05.22	
11	Занесення роботу у електронний архів	31.05.22	
12	Попередній захист атестаційної роботи	01.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ д.т.н., проф. Машталір В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 69 с., 35 рис., 1 дод., 30 джерел.

ФУТБОЛЬНЕ ПОЛЕ, БРОНЮВАННЯ, АДМІНІСТРУВАННЯ, CRUD, REST, VAADIN, SPRING.

Об’єктом роботи є система оренди футбольних полів.

Метою роботи є розробка CRUD вебзастосунку для автоматизації процесу адміністрування футбольних полів, спрощення роботи адміністратора та користувачів, підвищення якості та ефективності обслуговування користувачів, з точки зору ведення бізнесу – прискорення процесу аналізу даних.

Проведено аналіз предметної області в галузі футбольних полів, виявлено слабкі місця існуючих рішень, спроектовано базу даних та розроблено працездатну інформаційну систему для адміністрування полів.

У результаті роботи здійснена програмна реалізація системи адміністрування футбольних полів.

FOOTBALL FIELD, BOOKING, ADMINISTRATION, CRUD, REST, VAADIN, SPRING.

The object of the research is a system of soccer fields renting.

The aim of the work is to develop a CRUD web application to automate the process of administration of football fields, simplify the work of administrators and users, improve the quality and efficiency of customer service, in terms of doing business - speed up the data analysis process.

An analysis of the subject area in the field of football fields was carried out, weaknesses of existing solutions were identified, a database was designed and a workable information system for field administration was developed.

As a result, the software implementation of the football field administration system was implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CRUD – create, read, update, delete

БД – база даних

ВСТУП

Перехід до цифрової економіки – це необхідна вимога сьогоднішнього дня. Діджиталізація економіки дає можливість людині полегшити вирішення багатьох завдань, пов’язаних з роботою, з пошуком інформації, з якими він неодноразово стикається. Діджиталізація відкриває для людини широкі можливості в розвитку бізнесу. Особливе значення тут набувають комунікативні можливості цифрових каналів. Велика швидкість, зручність дозволило появи такого виду маркетингу, як digital-маркетинг [1].

Цифровий маркетинг (англ. Digital marketing, Діджитал-маркетинг) – загальний термін, який використовується для позначення таргетованого та інтерактивного маркетингу, товарів і послуг, що використовує цифрові технології для залучення потенційних клієнтів і утримання їх в якості споживачів. Головними завданнями цифрового маркетингу є просування бренду і збільшення збуту за допомогою різних методик. Цифровий маркетинг містить в собі великий вибір маркетингових тактик з просування товарів, послуг і брендів. Крім мобільних технологій, традиційних ТБ і радіо методи цифрового маркетингу використовують інтернет в якості основного комунікаційного посередника [2].

Залежно від виду діяльності, успіх у бізнесі може визначатися різними характеристиками, у тому числі вказаними нижче.

– збільшення часу, що користувачі витрачають на сервіс;
– зменшення показників відмов для лідів;
– підвищення коефіцієнтів конверсії;
– збільшення кількості відвідувачів, що повертаються.

Більшість проектів прогресивних вебзастосунків призводить до підвищення коефіцієнта конверсії для мобільних пристроїв. Додаткову інформацію можна вивчити у численних прикладах. Залежно від цілей можна зробити більш пріоритетними ті аспекти прогресивних вебзастосунків, які

мають більше значення для бізнесу, і це нормально. Функції прогресивних вебпрограм можна вибрати і запускати окремо [3].

Актуальність роботи полягає у тому, що в даний час адміністратори оперують дуже великими об’ємами даних. Кожен день здійснюється більше ста записів різними людьми та у різний час. І, завдяки подібним застосункам для адміністрування, можна зменшити ризик створення помилки адміністратором чи користувачем, що оптимізує та підвищить якість ведення бізнесу з оренди полів.

Тому, виходячи з актуальності та комерційного потенціалу для бізнесу, було вирішено присвятити тему дипломної роботи «Розробці програмного засобу по адмініструванню футбольних полів». Створивши за допомогою мови програмування Java, вебзастосунок, який буде являти собою CRUD застосунок для додавання, читання, оновлювання та видалення даних з бази даних.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Проект World Wide Web зародився на початку 90-х років у швейцарському інституті CERN. Спочатку він задумувався, як засіб, що полегшує пошук бібліотечної інформації, розподіленої між декількома серверами в CERN. Для цього він використовував концепцію «гіпертексту», розроблену раніше Тедом Нельсоном і Ванневаром Бушем [4].

У WWW було кілька важливих відмінностей від інших систем гіпертексту, які тоді вже існували (Note Code, HyperCard, Gopher):

– в WWW використовувалися тільки односпрямовані посилання, що дозволяло посилатися на інші ресурси без будь-яких дій з боку власника цього ресурсу;

– WWW не належав будь-якій компанії, тому можна було створювати програмне забезпечення незалежно і без всяких ліцензійних обмежень.

Ці риси зіграли велику роль у вибуховому зростанні популярності WWW в середині 90-х років, коли вийшов веббраузер Mosaic і наступний за ним комерційний аналог Netscape Navigator. WWW був побудований на трьох основних стандартах: URL (RFC2396), HTTP (RFC2616) і HTML. Наймолодший з них і, отже найбільш розвинений – HTML. Спочатку він був задуманий виключно для семантичної розмітки тексту (посилання, розділи, цитати та інше), потім у зв’язку з комерціалізацією до цього стандарту стали додаватися різні, часто не сумісні один з одним теги для художнього оформлення тексту [5]. Надалі ця тенденція була названа неправильною (з причин відсутності стандартизації та відсутності поділу смислової й оформлювальної частини документації) і консорціумом W3C, який розробляє стандарт HTML, був узятий зворотний курс – на залишення в HTML тільки семантичної розмітки, для оформлення ж був придуманий спеціальний

стандарт CSS (каскадні списки стилів).

Виходячи з цього потрібно сформулювати поняття вебвузла або говорячи по іншому вебсторінки.

1.1.1 Структура мережі інтернет

Вебвузол (вебсайт) – це комплекс вебсторінок та інших ресурсів, об’єднаних відповідно за змістом, розміщених в на одному домені.

Мережа Internet складається з великого числа з’єднаних між собою комп’ютерів, маршрутизаторів та іншого, необхідного для правильної роботи. Кожен елемент мережі Internet (вузол) має неповторний ключ – IP-адресу. Знаючи IP-адресу вузла, є можливість спробувати під’єднатися до нього, а маючи невеликі навички можна з’ясувати, кому ця адреса належить і в якому регіоні світу знаходиться [6]. IP-адреси заведено записувати у вигляді чотирьох груп цифр, розділених крапками, наприклад: 192.168.100.003 або 10.10.0.123.

Запам’ятати адреси всіх часто відвідуваних сторінок практично неможливо, для цього в мережі Internet існують спеціальні сервери DNS (Domain Name System), на яких виконується зберігання всіх списків зіставлення IP-адрес і символьних імен. Завдяки цьому, до серверів, користувач завжди переходить по потрібній IP-адресі, набравши в браузері тільки ім’я сторінки [7].

Після того, як було введено в рядок браузера ім’я необхідної сторінки, браузер автоматично отримує з DNS IP-адресу потрібного сервера і шле за цією адресою спеціальний запит на отримання сторінки (HTTP-запит). Спеціальна програма, що працює на сервері (вебсервер) обробляє цей запит і відправляє назад в браузер необхідну сторінку.

Очевидно, що всі дії по відображенню сторінки можна розділити на дві категорії, що:

– виконуються на стороні клієнта (клієнтський код або front-end);

– виконуються на стороні сервера (серверний код або back-end).

Бекенд і фронтенд терміни в програмній інженерії, які розрізняють згідно з принципом поділу відповідальності між зовнішнім поданням і внутрішньою реалізацією відповідно.

При цьому сервер зовсім нічого не знає про поточний стан клієнта, а клієнт – про стан сервера.

При розробці алгоритмів обміну потрібно завжди знати про це і своєчасно пересилати потрібні дані, що описують поточний стан або необхідні дії [8].

Залежно від середовища використання розрізняються і засоби реалізації частин. На стороні клієнта використовується зазвичай тільки HTML, JavaScript(AJAX), CSS, Flash. Важливо відрізняти вебзастосунки від звичайних вебсайтів, адже це хоч і схожі, але абсолютно різні вебструктури.

Вебзастосунок – клієнт-серверний застосунок, в якому клієнтом виступає браузер, а сервером – вебсервер. Логіка вебзастосунку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається через мережу. Однією з переваг такого підходу є той факт, що клієнт не залежать від конкретної операційної системи користувача, тому вебзастосунки є кроссплатформенними сервісами.

1.1.2 Принцип роботи та структура вебзастосунків

Вебзастосунки дозволяють відвідувачам швидко і легко знаходити необхідну інформацію на вебсайтах з великим обсягом інформації.

Даний вид вебзастосунків дозволяє здійснювати пошук у вмісті, впорядковувати вміст і переміщатися по ньому зручним для відвідувачів способом.

Вебзастосунок дозволяє зберігати дані безпосередньо в базі даних, а

також отримувати дані та формувати звіти на основі отриманих даних для аналізу. Як приклад можна навести інтерактивні сторінки банків, сторінки для контролю товарних запасів, соціологічні дослідження та опитування, а також форми для зворотного зв’язку з користувачами.

Найчастіше вебзастосунки складаються як мінімум з трьох основних компонентів:

– клієнтська частина – це те, що бачить користувач на сторінці (графічний інтерфейс). Графічний інтерфейс відображається в браузері. Користувач взаємодіє з вебзастосунком через браузер, клікаючи по посиланнях і кнопках;

– серверна частина – це програма або скрипт на сервері, що обробляє запити користувача (точніше, запити браузера). Найчастіше серверна частина вебзастосунку програмується на PHP або Java. При кожному переході користувача за посиланням браузер відправляє запит до сервера. Сервер обробляє цей запит, викликаючи деякий скрипт, який формує вебсторінку, описану мовою HTML, і відсилає Клієнту по мережі. Браузер тут же відображає отриманий результат у вигляді чергової вебсторінки;

– база даних (БД або СУБД) – програмне забезпечення на сервері, що займається зберіганням даних і їх видачою в потрібний момент. У разі форуму або блогу, збережені в БД дані – це пости, коментарі, новини, і так далі. База даних розміщується на сервері. Серверна частина вебзастосунку (PHP або Java код) звертається до бази даних, витягуючи дані, які необхідні для формування сторінки, запитаної користувачем. Браузер через Інтернет відсилає HTTP-запити вебсерверу. Вебсервер викликає код, написаний розробником вебпрограми. Написаний скрипт звертається до бази даних, якщо це потрібно. В результаті скрипт повертає клієнту вебсторінку, яку і відображає браузер [9].

Основна мова, якою програмується графічний інтерфейс вебзастосунків – це HTML. Дана мова описує структуру вебсторінки, розміщення на ній компонентів.

Оформлення вебсторінок, їх стиль і колірна схема описуються в

таблицях стилів – CSS. Для «оживлення» графічного інтерфейсу, надання йому динамічності, використовуються додаткові технології: скрипти JavaScript, а також вбудовані в вебсторінку компоненти, створені на Flash, Java або Silverlight [10].

Відсутність необхідності повністю перезавантажувати сторінку після кожного отримання даних від сервера може істотно прискорити роботу вебзастосунки. Така концепція має назву Asynchronous JavaScript and XML (асинхронний JavaScript і XML, Ajax). При використанні даного підходу динамічні запити до сервера відбуваються без видимого перезавантаження вебсторінки: користувач не помічає, коли його браузер запитує дані [11].

Згідно з класичним визначенням, база даних – це впорядкована сукупність інформації, що зберігається у вигляді множин, кожна з яких містить записи уніфікованого виду. Системи управління базами даних (СУБД) надають програмісту найпотужніший інструментарій для створення, оновлення та обробки великих обсягів інформації, що має складну структуру. У класичній теорії виділяють три типи, три структури баз даних: ієрархічну, мережеву і реляційну. В даний час домінуюче положення займають реляційні бази даних. Лідером серед баз даних, що застосовуються для розробки вебзастосунків, на сьогодні, безумовно, є MySQL. Головна перевага MySQL (плавно переходить в недолік:) – її простота. Як наслідок – найвища швидкість виконання SQL-запитів і необхідність явного програмування основних правил підтримки цілісності і несуперечливості даних на рівні сервера [12].

Серед інших баз даних, що застосовуються для веброзробок, відзначаються Oracle і PostgreSQL. PostgreSQL – це вільно розповсюджувана СУБД з відкритим вихідним кодом, орієнтована головним чином на роботу в UNIX-подібних системах [13].

Найбільш відомі приклади вебзастосунків: вебпошта (Http://gmail.com), інтернет-магазини (http://amazon.com), онлайн аукціони (http://ebay.com). Однак область застосування вебзастосунків набагато ширша, ніж електронний бізнес, вони використовуються у багатьох наукових і комерційних областях.

Хоча можливі різні варіації, найчастіше використовується трирівнева архітектура для побудови вебзастосунків: веббраузер, якась технологія надання динамічного вебконтенту і база даних. Веббраузер посилає запити середнього рівня, який обслуговує їх роблячи запити до бази даних і представляючи результати в користувацький інтерфейс [14].

Наприклад, розглянемо найпопулярнішу на сьогоднішній день платформу для вебзастосунків LAMP – Linux, Apache, MySQL, PHP. Тут роль середнього рівня відіграє вебсервер Apache з встановленим на ньому модулем підтримки мови програмування PHP, а в якості бази даних виступає MySQL.

1.2 Переваги та недоліки вебзастосунків

Переваги:

– незалежність від клієнтської, а часто і серверної платформи;

– невимогливість до ресурсів клієнта;

– не вимагає встановлення на клієнтському комп’ютері будь-якого програмного забезпечення крім браузера;

– «вбудовані» мережеві можливості (можливість працювати з декількома клієнтами одночасно);

– зберігаються всі дані при переході клієнта з машини на машину.

