

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій**

Курсова робота

з дисципліни «Програмування»

на тему: «Розклад та результати спортивних змагань»

Виконав:
студент 1 курсу, групи ІА-33
Василець Віталій Володимирович

(підпис)

Керівник:
асистент кафедри ІСТ
Мягкий Михайло Юрійович

(підпис)

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____
(підпис)

Київ – 2024 року

	ЗМІСТ	
ВСТУП		3
1 ВИМОГИ ДО СИСТЕМИ		4
1.1 Функціональні вимоги до системи		4
1.2 Нефункціональні вимоги до системи		4
2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ		5
2.1 Діаграма прецедентів		5
2.2 Опис сценаріїв використання системи		6
3 АРХІТЕКТУРА СИСТЕМИ		12
4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ		14
4.1 Загальна структура проекту		14
4.2 Компоненти рівня доступу до даних		15
4.3 Структура класів системи		16
ВИСНОВКИ		20
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ		22
ДОДАТОК А Лістинг програми		25

ВСТУП

Спорт - вид діяльності, який має доволі довгу історію. Люди завжди хотіли виявляти з посеред себе найбільш сильних, спритних та витривалих. Перші види спорту були доволі примітивними та простими, проте саме завдяки ним почали зароджуватися популярні в наш час спортивні ігри, а також люди почали цікавитись спортом як більш глобальним поняттям. Поступово, крок за кроком, спорт став масовою культурою. Він почав об'єднувати людей не тільки з одного міста, регіону, країни, але й людей з усього світу. Люди які займались спортом, стали бути не просто людьми які грають в ігри, але вони стали професіоналами в своїй справі. Через свою масовість, спорт став потребувати своєї структури. Тому на сьогоднішній час ми маємо багато турнірів та ще більше команд. З появою потреби фанатів спорту слідкувати за результатами своїх улюблених команд та спортсменів, з'явилася потреба в створенні певного ресурсу для цього. Кожен вболівальник бажає бачити як закінчився той чи інший матч, або коли буде наступний. Крім того, кожен вболівальник хотів би бачити таку інформацію в певному структурованому вигляді, завдяки якому можна було б зрозуміти яка команда грає, коли відбудеться матч, проти кого йде змагання, та який результат гри, якщо вона вже закінчилась. Для цього потрібно створити певний сервіс, який би відповідав вищезазначеним потребам, та виконував відповідні завдання.

Метою цієї роботи є створення такої системи, яка допомогла б людям отримати доступ до розкладу та результатів спортивних ігор. Також, ця система повинна виконувати такі функції:

- система повинна дозволяти адміністратору працювати з даними про змагання та їх результатами;
- система повинна дозволяти користувачам переглядати результати та розклад у зручному та зрозумілому форматі;
- користувач та адміністратор повинен мати змогу легко працювати з інтерфейсом системи, в залежності від своїх задач, які потрібно виконати.

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

Система повинна відповідати наступним функціональним вимогам:

- гість повинен мати змогу переглядати розклад змагань;
- гість повинен мати змогу переглядати результати змагань;
- гість повинен мати змогу проводити пошук результатів або розкладу ігор по назві команди;
- адміністратор повинен мати можливість створювати розклад змагань для команд;
- адміністратор повинен мати можливість редагувати розклад змагань для команд;
- адміністратор повинен мати можливість видаляти матч з розкладу змагань для команд;
- адміністратор повинен мати можливість вводити результати матчів.

1.2 Нефункціональні вимоги до системи

Система має відповідати наступним нефункціональним вимогам:

- система повинна бути реалізована як веб-інтерфейс;
- система повинна підтримуватись на різних пристроях;
- система повинна мати зрозумілу та відкриту структуру;
- інтерфейс повинен бути оптимальним для користувача;
- система повинна бути зручною для користування як для звичайного користувача, так і для адміністратора.

2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: звичайний користувач (гість) та адміністратор.

Адміністратор може виконувати ті самі дії, що і звичайний користувач, а також він має змогу додавати, редагувати та видаляти розклад змагань. Усі сценарії використання детальніше описані далі.