– Недоліки та способи їх обходу:

– низький час реакції – орієнтація на клієнтські технології, типу JavaScript, DOM, Flash, XUL;

– протокол HTTP не зберігає стану – механізм cookies, сесії;

– мала захищеність – захищені з’єднання HTTPS, авторизація вбудована в HTTP або на основі сесій;

– нерозвиненість мови HTML в сенсі форм – розробка стандарту XForms, Flash, Java applets [15].

Виходячи з цієї інформації, можемо підсумувати. Як вже було зазначено раніше, одна вебпрограма може зв’язатися з усіма пристроями. Звичайно, вебсайт чи застосунок повинен бути запрограмований таким чином, щоб його можна було показати незалежно від операційної системи пристрою. Якщо це не адаптивний вебсайт, то можуть виникнути проблеми з його відображенням на iOS, Android або Windows Phone. Останній вже майже і не підтримується. В такому випадку бізнесу доведеться витратити гроші, які мали на меті заощадити на розробці програми, на покращення вебсайту. Немає користі створювати вебпрограму, якщо не адаптувати свій вебсайт так, щоб його можна було якісно показати на будь-якому пристрої.

1.3 Сучасний стан та перспективи

Сьогодні вебзастосунки продовжують набирають популярність. Найбільш відвідуваними сайтами стають не чисто інформаційні, гіпертекстові сайти, а ті, які надають будь-який сервіс, яку-небудь взаємодіють з користувачем. Але навіть і звичайні інформаційні сайти часто використовують системи управління контентом для зручності управління інформацією, так що і їх теж можна зарахувати до вебзастосунків. Деякі вже ставлять веб в один ряд з Windows, Linux та іншими в якості нової платформи для виконання застосунків [16].

1.4 Огляд наявних програмних рішень

Розробка CRUD вебзастосунку неможлива без аналізу наявних рішень і накопиченого досвіду в цій сфері. Перш за все потрібно в деталях проаналізувати і зрозуміти, що собою являє CRUD вебзастосунок.

В рамках комп’ютерного програмування абревіатура CRUD означає створення, читання, оновлення та видалення. Це чотири основні функції всіх подібних застосунків. Крім того, кожна буква в абревіатурі може посилатися на всі функції, що виконуються в реляційних програмах баз даних і зіставляються зі стандартним методом HTTP, оператором SQL або операцією DDS [17].

CRUD також описує контракт на встановлення зв’язку з користувацьким інтерфейсом, що дозволяє переглядати, шукати та змінювати інформацію за допомогою зручного інтерфейсу. Сутності зчитуються, створюються, оновлюються та видаляються. Ці самі сутності можна змінити, взявши дані зі сервісного шару (частина архітектури) перед тим, як відправити дані назад до сервісу для оновлення. Крім того, CRUD орієнтований на дата класи та стандартизоване використання методів HTTP.

Більшість програм мають свою реалізацію функцій CRUD. Насправді кожному програмісту в певний момент доводиться стикатися з CRUD. Не кажучи вже про те, що застосунок CRUD – це програма, яка використовує форми для отримання та повернення даних із бази даних, що є основним при роботі з базами даних [18].

Перше посилання на операції CRUD було зроблено Хаїмом Кіловим у 1990 році у статті під назвою «Від семантичного до об’єктно-орієнтованого моделювання даних». Однак цей термін вперше став популярним у книзі Джеймса Мартіна 1983 року, «Управління середовищем бази даних». Основні принципи:

– CREATE : виконує оператор INSERT для створення нового запису;

– READ: зчитує записи таблиці на основі первинного ключа з вхідним параметром;

– UPDATE: виконує оператор UPDATE в таблиці на основі вказаного первинного ключа для запису в вираз спільно з оператором WHERE;

– DELETE: видаляє вказаний рядок у виразі спільно з оператором WHERE.

Залежно від вимог системи у різних користувачів можуть бути різні цикли CRUD. Клієнт може використовувати CRUD для створення облікового запису і доступу до нього при поверненні на певний сайт. Потім користувач може оновити персональні дані або змінити платіжну інформацію [19].

В епоху Web 2.0 операції CRUD лежали в основі більшості динамічних вебсайтів. Однак потрібно відрізняти CRUD від методів HTTP. Наприклад, якщо потрібно створити новий запис, то необхідно використовувати метод «POST». Для оновлення запису необхідно використовувати «PUT» або «PATCH». Якщо потрібно видалити запис, то варто скористатися «DELETE» методом. Та ці правила скоріше є думкою деякої частини програмістів. Існує й інше твердження та навіть поняття, що хорошим тоном програмування, буде використовувати тільки POST метод для передачі даних. При цьому реалізуючи свою окрему логіку для кожного метода, що взаємодіє через POST.

Розробник застосунків має безліч варіантів виконання операцій CRUD. Одним з найбільш ефективних варіантів є створення набору збережених процедур в SQL для виконання операцій [20]. Іншими словами регулярний вираз SQL. Що стосується збережених процедур CRUD є кілька загальних конвенцій про імена:

– ім’я процедури має закінчуватися реалізованим ім’ям операції CRUD;

– префікс не повинен збігатися з префіксом, використаним для інших користувацьких збережених процедур;

– процедури CRUD для однієї й тієї ж таблиці повинні бути згруповані разом, якщо використовується ім’я таблиці після префікса.

Після додавання процедур CRUD можна оновити схему бази даних, визначивши сутність бази даних, в якій будуть реалізовані операції CRUD.

CRUD запобігає випадковому перегляду і змінам [21]. Ролі додатків – це метод SQL Server, який дозволяє коду перемикати ідентифікатори без повідомлення користувачу. Для роботи зі спеціальними інструкціями SQL користувачі повинні мати необхідні дозволи для таблиць бази даних. Після отримання дозволу користувачі можуть читати і маніпулювати даними в таких

програмах, як Excel, Word та інших. Користувачі можуть навіть обійти бізнес-правила програми.

Проте, це небажана ситуація, яку можна запобігти за допомогою ролі програми. Завдяки інтегрованій безпеці доступу до бази даних через ролі програми, ці типи лазівок можуть бути закриті. Після додавання ролі, надається доступ і призначається пароль. Пароль також закодований в застосунку, що ускладнює його зміну. Для маніпулювання даними слід використовувати метод CRUD [22].

Проаналізувавши велику кількість інформації і спробувавши знайти схожі рішення, був сформульований висновок, що на даному етапі, не можливо знайти якийсь застосунок або якесь програмне рішення, яке хоча б частково могло задовольнити вирішення поставленої задачі. Основна складність пошуку подібних реалізацій полягає в тому, що вони створюються для задоволення потреби конкретного бізнесу. І виходячи з цього, код або архітектура цих програмних рішень не може бути розміщена для публічного доступу. Однак, отриманої інформації достатньо для того, щоб з упевненістю почати етап вибору методів розробки для реалізації та проектування саме CRUD вебзастосунку, який повинен повністю задовольнити поставлені бізнес задачі, а саме: бути крос платформенним, мати автентифікацію та авторизацію, простий та зрозумілий інтерфейс і при цьому реалізовувати CRUD операції.

1.5 Постановка задачі

Об’єктом роботи є система оренди футбольних полів.

Метою роботи є розробка CRUD вебзастосунку для автоматизації процесу адміністрування футбольних полів, спрощення роботи адміністратора та користувачів, підвищення якості та ефективності обслуговування

користувачів, з точки зору ведення бізнесу – прискорення процесу аналізу даних.

Для досягнення цілей автоматизації задач треба виконати наступні задачі:

– провести аналіз предметної області в галузі оренди футбольних полів;

– виявити слабкі місця у вже існуючих подібних застосунках;

– розробити інформаційну та працездатну систему для адміністрування полів;

– розробити БД з необхідними SQL-запитами;

– розробити програмне засобу на основі вищеописаних даних.

Призначенням реалізації цього програмного засобу є:

– введення, редагування, оновлення та видалення даних про користувачів, поля, та бронювання застосунку;

– перегляд інформації про користувачів, поля та бронювання застосунку.

2 ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Структура та технології реалізації

Враховуючи специфіку маркетингової індустрії, для якої розробляється вебзастосунок, процес переходу до його використання може виявитися складним і доволі тривалим, якщо говорити про реальне використання на справжньому проекті, де повинен бути плавний перехід з тестуванням кожного модуля та налагодженням системи (рис. 2.1).

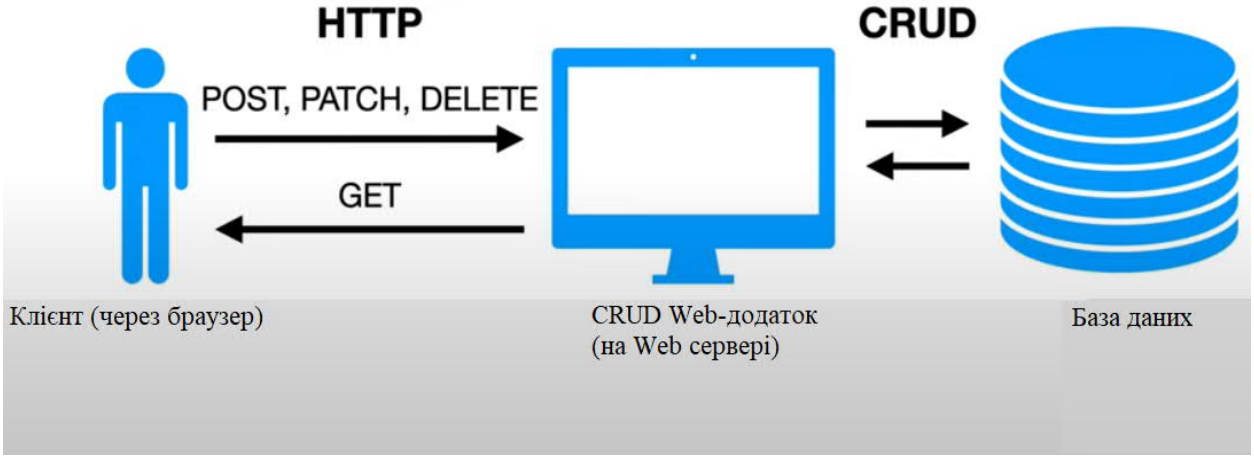

Рисунок 2.1 – Структура застосунку, що має бути реалізована

Для написання застосунку, було обрано мову програмування Java. Мова Java спочатку позиціонувався розробником, як мова для інтернету. З десктопними застосунками у Java не склалося, але зате на стороні сервера Java застосовується дуже широко і має величезну кількість різних цікавих можливостей. Перш за все існує кілька конкуруючих серверів вебзастосунків, які незважаючи на відмінності дотримуються деяких стандартів, встановлених Sun, а значить більшість застосунків без будь-яких значних модифікацій можуть бути перенесені з сервера на сервер. Крім того, існує кілька різних рівнів складності і з різними підходами фреймворків для розробки

вебзастосунків (тобто бібліотек класів, на основі яких будується вебзастосунок). Це фреймворки для структурування застосунків на основі патерну MVC (Struts, Spring), бібліотеки для побудови шаблонів вебсторінок (JSTL, Velocity, Java ServerFaces), бібліотеки для відображення реляційної таблиці на об’єкти і назад (Hibernate).

Саме такий потужний фреймворк, як Spring був обраний для розробки основного функціоналу та модуля безпеки. В сучасних ентерпрайз проектах важко уявити java проект, який міг би обійтися без використання Spring. І говорячи про цей фреймворк, потрібно розуміти, що це не один клас чи технологія. Він складається з певної кількості пакетів не залежних один він одного і об’єднаних під однією назвою. Перш ніж приступити до проектування, потрібно проаналізувати і зрозуміти, які пакети можуть бути використані.

Spring Framework (або коротко Spring) – універсальний фреймворк з відкритим кодом для Java-платформи. Він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні застосунків ентерпрайз масштабу [23]. Крім того, особливості ядра Spring можуть бути застосовні в будь-якій Java програмі. Також існує безліч розширень і удосконалень для побудови вебзастосунків на Java Enterprise платформі. Саме цим Spring заробив велику популярність і визнається розробниками як стратегічно важливий фреймворк (рис. 2.2).

Одним з компонентів фреймворку є Spring Security, який являє собою інструмент, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості реалізації безпеки для корпоративних застосунків, створених з допомогою Spring Framework.

Основні характеристики:

– аутентифікація та авторизація користувачів;

– захист від атак, таких як фіксація сесії, підробка міжсайтових запитів і так далі;

– можливість інтеграції з Servlet API;

– опціонально підключається модуль Spring Web MVC.

Рисунок 2.2 – Схематичне зображення основних технологій Spring

Spring Boot – комплексний фреймворк для створення і запуску застосунків з мінімальними зусиллями і налаштуваннями [24]. Цей модуль ділиться на два стеки: заснований на API сервлетах Spring MVC і реактивний Spring WebFlux.

Spring WebFlux – вебплатформа, створена, щоб по максимуму використовувати переваги сучасних багатоядерних процесорів і обробляти величезну кількість одночасних підключень [25].

Spring MVC побудований на API сервлетах і використовує архітектуру синхронного вводу-виводу з моделлю «один запит на потік». У Spring Boot також можна опціонально підключити бібліотеку Reactor для створення реактивних систем на JVM.

Основні характеристики:

– вбудовані контейнери Tomcat, Jetty або Undertow, що працюють безпосередньо без розгортання War-файлів;

– готові стартові залежності, що спрощують конфігурацію збірки;

– можливість конфігурувати проект прямо в браузері за допомогою Spring Initializr;

– автоматичне налаштування сторонніх бібліотек (по можливості);

– готові до роботи функції, такі як збір метрик, перевірка працездатності і використання зовнішньої конфігурації;

– немає кодогенерації та не потрібна конфігурація XML – все конфігурується через анотації.

Разом з Spring Boot в проектах зазвичай використовуються Spring Security і Cloud. За допомогою Spring Boot можна створювати:

– мікросервіси;

– реактивні системи;

– вебзастосунки.