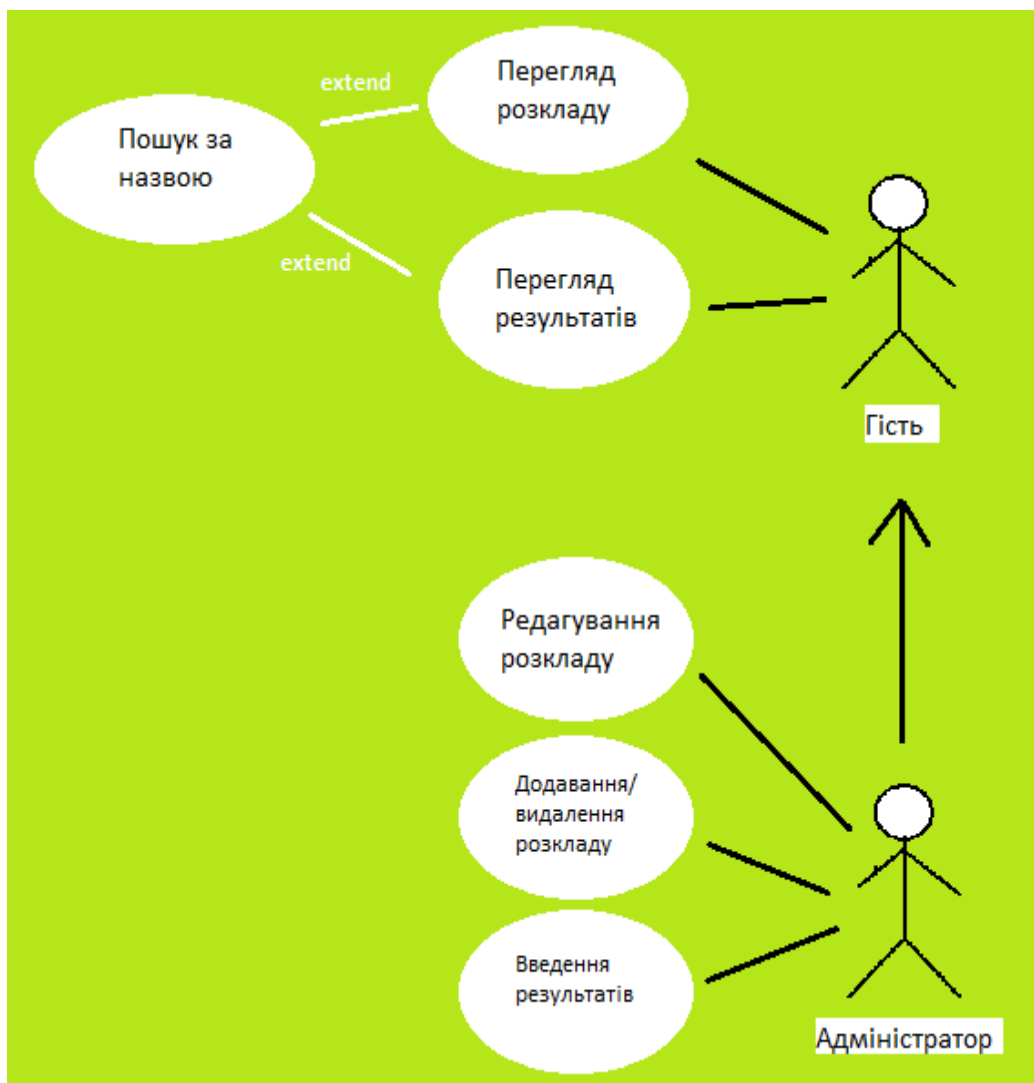


Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.17.