Spring Data – модуль забезпечує застосункам доступ до даних через реляційні та нереляційні бази даних, map-reduce фреймворки і хмарні сервіси. Spring Data містить безліч підпроектів, призначених для певних СУБД. Серед них є, наприклад, MySQL, MongoDB, Redis і багато інших. Також можна використовувати підмодуль, розроблений спільнотою Spring для більш специфічних баз даних на кшталт ArangoDB, Google Datastore, Microsoft Azure Cosmos DB і других [26].

Основний механізм, що реалізується в Spring Data – репозиторій. Це набір інтерфейсів, що використовують JPA Entity для взаємодії з даними.

Основні характеристики:

– налаштування відображення сутностей в БД на Java-об’єкти;

– створення динамічних запитів в базу даних;

– базові класи для різних завдань;

– прозорий аудит об’єктів;

– можливість інтегрувати власний код репозиторію;

– проста інтеграція з Spring через JavaConfig, а також кастомних просторів імен XML;

– розширена інтеграція з контролерами Spring MVC.

Spring Data використовується скрізь, де потрібен доступ до даних, і легко інтегрується з іншими модулями Spring.

Spring Cloud – з Spring Cloud можна легко і швидко створювати шаблони в розподілених системах. З прикладів таких шаблонів: управління конфігурацією, виявлення сервісів, інтелектуальна маршрутизація, мікропроксі, одноразові токени і багато іншого. Шаблони, створені за допомогою Spring Cloud, будуть добре працювати в будь-якому розподіленому середовищі, включаючи ноутбук, центри обробки даних і PaaS-платформи, такі як Cloud Foundry [27].

Spring Cloud також складається з безлічі підпроектів для різних цілей. Так, Spring Cloud Azure інтегрує Spring зі службами Azure. Spring Cloud Stream використовується для створення керованих подіями мікросервісів (event-driven microservices) і так далі.

Основні характеристики:

– розподілена конфігурація;

– реєстрація та виявлення сервісів;

– маршрутизація;

– зв’язок між сервісами (service-to-service calls);

– балансування навантаження;

– вибір лідера і стан кластера;

– розподілений обмін повідомленнями;

В загальному Spring Cloud містить багато корисних інструментів для мікросервісів і розподілених систем.

Spring Integration дозволяє полегшити обмін повідомленнями в застосунках на основі Spring, підтримує інтеграцію з зовнішніми системами і дає інструменти для обробки даних з різних джерел. Один з підпроектів Spring

Cloud, Spring Cloud Stream, використовує Spring Integration як двигун для мікросервісів, керованих подіями.

Основні характеристики:

– багато шаблонів для інтеграції застосунків підприємства;
– інтеграція з зовнішніми системами;
– вебсервіси (SOAP і REST);
– велика підтримка JMX;
– Mbeans-компоненти.

Spring Integration підключається до проекту, якщо потрібно зв’язати POJO (Plain Old Java Object) за допомогою парадигми обміну повідомленнями без впровадження залежностей. Також Integration дозволяє взаємодіяти із зовнішніми системами за допомогою адаптерів, каналів і шлюзів. Адаптери каналів використовуються для односторонньої інтеграції (відправлення або отримання), а шлюзи – для сценаріїв запиту / відповіді (вхідного або вихідного).

Spring Batch – платформа для розробки пакетних застосунків. Spring Batch підійде як для простих, так і для складніших проектів – платформа легко масштабується і може обробляти великі обсяги інформації.

Основні характеристики:

– управління транзакціями;
– обробка на основі фрагментів даних;
– декларативне введення;
– вебінтерфейс адміністрування (Spring Cloud Data Flow).

Spring Batch підійде для застосунків з багаторазово використовуваними функціями, щоб обробляти великі обсяги записів. Серед таких функцій – ведення логів і трасування, управління транзакціями, статистика обробки завдань, перезапуск і пропуск завдань, управління ресурсами та інші.

Також не можливо не згадати систему автоматичного збирання проекту Maven, адже не можливо сьогодні уявити проект java без цього програмного

модуля. Maven – це інструмент для збирання Java проекту: компіляції, створення jar, створення дистрибутива програми, генерації документація.

Переваги Maven:

– незалежність від ОС. Збірка проекту відбувається в будь-який операційній системі;

– управління залежностями. Рідко які проекти пишуться без використання сторонніх бібліотек (залежностей). Ці сторонні бібліотеки часто теж в свою чергу використовують бібліотеки різних версій. Мавен дозволяє управляти такими складними залежностями. Саме це дозволяє вирішувати конфлікти версій і в разі необхідності легко переходити на нові версії бібліотек;

– можлива збірка з командного рядка. Таке часто необхідно для автоматичного складання проекту на сервері (Continuous Integration);

– хороша інтеграція з середовищами розробки. Основні середовища розробки на java легко відкривають проекти, які збираються за допомогою maven. При цьому найчастіше проект налаштовувати не потрібно – він відразу готовий до подальшої розробки. Як наслідок – якщо з проектом працюють в різних середовищах розробки, то maven зручний спосіб зберігання налаштувань;

– файл налаштування середовища розробки і для збірки один і той же, а це значить, що буде менше дублювання даних і відповідно помилок;

– декларативний опис проекту.

Вище були розглянуті основні модулі Spring. Насправді їх набагато більше. Є ще Spring for Android для створення Android-застосунків, Spring CredHub для взаємодії з CredHub-сервером, Spring LDAP і багато інших.

Також є багато корисних та зручних фреймворків для побудови клієнтської сторони програмного засобу на мові Java. Одним з найкращих є фреймворк Vaadin.

У клієнт-серверній архітектурі місце Java-програми переважно на серверній стороні, при цьому вебінтерфейс пишеться окремою групою фронт-

енд розробників на JavaScript. Java не пропонує зручних засобів для створення сучасного вебінтерфейсу ні з погляду дизайну, ні з погляду реалізації клієнт-серверної взаємодії.

Vaadin підтримує всі найпоширеніші браузери як звичайних комп'ютерів, так і мобільних пристроїв та планшетів. Вся розробка ведеться на Java, але Java-код виконується лише на сервері, на клієнті ж виконується чистий JavaScript.

Структурно Vaadin складається з серверного API, клієнтського API, набору компонентів інтерфейсу з обох сторін, механізму тем для оформлення інтерфейсу і моделі даних, що дозволяє пов'язувати серверні компоненти безпосередньо з даними. Можна використовувати дві основні моделі розробки: на стороні сервера та на стороні клієнта (браузера).

Серверна модель розробки для Vaadin є основною і дозволяє створювати закінчені програми без розробки на стороні клієнта. При цьому використовується AJAX – механізм Vaadin Client-Side Engine, який формує інтерфейс користувача в браузері. Серверний підхід дозволяє практично забути про те, що технологія ведеться під інтернет, і розробляти інтерфейс майже як традиційну Java-програму з безпосереднім доступом до даних і сервісів на сервері. При цьому серверна частина Vaadin подбає і про формування інтерфейсу користувача в браузері, і про AJAX-взаємодію між браузером і сервером. Vaadin здійснює рендеринг інтерфейсу програми серверної сторони в браузері і реалізує всі деталі обміну клієнта і сервера.

Серверна частина програми Vaadin виконується як звичайний сервлет сервера програм Java. Це чиста Java в JAR-файлі, який може додаватися до будь-якого стандартного вебзастосунку і працює на будь-якому контейнері сервлетів або портлетів від Tomcat до Oracle WebLogic. Сервлет приймає HTTP-запити від клієнта і інтерпретує їх як події конкретної сесії. Події асоційовані з компонентами інтерфейсу користувача і доставляються до обробників (event listeners), визначених у застосунку. Якщо логіка інтерфейсу користувача вносить зміни в компоненти інтерфейсу користувача з боку

сервера, сервлет рендерить їх для відображення в веббраузері і формує відповідь. Двигун клієнтської частини, який виконується в браузері, отримує відповідь і на його основі робить зміни у завантаженій у браузері вебсторінці.

Клієнтська модель дозволяє розробляти віджети та програми на мові Java, які потім компілюються у JavaScript, що виконується в браузері, за допомогою компілятора Vaadin Compiler, заснованого на Google Web Toolkit (GWT). Можна використовувати безпосередньо JavaScript. Це надає повний доступ до структури DOM та максимальний контроль над браузером [28].

Аналізуючи наведену інформацію, було прийнято рішення використати для реалізації вебзастосунку наступні технології:

– Java EE;
– Apache Tomcat Server;
– PostgreSQL;
– Spring Boot;
– Spring Security;
– Spring Data;
– Vaadin 14+;
– Maven.

Вибір правильних технологій та фреймворків є дуже важливим при розробці програмного забезпечення. Тому технології були обрані таким чином, щоб унеможливити конфлікти несумісності, не перевантажити код не потрібними технологіями, зробити розробку швидкою (допоможе добре написана документація Spring та Vaadin). Отже, виходячи з цього потужність мови програмування Java відкриє доступ до необхідних можливостей розробки вебзастосунків, система автоматичного збирання проекту Maven сильно спростить розробку підключивши всі необхідні залежності та конфігурації. Vaadin дозволе швидко розробити клієнтську частину та полегшить взаємодію між сервером та клієнтською стороною застосунку. Spring Security надасть технології безпеки, Spring Data дозволить взаємодіяти з базою даних PostgreSQL, а Spring Boot об’єднає в собі всі модулі і

використовуючи контейнер для сервлетів Apache Tomcat Server запустить застосунок на локальному сервері.

2.2 Проектування архітектури застосунку

2.2.1 Що таке архітектура застосунку

Програмна архітектура будь-якої програми описує її основні компоненти, їх відношення і те, як вони взаємодіють один з одним. По суті, він служить кресленням. Забезпечує абстракцію для управління складністю системи та встановлення зв’язку і координації між компонентами.

Ось деякі ключові моменти:

– архітектура допомагає визначити рішення, що відповідає всім технічним і експлуатаційним вимогам, із загальною метою оптимізації продуктивності і безпеки;

– проектування архітектури передбачає перетин потреб організації та потреб команди розробників. Кожне рішення може мати значний вплив на якість, швидкодійність та багато іншого.

Отже, з урахуванням сказаного, перейдемо до того, чому архітектура програмного забезпечення важлива (рис. 2.3).

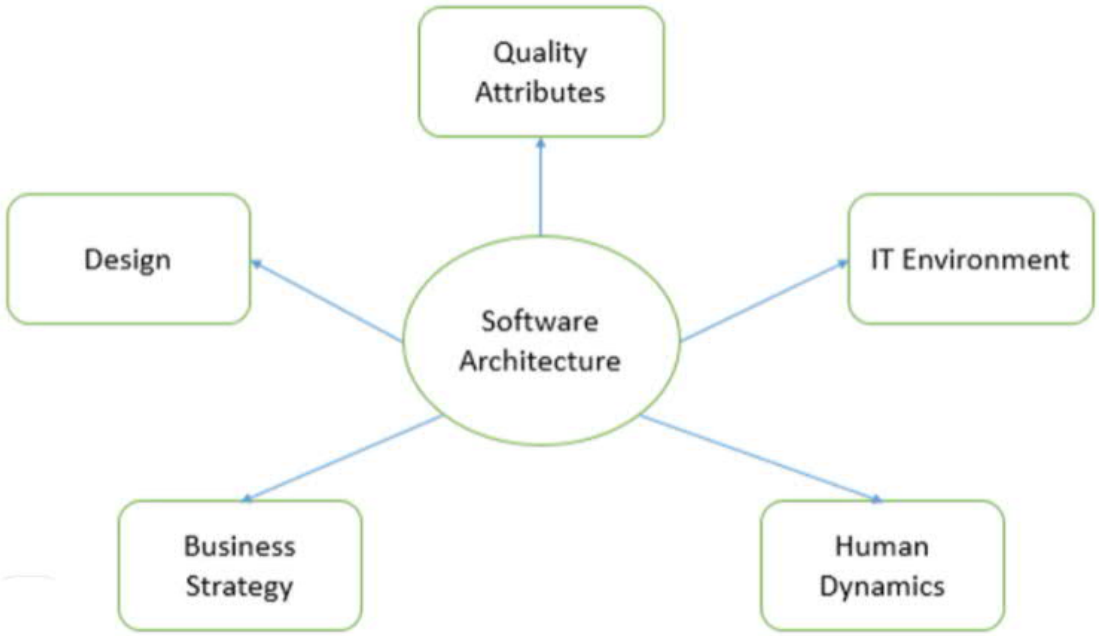

Рисунок 2.3 – Основні складові архітектури програмного забезпечення

Ключовим елементом успішного створення чого-небудь є правильна база. Будь то будівництво будівлі чи чогось іншого. Якщо не отримаємо правильну базу, доведеться почати все спочатку, іншого шляху немає.

Створення вебзастосунку нічим не відрізняється. Архітектура є його основою і повинна бути ретельно продумана, щоб уникнути будь-яких серйозних змін дизайну і рефакторинга коду в більш пізній момент часу.

Багато інженерів скажуть, що не кожен програміст захоче заглиблюватися в перепроектування. Вона пожирає час, як чорна діра. Може здвинути дату здачі проекту далі за календарем на місяці, а то й більше.

Це також залежить від того, на якому етапі процесу розробки було зайдено в глухий кут через поспішні рішення, прийняті на початкових етапах проектування. Тому, перш ніж навіть доторкнутися до коду потрібно зробити базову архітектуру правильною.

Хоча розробка програмного забезпечення – це ітеративний і еволюційний процес, не завжди досягається досконаліть з першого разу. Однак, це не може бути виправданням. Часто виникає плутанина між

дизайном програмного забезпечення та архітектурою, тому це питання буде розглянуто.

Архітектура програмного забезпечення використовується для визначення скелета і високорівневих компонентів системи, а також того, як всі вони будуть працювати разом. Наприклад, чи потрібна безсерверна архітектура, яка розділяє застосунок на два компоненти: BaaS (backend-as-a-service) і FaaS (functions-as-a-service)?

Або потрібно щось на зразок архітектури мікросервісу, в якій різні функції чи завдання розділені на окремі відповідні модулі або кодові бази?

Вибір архітектури визначатиме, як застосунок буде справлятися з продуктивністю, відмовостійкістю, масштабованістю і надійністю. Розробка програмного забезпечення відповідає за розробку рівня коду і за те, що робить кожен модуль, область дії класів, призначення функцій та інше. При стратегічному використанні вони можуть зробити програміста значно ефективнішим, дозволивши йому уникнути винаходу колеса, замість цього використовуючи методи, вже вдосконалені іншими.

2.2.2 Шаблони архітектури програмного забезпечення

Клієнт-серверна архітектура працює за моделлю «запит-відповідь» [29]. Клієнт відправляє запит на сервер для отримання інформації, і сервер відповідає на нього (рис. 2.4). Кожен вебсайт, який переглядається, будь то блог на Wordpress або вебзастосунок, такий як Facebook, Twitter або банківський застосунок, побудований на архітектурі клієнт-сервер.

Ще однією наявною архітектурою є мікросервіси (рис. 2.5). В архітектурі мікросервісів різні функції та завдання поділяються на окремі відповідні модулі (кодові бази), які працюють спільно один з одним, утворюючи велику службу в цілому. Ця архітектура полегшує і спрощує обслуговування застосунків, розробку функцій, тестування і розгортання в порівнянні з монолітною архітектурою.

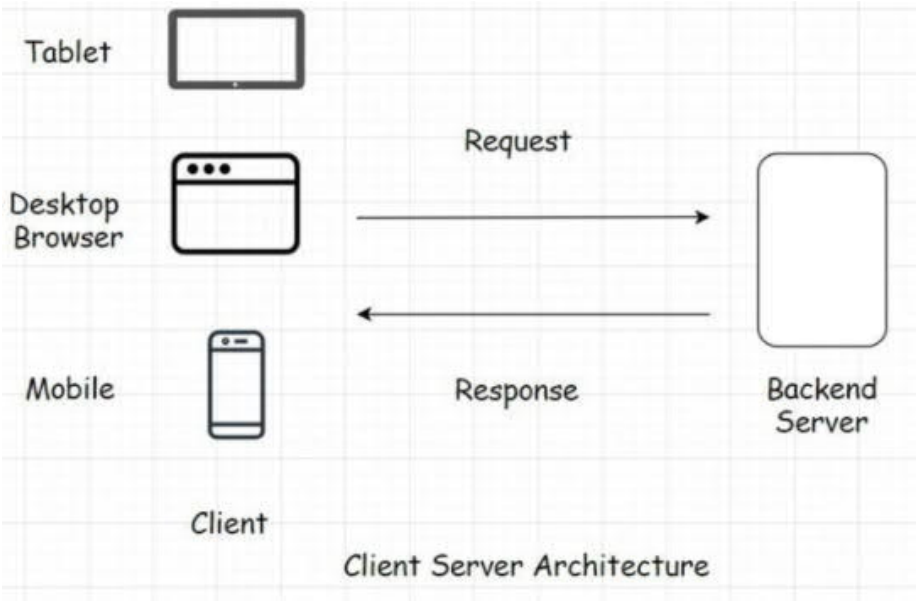

Рисунок 2.4 – Схема клієнт-серверної архітектури

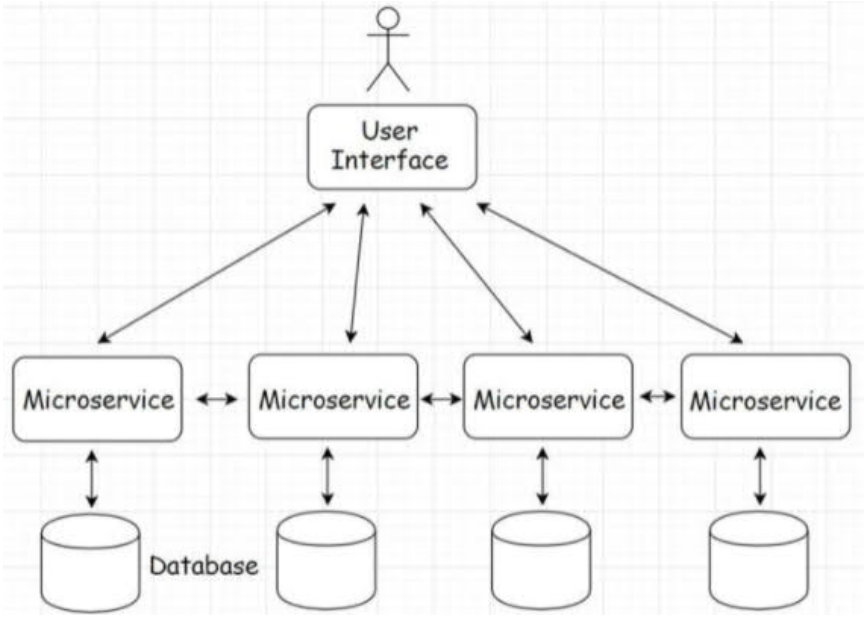

Рисунок 2.5 – Схема мікросервісної архітектури

Vaadin також має свій шаблон архітектури (рис. 2.6), якого потрібно маємо дотримуватися при роботі з цим фреймворком. При роботі з конфігурацією Vaadin треба розуміти такі моменти:

– клієнтське ядро Vaadin використовується для візуалізації виводу або дій користувача за допомогою основного методу HTTP та браузера. Це покоління представницького блоку повністю автоматизоване. Доведеться кодувати тільки бекенд, всі розмітки генеруватимуться автоматично;

– серверна частина обробляє бізнес-частину, яка отримує запит на основі подій та готує відповідь для клієнта. Зв’язок між двома рівнями відбувається за протоколами HTTP;

– термінальний адаптер приймає запит і обробляє його, використовуючи компонент інтерфейсу користувача на стороні сервера, який є класом JAVA на основі сервера, для генерації відповіді, який буде відображатися з використанням іншого компонента GWT (Google Web Tool Kit). Це також називається Vaadin Servlet API, який розширює властивості сервлета, отримує запити від різних клієнтів та визначає відповідь користувача;

– Vaadin використовує GWT і, отже, забезпечує швидший висновок та покращену масштабованість, ніж звичайна програма на основі Java Script [30].

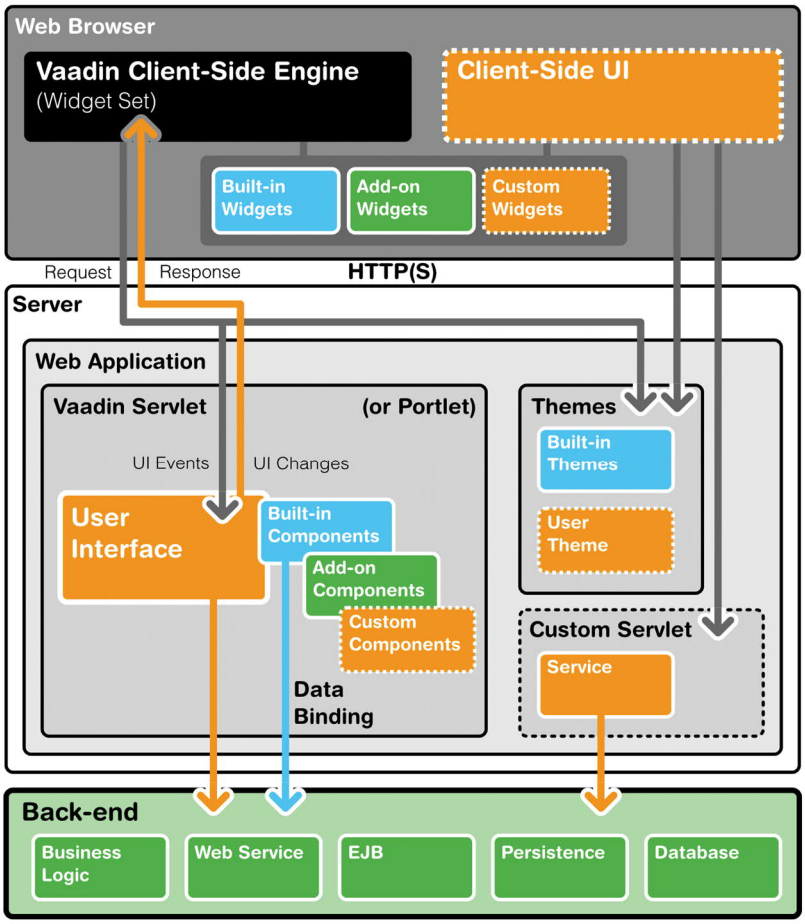

Рисунок 2.6 – Архітектура Vaadin

2.3 Проектування бази даних

Проектування бази даних – це процес створення проекту, який буде підтримувати формулювання місії та цілей бізнесу для необхідної системи баз даних. Дотримуються двох основних підходів до проектування бази даних. Це:

– знизу вгору;

– зверху донизу.

Підхід знизу вгору починається на фундаментальному рівні атрибутів (тобто властивостей сутностей та відношень), які через аналіз асоціацій між атрибутами об’єднуються у відношення.

Більш доцільною стратегією для проектування складних баз даних є використання підходу зверху донизу, який починається з розробки моделей даних, які містять кілька сутностей і відношень високого рівня, а потім застосовувати послідовні уточнення зверху вниз для ідентифікації сутностей нижчого рівня, відносини та пов’язані з ними атрибути.

Наступним етапом буде проектування бази даних. Визначимо структуру бази, тобто поля таблиць та логічні зв’язки і перенесемо це все в застосунок Intelij Idea DB, який дозволить автоматично створити схему структури бази даних (рис. 2.7).

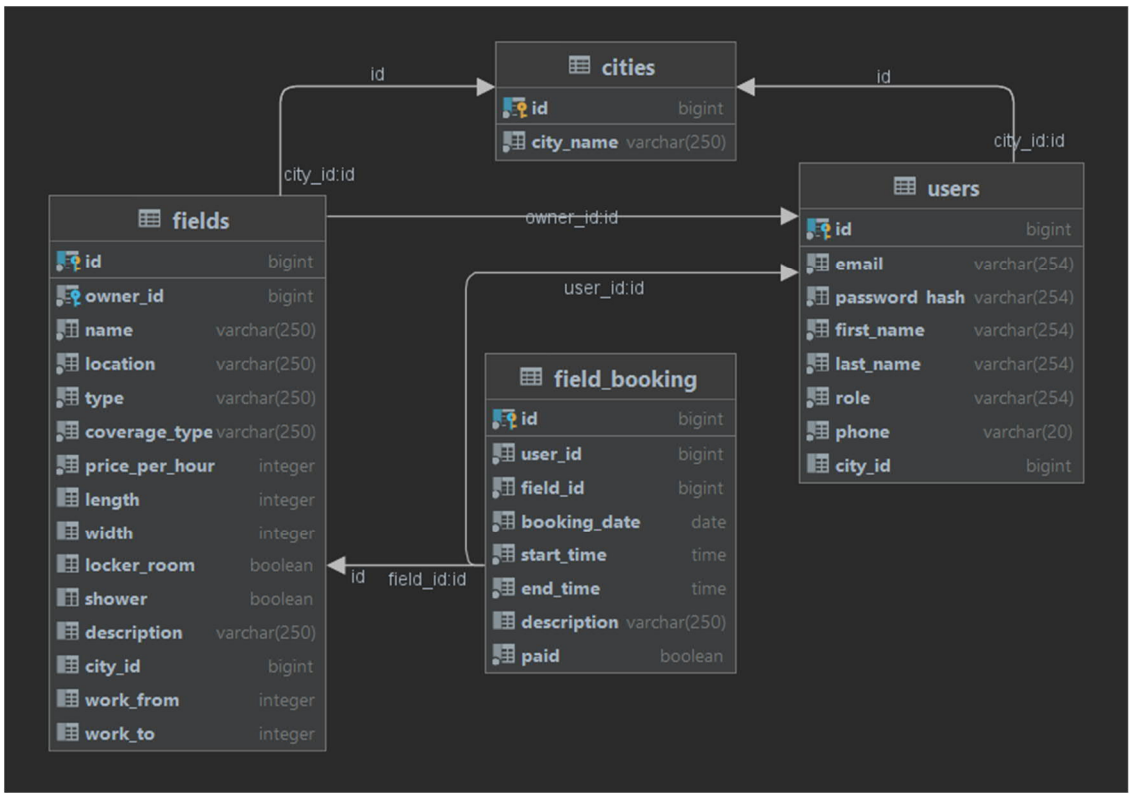

Рисунок 2.7 – Структура БД

3 РОЗРОБКА CRUD ВЕБЗАСТОСУНКУ

3.1 Розробка бази даних та інструментів роботи з нею

3.1.1 Створення бази даних PostgreSQL

Визначимо основні сутності ІС:

Сутність «User» містить всю інформацію про користувачів застосунку та має наступні атрибути:

– id – ідентифікатор користувача, має тип integer, є первинним ключем, не може бути NULL;

– first_name – ім’я, тип text;

– last_name – прізвище, тип text;

– emai – електронна пошта користувача, обов’язкове поле, не може бути NULL, тип text;

– role – роль користувача у рамках застосунку, тип text;

– phone – телефон користувача, тип text;

– city_id – ідентифікатор міста, у якому перебуває користувач, має тип integer, є зовнішнім ключем, не може бути NULL;

– password_hash – закодований пароль користувача, тип text, такий спосіб є найбільш безпечним для зберігання паролів.

Сутність «Field» містить всю інформацію про футбольні поля застосунку та має наступні атрибути:

– id – ідентифікатор футбольного поля, має тип integer, є первинним ключем, не може бути NULL;

– owner_id – ідентифікатор власника поля, користувача, має тип integer, є зовнішнім ключем, не може бути NULL;

– name – назва поля, тип text;

– location – адреса поля, тип text;

– type – тип поля (для якого футболу використовується), тип text;

– coverage_type – тип матеріалу покриття поля, тип text;

– price_per_hour – ціна за 1 годину оренди поля, тип integer;

– length – довжина поля, тип integer;

– width – ширина поля, тип integer;

– locker_room – наявність роздягальні на території поля, тип boolean;

– shower – наявність душу на території поля, тип boolean;

– description – додатковий опис поля, тип text;

– city_id – ідентифікатор міста, у якому розташоване поле, має тип integer, є зовнішнім ключем, не може бути NULL;

– work_from – з якого часу працює поле, тип integer;

– work_to – до якого часу працює поле, тип integer.

Сутність «Field_Booking» містить всю інформацію про бронювання полів та має наступні атрибути:

– id – ідентифікатор бронювання, має тип integer, є первинним ключем, не може бути NULL;

– user_id – ідентифікатор орендатора поля, користувача, має тип integer, є зовнішнім ключем, не може бути NULL;

– field_id – ідентифікатор поля, має тип integer, є зовнішнім ключем, не може бути NULL;

– booking_date – дата, на яку заплановане бронювання, тип date;

– start_time – час початку бронювання, тип time;

– end_time – час завершення бронювання, тип time;

– description – додаткові записи, які можуть залишати як власник, так і орендатор, тип text;

– paid – поле маркеру оплати за оренду, тип boolean.

Сутність «Cities» містить всю інформацію про користувачів застосунку та має наступні атрибути:

– id – ідентифікатор міста, має тип integer, є первинним ключем, не може бути NULL;

– city_name – назва міста, тип text.

Виходячи з того, що для роботи з базою даних використовується Hibernate, то перед написанням класів сутностей потрібно створити базу даних за допомогою зручною графічного інтерфейсу PostgreSQL. Використовуючи його, не доведеться писати навіть жодного запиту. Переглянемо структуру бази за допомогою інтерфейсу PostgreSQL (рис. 3.1 – 3.3).

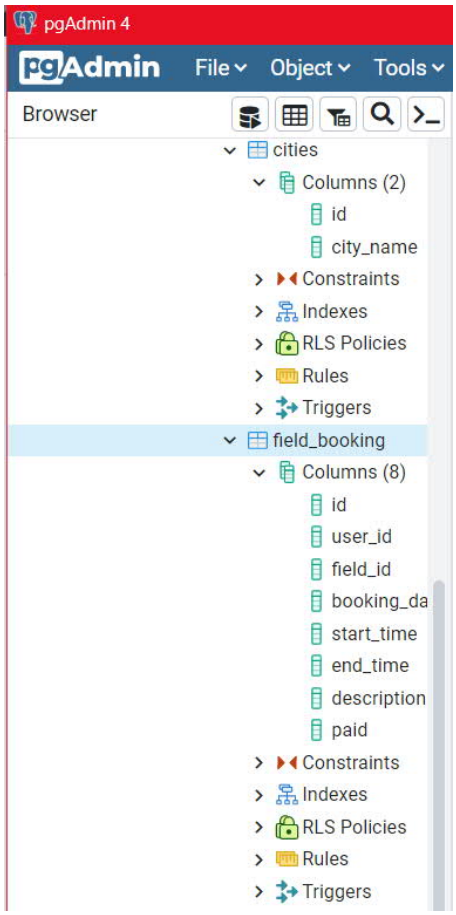

Рисунок 3.1 – Створені таблиці Cities та Field_bookings в PostgreSQL

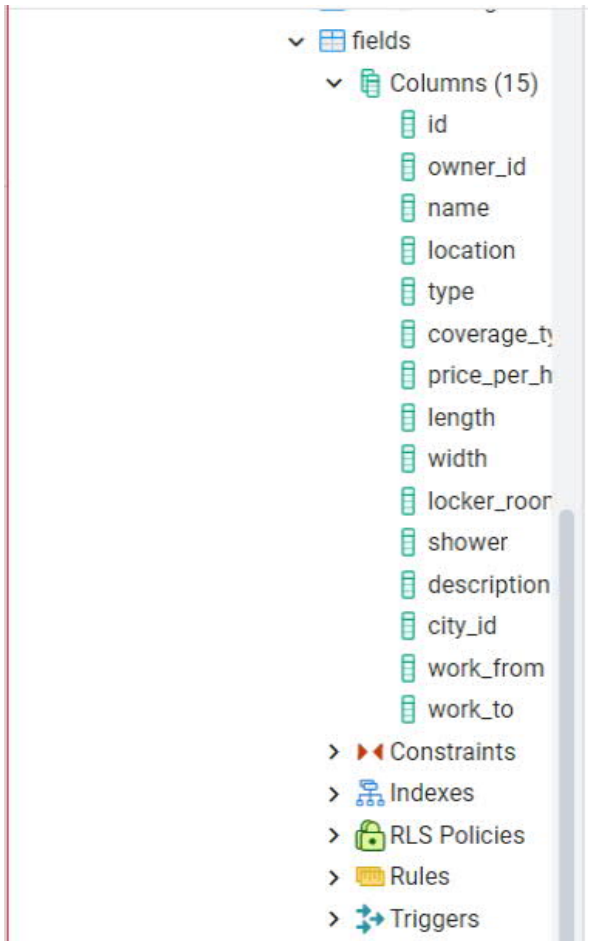

Рисунок 3.2 – Створена таблиця Fields в PostgreSQL

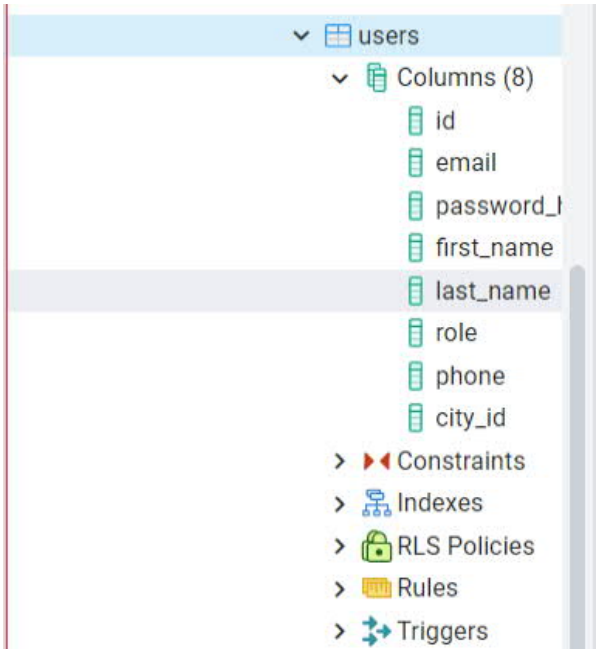

Рисунок 3.3 – Створена таблиця Users в PostgreSQL

3.1.2 Створення DAO слою застосунку

Рівень DAO зазвичай складається з великої кількості шаблонного коду, який можна спростити. Переваг такого спрощення багато: зменшення кількості артефактів, які необхідно визначити та підтримувати, узгодженість шаблонів доступу до даних та узгодженість конфігурації.

Spring Data робить ще один крок уперед у цьому спрощенні та дозволяє повністю видалити реалізації DAO. Інтерфейс DAO є єдиним артефактом, який потрібно явно визначити.

Щоб почати використовувати модель програмування Spring Data з JPA, інтерфейс DAO повинен розширити інтерфейс репозиторію, специфічний для JPA, JpaRepository. Це дозволить Spring Data знайти цей інтерфейс та автоматично створити для нього реалізацію.

Розширюючи інтерфейс, отримуємо найактуальніші методи CRUD для стандартного доступу до даних, доступних у стандартному DAO.

Все, що потрібно, успадкуватися від інтерфейсу JpaRepository та передати клас-сутність. Подивимось на приклад (рис. 3.4).

```
public interface FieldBookingRepository extends JpaRepository<FieldBooking, Long> {

    List<FieldBooking> getAllByFieldAndBookingDateOrderByStartTime(Field field, LocalDate bookingDate);

    List<FieldBooking> findAllByUser(User user);
}
```

Рисунок 3.4 – Інтерфейс FieldBookingRepository

Обов’язковим також є реалізація класів-сервісів для всіх сутностей. Хорошим тоном є створення загального інтерфейсу сервісу. Подивимось приклад реалізації такого інтерфейсу у застосунку (рис. 3.5).

```java
public interface CrudService<T> {

    JpaRepository<T, Long> getRepository();

    List<T> findAll(String filterValue);

    T createNewOne();

    @PreAuthorize("hasAuthority('" + Role.ADMIN + "')")
    void deleteAll(Iterable<? extends T> collection);

    default T save(T entity) { return getRepository().save(entity); }

    @PreAuthorize("hasAuthority('" + Role.ADMIN + "')")
    default void delete(T entity) {
        if (entity == null) {
            throw new NotFoundException();
        }
        getRepository().delete(entity);
    }

    @PreAuthorize("hasAuthority('" + Role.ADMIN + "')")
    default void deleteById(Long id) { delete(getById(id)); }

    default List<T> findAll() { return getRepository().findAll(); }

    default List<T> findAll(Sort sort) { return getRepository().findAll(sort); }

    default long count() { return getRepository().count(); }
```

Рисунок 3.5 – Інтерфейс CrudService

3.2 Реалізація безпеки застосунку за допомогою Spring Security

Безпека Spring забезпечує автентифікацію та авторизацію програми за допомогою простих фільтрів сервлетів. Вебпрограми піддаються загрозам безпеці та атакам, оскільки вони доступні будь-якому користувачеві Інтернету. Можуть існувати деякі кінцеві точки REST з обмеженим доступом для певних користувачів, наприклад оновлення записів або операції, пов’язані з адмініструванням.

Існує можливість використовувати Spring Security для захисту URL-адрес. Spring Security – це інфраструктура безпеки, яка захищає корпоративні програми на основі Jakarta EE, надаючи потужні функції безпеки, такі як автентифікація та авторизація. Це стандарт де-факто для захисту програм на основі Spring.

Безпека Spring працює над такими трьома основними концепціями:

– аутентифікація;

– авторизація;

– сховище паролів;

– фільтри сервлетів.

Для конфігурування безпеки застосунку потрібно створити клас SecurityConfiguration та метод «configure» у цьому класі (рис. 3.6). У цьому методі виконується основна конфігурація безпеки застосунку.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    // Not using Spring CSRF here to be able to use plain HTML for the login page
    http.csrf().disable() HttpSecurity

            // Register our CustomRequestCache, that saves unauthorized access attempts, so
            // the user is redirected after login.
            .requestCache().requestCache(new CustomRequestCache()) RequestCacheConfigurer<HttpSecurity>

            // Restrict access to our application.
            .and().authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry

            // Allow all flow internal requests.
            .requestMatchers(SecurityUtils::isFrameworkInternalRequest).permitAll()

            // Allow all requests by logged in users.
            .anyRequest().hasAnyAuthority(Role.getAllRoles())

            // Configure the login page.
            .and().formLogin() FormLoginConfigurer<HttpSecurity>
            .loginPage(LOGIN_URL)
            .permitAll()
            .loginProcessingUrl(LOGIN_PROCESSING_URL)
            .defaultSuccessUrl(LOGIN_SUCCESS_URL, alwaysUse: true)
            .failureUrl(LOGIN_FAILURE_URL)
            .successHandler(authenticationSuccessHandler)

            // Configure logout
            .and().logout().logoutSuccessUrl("/");
}
```

Рисунок 3.6 – Метод «configure» класу SecurityConfigurating

3.3 Реалізація користувацького інтерфейсу

3.3.1 Сторінка авторизації

Перша сторінка яку необхідно описати – це login (рис. 3.7). Як вже було сказано, Spring Security надає вже готове рішення зі сторінкою для введення логіна і пароля «з-під капоту». Тобто, достатньо лише підключити до pom.xml файлу необхідну залежність. Але готового рішення не завжди буває достатньо. Тому було прийнято рішення. Створити Enum з ролями для прав доступу. Створити окремий клас, який наслідується від WebSecurityConfigurerAdapter та який перевизначить такі методи, як: configure, userDetailsService. В свою чергу – це дозволить обмежити доступ до сторінки з додаванням нового ліда до бази даних, заборонить видалення даних звичайним користувачам без відповідних прав, а також налаштувати пароль і логін для кожної ролі.

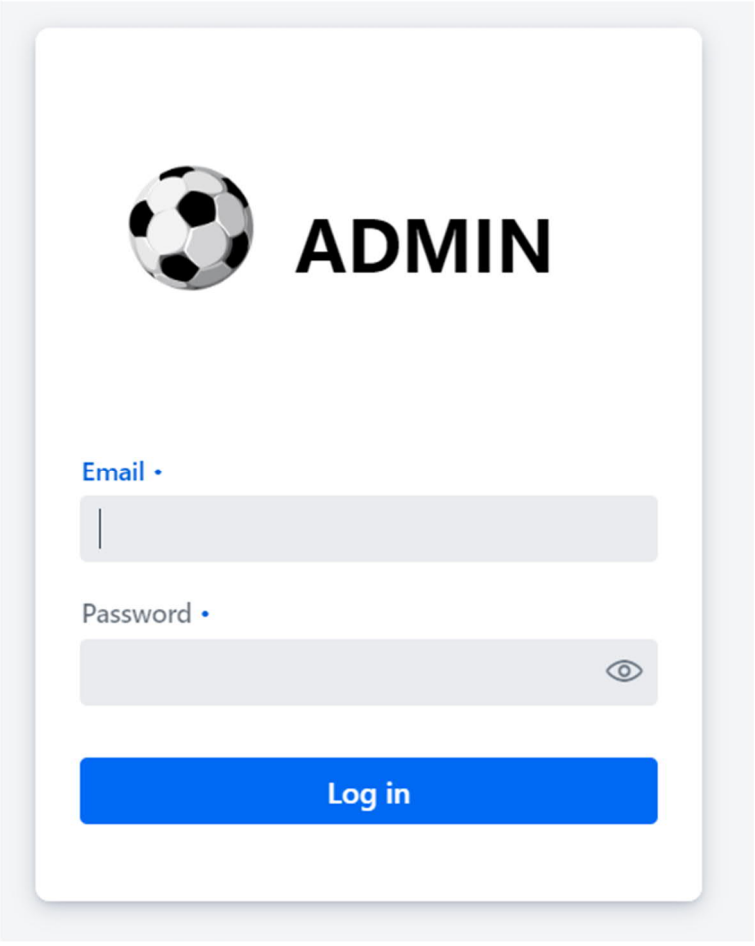

Рисунок 3.7 – Сторінка Login

3.3.2 Головне меню

Головне меню являє собою представлення в зручному вигляді всіх даних на поля та бронювання. В залежності від ролі користувача, меню має два різні вигляди, а саме :

– меню для простого користувача (рис. 3.8);
– меню для адміністратора (рис. 3.9).

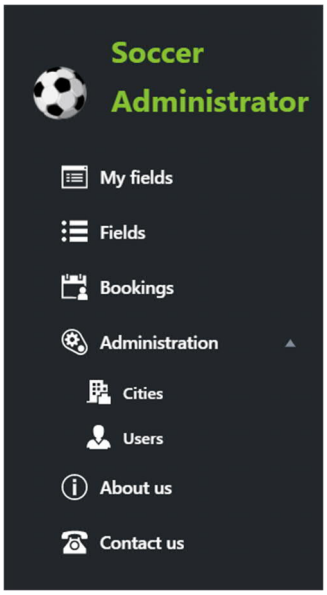

Рисунок 3.8 – Вигляд меню для адміністратора

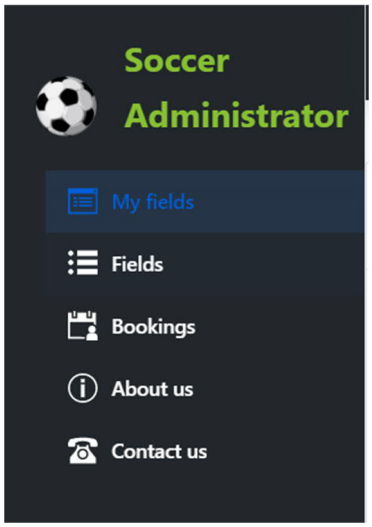

Рисунок 3.9 – Вигляд меню для користувача

Меню для адміністратора та для користувача мають декілька однакових пунктів, а саме:

– «MyField» – відкриває сторінку с полями авторизованого користувача;

– «Fields» – відкриває сторінку з усіма полями у базі даних для їх бронювання;

– «Aboout us» – відкриває сторінку опису програмного засобу;

– «Contact us» – відкриває сторінку зв’язку с технічною підтримкою застосунку;

– «Bookings» – відкриває сторінку управління всіма наявними у застосунку бронюваннями для ролі адміністратора, та бронюваннями поточного користувача для ролі користувача.

Авторизація с роллю адміністратора додає пункт «Administration» до меню, у якому є такі пункти:

– «Users» – відкриває сторінку управління всіма наявними у застосунку користувачами;

– «Cities» – відкриває сторінку управління всіма наявними у застосунку містами.

3.3.3 Основні сторінки

Спершу роздивимось сторінки «Users», «Cities» та «Bookings» (рис. 3.10 – 3.12). Ці сторінки, окрім «Bookings», доступні лише адміністратору та служать для управлінням відповідними записами у базі даних.

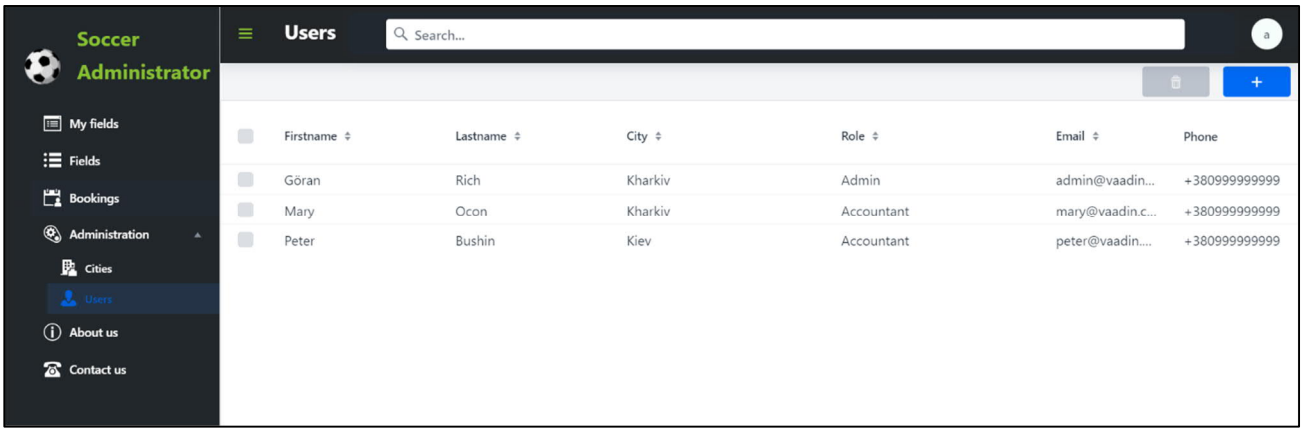

Рисунок 3.10 – Сторінка «Users»

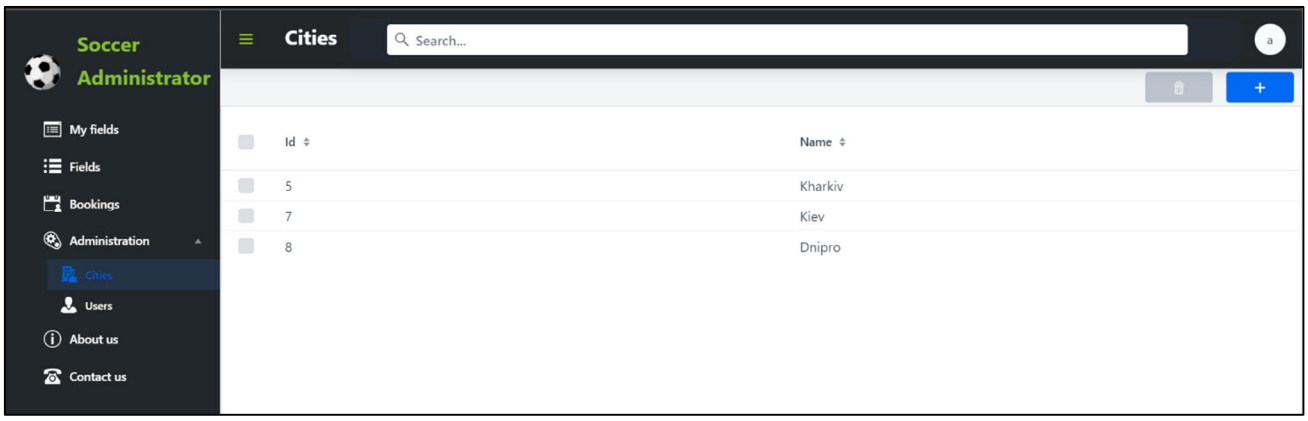

Рисунок 3.11 – Сторінка «Cities»

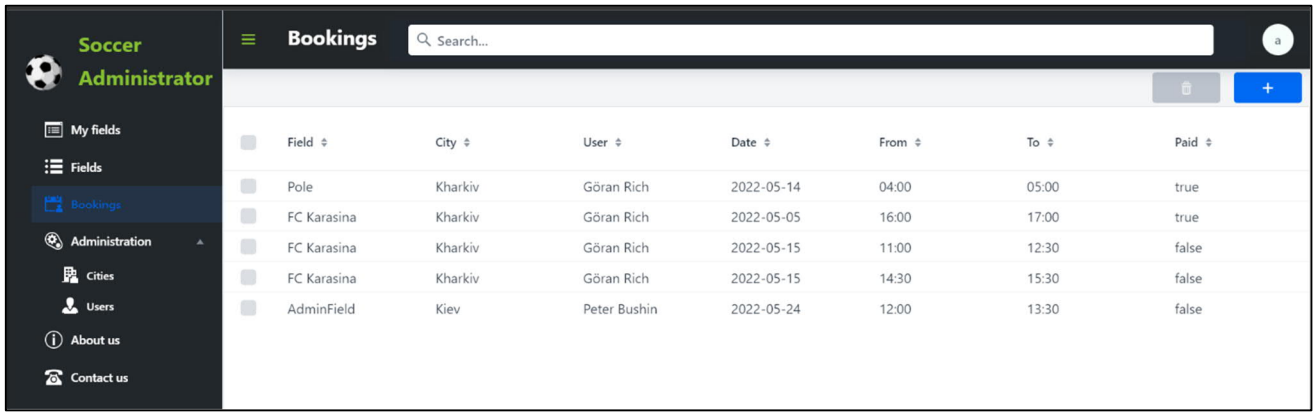

Рисунок 3.12 – Сторінка «Bookings»

Всі сторінки управліннями записами в базі даних мають однакову поведінку. Детальніше роздивимось функціонал на прикладі сторінки «Bookings». При натисканні на кнопку «+» відкривається форма з пустими полями для створення нового запису відповідного типу (рис. 3.13).

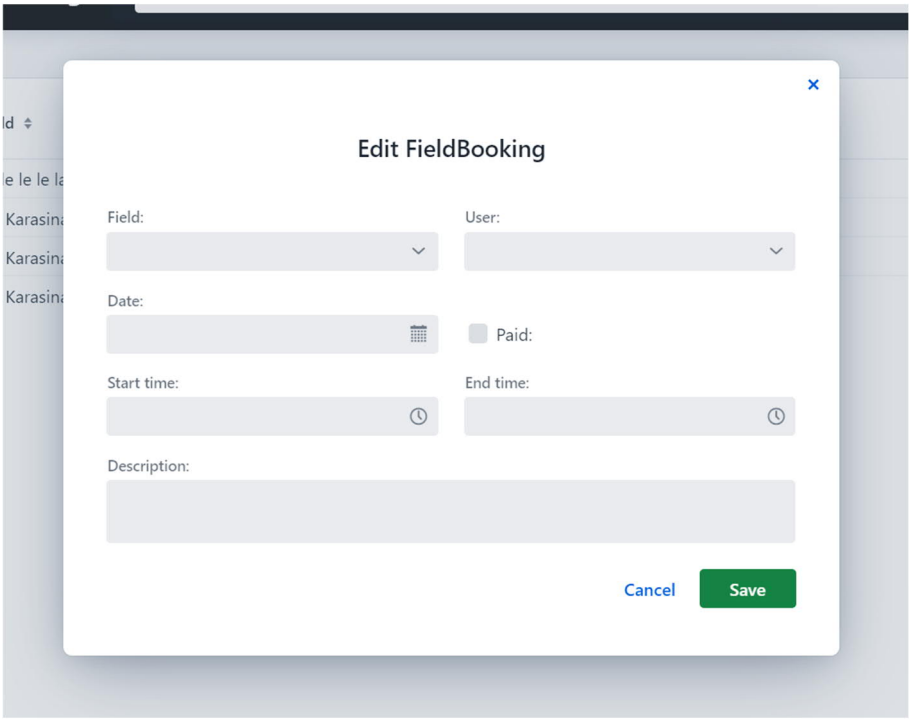

Рисунок 3.13 – Форма створення нового запису на сторінці «Bookings»

Усі комбобокси автоматично заповнюються записами з бази даних. У цій формі наявні два комбобокси, «Field» та «User» (рис. 3.14, 3.15).

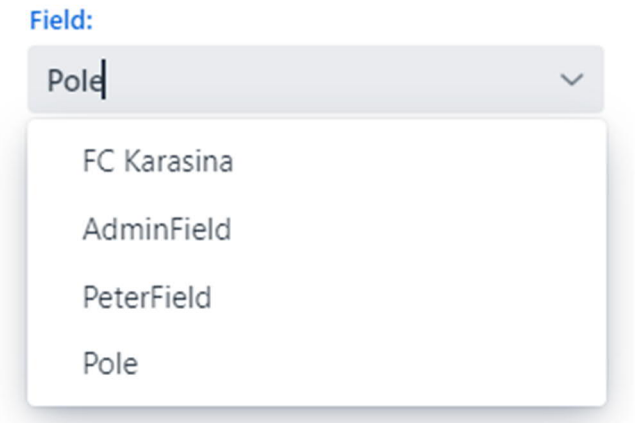

Рисунок 3.14 – Комбобокс «Field»

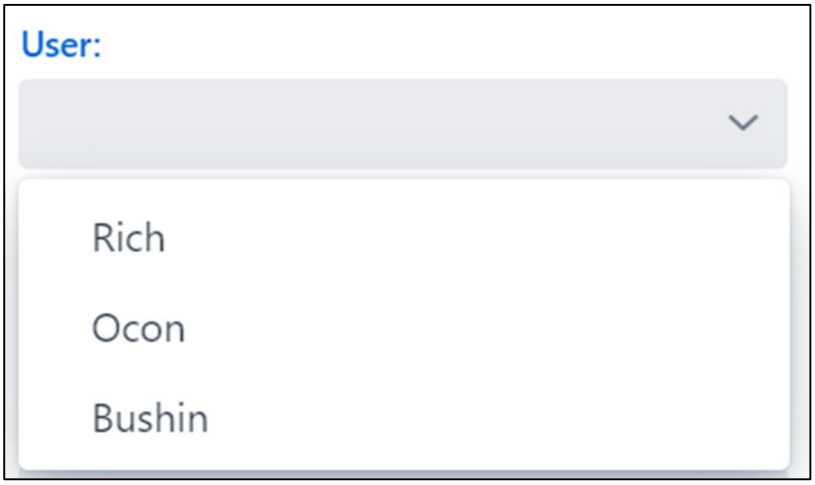

Рисунок 3.15 – Комбобокс «User»

Класи-компоненти DatePicker та TimePicker фреймворку Vaadin надають готові рішення для введення користувачем дати та часу (рис. 3.16, 3.17)

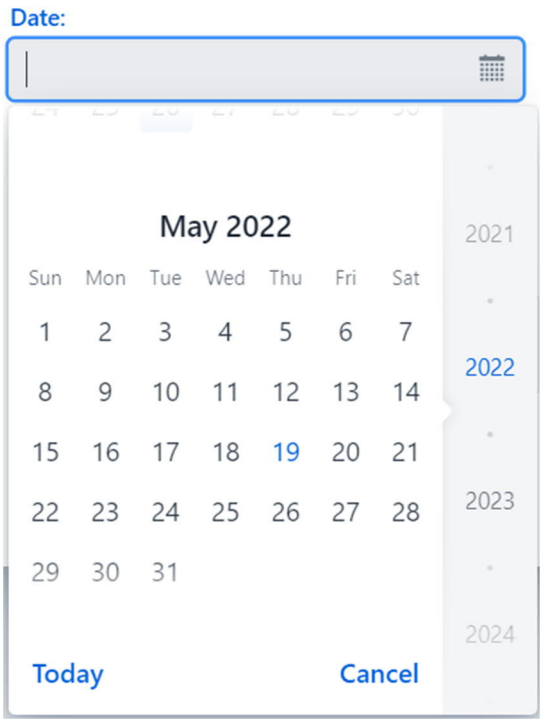

Рисунок 3.16 – Введення дати у поле «Date»

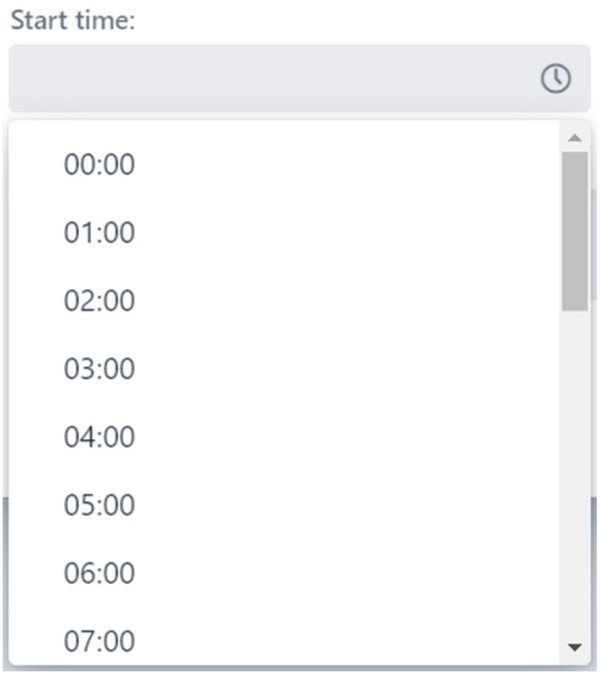

Рисунок 3.17 – Введення часу у поле «Start time»

Далі розглянемо редагування запису. Форму для редагування можна відкрити натиснувши на відповідний запис у таблиці (рис. 3.18). Існує можливість внести усі потрібні зміни та натиснути кнопку «Save» і всі зміни збережуться у базі.

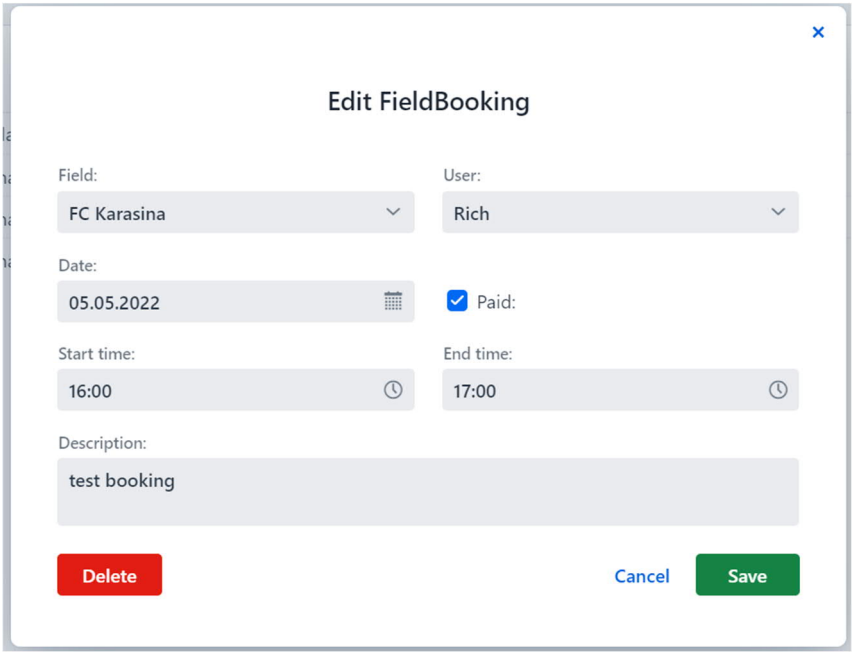

Рисунок 3.18 – Форма редагування запису «FieldBooking»

Тепер розглянемо дві основні сторінки для звичайного користувача. Була поставлена задача створити дві сторінки :

– для адміністрування своїх власних полів;

– для перегляду всіх наявних полів та їх бронювання.

Проаналізувавши ці вимоги, було розроблено сторінки «MyField» та «Fields» (рис. 3.19, 3.20).

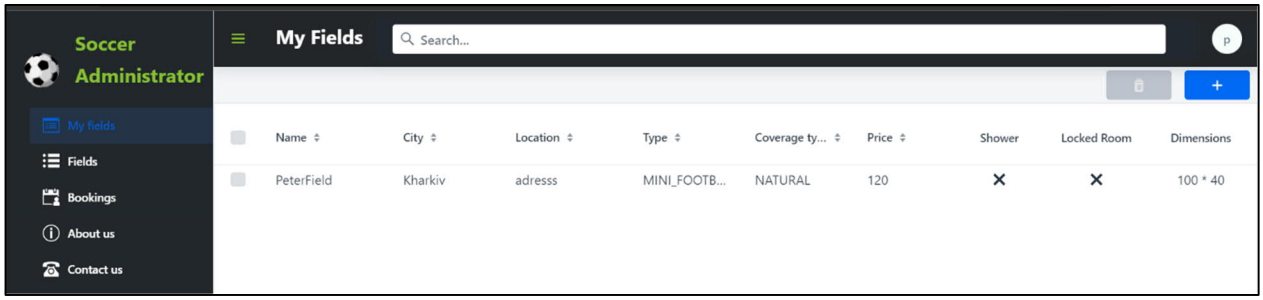

Рисунок 3.19 – Сторінка «MyField»

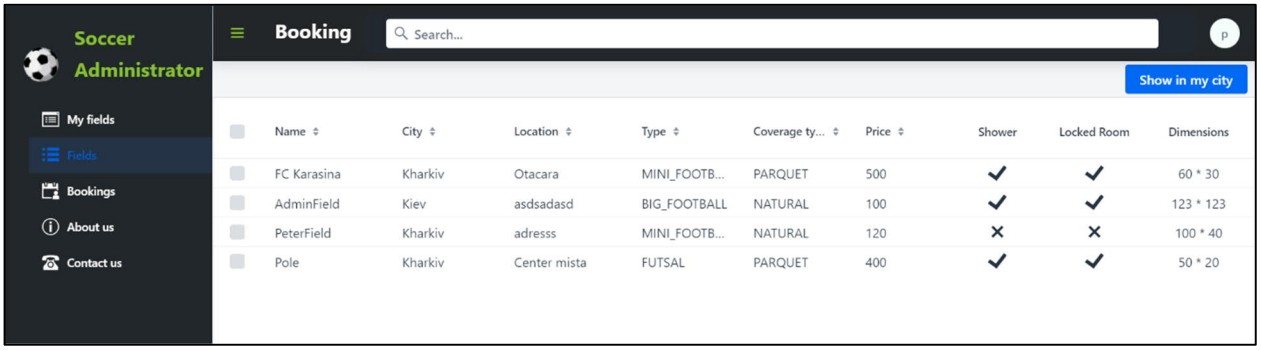

Рисунок 3.19 – Сторінка «Fields»

На сторінці «MyField» інтерфейс такий же як і на сторінках адміністратора. Користувач може створювати нові записи та редагувати їх (рис. 3.20).

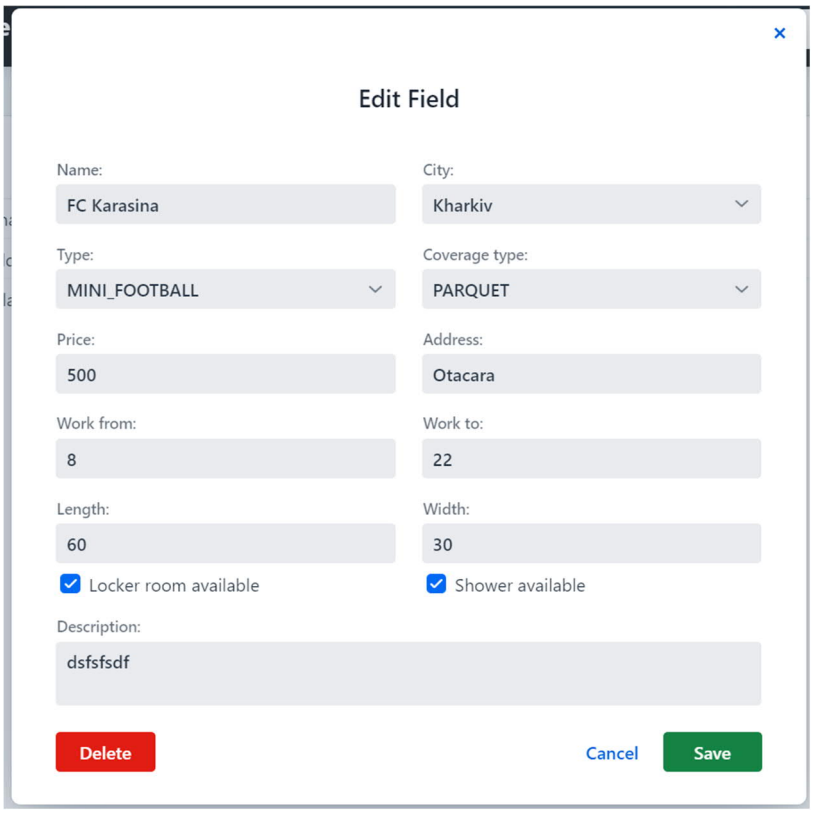

Рисунок 3.20 – Форма редагування запису «Field»

Сторінка «Fields» це, напевно, найважливіша та основна сторінка застосунку. Ця сторінка відрізняється від інших своїм функціоналом та можливостями. Цю сторінку користувач відкриває, коли хоче знайти підходяще для нього поле та забронювати його на певний час. Відображення записів зроблене так само, як і на інших сторінках, але верхнє меню з кнопками має інший вигляд. На не потрібні кнопки «Видалити» чи «Створити», бо вони не відповідають вимогам цієї сторінки. Тому було прийняте рішення додати до функціонального меню кнопку «Показати у моєму місті» (рис. 3.21). На перший погляд не дуже важлива кнопка, але вона дає можливість витрачати набагато менше часу на пошук потрібного поля. При її натисканні поля у списку фільтруються за містом, яке вказано у користувача в профілі та текст у кнопці змінюється на «Показати всі» (рис. 3.22).

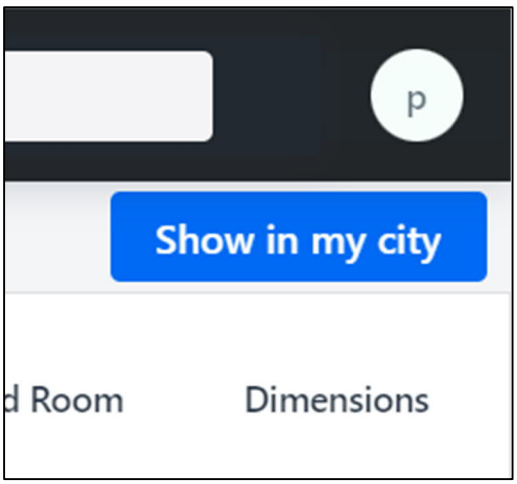

Рисунок 3.21 – Кнопка «Показати у моєму місті»

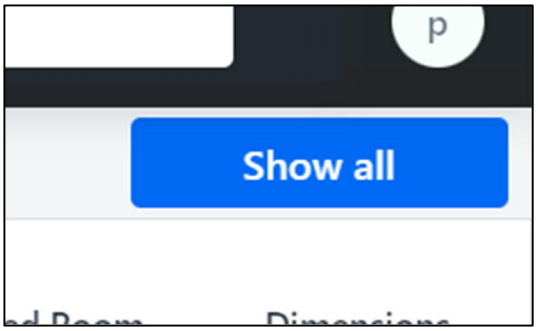

Рисунок 3.22 – Кнопка «Показати у моєму місті» в активованому стані

Наступною дією користувача, після переходу на сторінку бронювання, є натискання на відповідний запис с полем. Після цього відкривається форма бронювання. У цій формі є дані про поле. Спочатку відображається лише назва поля та прайс за одну годину, це згорнутий стан деталей про поле (рис. 3.23). Для того що подивитися всю інформацію потрібно розгорнути розділ «Більше інформації» (рис. 3.24).

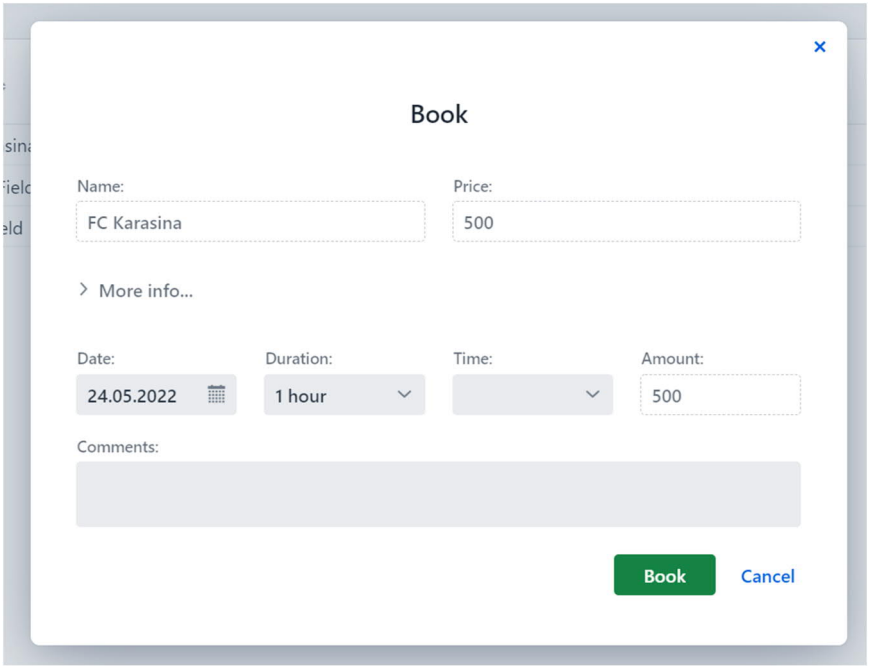

Рисунок 3.23 – Згорнутий стан деталей

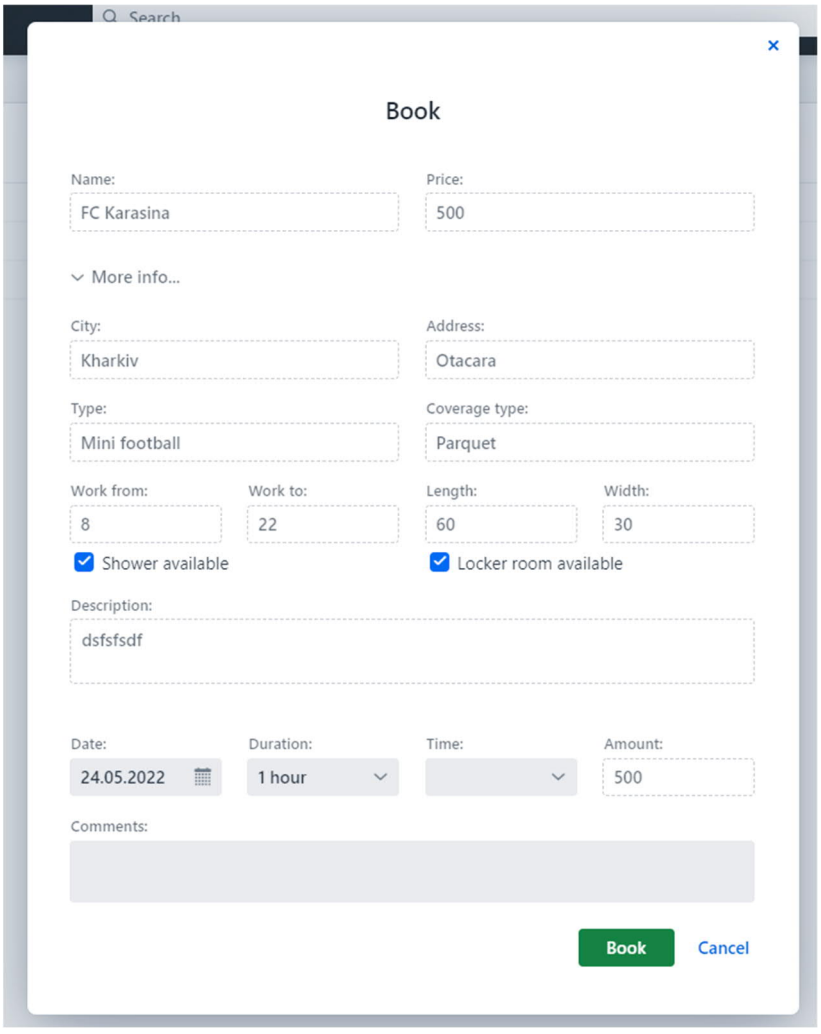

Рисунок 3.24 – Розгорнутий стан деталей

Наступний крок для користувача – вибір дати та тривалість на яку він хоче забронювати поле. Тривалості бронювання визначені у програмі, було обрано два найпопулярніші види тривалості:

– одна година;

– півтори години.

Їх вибір виконується у комбобоксі (рис. 3.25).

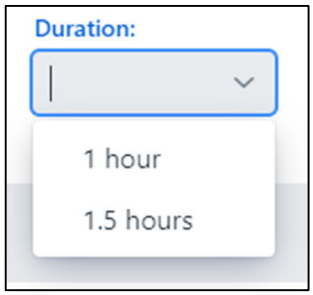

Рисунок 3.25 – Список тривалостей бронювання

Після вибору дати та тривалості спрацьовує основний алгоритм програми, який вираховує вільні проміжки часу за цими двома параметрами, та вже існуючими бронюваннями у цей час, і заповнює список можливих проміжків часу бронювання (рис. 3.26, 3.27).

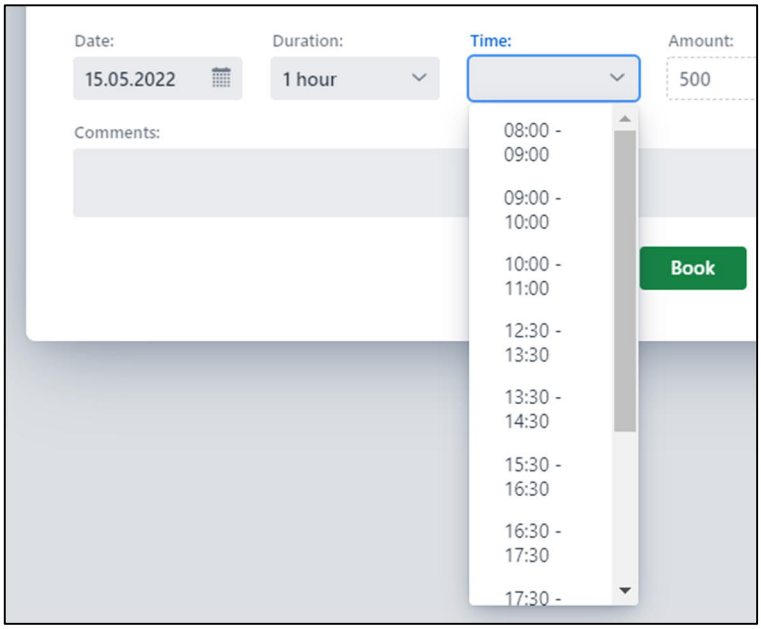

Рисунок 3.26 – Сформований список для тривалості «одна година»

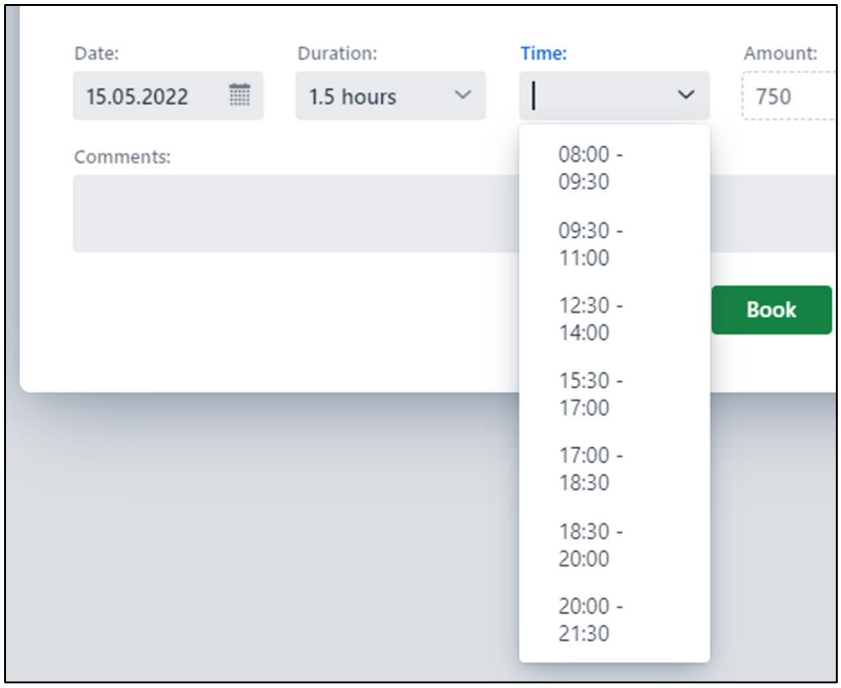

Рисунок 3.27 – Сформований список для тривалості «півтори години»

Як бачимо у відсутні деякі проміжки часу, бо вже існують бронювання на ті періоди (рис. 3.28).

☐	FC Karasina	Kharkiv	Göran Rich	2022-05-05	16:00	17:00	true
☐	FC Karasina	Kharkiv	Göran Rich	2022-05-15	11:00	12:30	false
☐	FC Karasina	Kharkiv	Göran Rich	2022-05-15	14:30	15:30	false

Рисунок 3.28 – Наявні бронювання поля «FC Karasina» на п’ятнадцяте квітня

ВИСНОВКИ

У ході кваліфікаційної роботи було розроблено CRUD вебзастосунок для автоматизації процесу адміністрування футбольних полів. Говорячи іншими словами, було створено програму букінгу для полів.

Хоча, такі платформи – це перш за все прикладне програмне забезпечення для готельного бізнесу, призначене для автоматизації розселення та огляду кімнат, зокрема для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про номера та бронювання, але реалізованого функціоналу достатньо для розв’язання основних задач в процесі адміністрування футбольних полів.

Будь-який програмний продукт повинен підтримуватись, оновлюватись, доопрацьовуватися. Тому, всі інші функції, які повинен мати повноцінний застосунок для адміністрування, повинні бути розроблені та внесені окремим модулем в майбутньому. Це можливо завдяки продуманим архітектурним рішенням зробленим на етапі проектування та вибору технологій. Адже, саме Java в поєднанні з системою автоматичного збирання проекту Maven до якої в вигляді залежностей підключені технології Spring Framework, дозволяє досягти модульності при розробці програмного забезпечення.

Розроблений програмний вебзастосунок автоматизує процес збору та обробки інформації. Основними функціями є додавання, редагування, видалення та перегляд інформації на полів та їх бронювань з БД. А отже, можна з впевненістю сказати, що поставлену задачу реалізовано. В процесі розробки були розглянуті та проаналізовані різноманітні архітектурні та проектні рішення. В процесі розробки були покращені навички програмування. Також було освоєно досить не просту технологію Spring Framework, яку вдалося, реалізувати на практиці. Знову ж таки, говорячи про

цей фреймворк, не можливо не згадати чималу кількість технологій, які були не тільки розглянуті, а й використані та поєднанні між собою для реалізації бізнес-логіки застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Цифрова економіка. URL: https://yur-gazeta.com/publications/legal-business-in-ukraine/cifrova-ekonomika-yaka-yuridichna-pidtrimka-yiy-potribna.html (дата звернення 18.04.2022).

2. Digital marketing. URL: https://ru.wikipedia.org/wiki/ Digital_marketing (дата звернення 18.04.2022).

3. Вплив вебзастосунку на бізнес. URL: https://web.dev/i18n/ru/drive-business-success/ (дата звернення 18.04.2022).

4. Інтернет. URL: https://ru.wikipedia.org/wiki/ Інтернет (дата звернення 18.04.2022).

5. Стандарти WWW. URL: https://uk.wikipedia.org/wiki/WWW_Consort (дата звернення 18.04.2022).

6. Що таке IP. URL: https://2ip.ua/ru/blog/ip-address (дата звернення 18.04.2022).

7. DNS. URL: https://www.comss.ru/page.php?id=7359 (дата звернення 18.04.2022).

8. Фронтенд і бекенд. URL: https://internetdevels.ua/blog/front-end-development-vs-back-end-development (дата звернення 18.04.2022).

9. Структура вебзастосунків. URL: https://highload.today/veb-prilozheniya/ (дата звернення 18.04.2022).

10. JavaScript. URL: https://blog.ingate.ru/seo-wikipedia/java-script/ (дата звернення 18.04.2022).

11. Asynchronous JavaScript and XML. URL: https://medium.com/0xcode/ajax-asynchronous-javascript-and-xml-2af501d9ad6 (дата звернення 18.04.2022).

12. СУБД. URL: https://itglobal.com/ru-ru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/ (дата звернення 18.04.2022).

13. Огляд засобів створення інтерфейсів вебзастосунків на мові Java. URL: http://masters.donntu.org/2013/fknt/riabinin/library/article1.htm (дата звернення 18.04.2022).

14. Популярні вебзастосунки. URL: https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html (дата звернення 18.04.2022).

15. Переваги та недоліки вебзастосунків. URL: https://dou.ua/forums/topic/25444/ (дата звернення 18.04.2022).

16. Бази даних. URL: https://www.wwwmaster.ru/bazy-dannyh-dlya-web (дата звернення 18.04.2022).

17. Вебзастосунки. URL: https://dou.ua/lenta/articles/serverless-python/ (дата звернення 18.04.2022).

18. CRUD. URL: https://ru.wikipedia.org/wiki/ Digital_marketing (дата звернення 18.04.2022).

19. Цикли CRUD. URL: https://qna.habr.com/q/538933 (дата звернення 18.04.2022).

20. Процедури в SQL. URL: https://metanit.com/sql/sqlserver/11.1.php (дата звернення 18.04.2022).

21. Web-API. URL: http://www.joelsoftware.com/articles/APIWar.html (дата звернення 18.04.2022).

22. Ролі у застосунку. URL: https://medium.com/IvanZmerzlyi (дата звернення 18.04.2022).

23. Що таке CRUD. URL: https://stackify.com/what-are-crud-operations/ (дата звернення 18.04.2022).

24. Spring Framework. URL: https://spring.io/projects/spring-framework (дата звернення 18.04.2022).

25. Spring WebFlux. URL: https://www.baeldung.com/spring-webflux (дата звернення 18.04.2022).

26. Spring Data. URL: https://www.baeldung.com/spring-data (дата звернення 18.04.2022).

27. Spring Cloud. URL: https://www.baeldung.com/spring-cloud (дата

звернення 18.04.2022).

28. Огляд модулів Spring. URL: https://tproger.ru/articles/spring-modules-overview/ (дата звернення 18.04.2022).

29. Архітектура ПЗ. URL: https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101 (дата звернення 18.04.2022).

30. Архітектура Vaadin. URL: https://coderlessons.com/tutorials/veb-razrabotka/uchit-vaadin/vaadin-arkhitektura (дата звернення 18.04.2022).