Таблиця 2.1 – Сценарій використання «Перегляд розкладу спортивних змагань»

Назва	Перегляд розкладу спортивних змагань
ID	1
Опис	Користувач, зайшовши на сторінку розкладу, може побачити які команди грають, та коли.
Актори	Гість, Адміністратор
Вигоди компанії	Це базовий функціонал для системи такої мети. Без цієї функції система не має сенсу.
Частота користування	Постійно
Тригери	Користувач заходить на сторінку з розкладом
Передумови	Немає
Постумови	Користувач має можливість бачити розклад змагань
Основний розвиток	Користувач заходить у застосунок та бачить розклад спортивних змагань
Альтернативні розвитку	—
Виняткові ситуації	—

В таблиці 2.2 представлений сценарій використання «Перегляд результатів спортивних змагань»

Таблиця 2.2 – Сценарій використання «Перегляд результатів спортивних змагань»

Назва	Перегляд результатів спортивних змагань
ID	1
Опис	Користувач, зайшовши на сторінку розкладу, може побачити результати матчів які закінчились
Актори	Гість, Адміністратор
Вигоди компанії	Це одна з потрібних функцій, без якої застосунок не буде користуватися попитом.
Частота користування	Постійно
Тригери	Користувач заходить на сторінку з розкладом
Передумови	Немає
Постумови	Користувач має можливість бачити результати змагань
Основний розвиток	Користувач заходить у застосунок та бачить результати спортивних змагань
Альтернативні розвитку	—
Виняткові ситуації	—

В таблиці 2.3 представлений сценарій використання «Пошук по назві команди»

Таблиця 2.3 – Сценарій використання «Пошук по назві команди»

Назва	Пошук по назві команди
ID	3
Опис	Користувач може переглядати розклад команди яка йому потрібна за допомогою пошуку.

Актори	Гість, Адміністратор
Вигоди компанії	Це дуже корисна функція, за допомогою якої сервіс матиме більшу популярність ніж у інших схожих без неї.
Частота користування	Часто
Тригери	Користувач вводить назву команди в поле пошуку
Передумови	Існування поля пошуку на сторінці
Постумови	Користувач бачить розклад та результати потрібної команди
Основний розвиток	Користувач вводить назву команди в поле для пошуку, тисне Enter, та бачить результати по запиту саме для потрібної команди
Альтернативні розвитку	—
Виняткові ситуації	Команди не має в розкладі

В таблиці 2.4 представлений сценарій використання «Додавання розкладу/результату»

Таблиця 2.4 – Сценарій використання «Додавання розкладу/результату»

Назва	Додавання розкладу/результату
ID	4
Опис	Адміністратор може додати нову гру або результат гри
Актори	Адміністратор

Вигоди компанії	Можливість додавати інформацію про нову гру, дозволить додавати змагання, навіть про яких не має інформації в інтернеті.
Частота користування	Часто
Тригери	Адміністратор використовує кнопку додавання гри
Передумови	Користувач зайшов в систему як адміністратор
Постумови	До сторінки перегляду розкладу додалася нова гра
Основний розвиток	Адміністратор використовує відповідну функцію, вводить основну інформацію, та додає нову гру
Альтернативні розвитку	–
Виняткові ситуації	–

В таблиці 2.5 представлений сценарій використання «Видалення змагання»

Таблиця 2.5 – Сценарій використання «Видалення змагання»

Назва	Видалення змагання
ID	5
Опис	Змагання видалється з сторінки перегляду
Актори	Адміністратор
Вигоди компанії	Можливість видаляти певні змагання, якщо вони будуть помилкові, або з різних інших причин

Частота користування	Рідко
Тригери	Адміністратор натиснув кнопку “Видалити змагання”
Передумови	Користувач зайшов в систему як адміністратор
Постумови	Змагання видаляється з сторінки перегляду
Основний розвиток	Адміністратор знаходить потрібну гру, натискає видалити, та гра прибирається з сторінки перегляду
Альтернативні розвитку	—
Виняткові ситуації	—

В таблиці 2.6 представлений сценарій використання «Редагування змагань»

Таблиця 2.6 – Сценарій використання «Редагування змагань»

Назва	Редагування змагань
ID	6
Опис	Адміністратор редагує інформацію про змагання
Актори	Адміністратор
Вигоди компанії	Така функція дає можливість виправляти помилки допущені при додаванні змагання, а також в разі непередбачуваної зміни
Частота користування	Рідко

Тригери	Адміністратор натискає на відповідну кнопку
Передумови	Адміністратор знаходить змагання з інформацією яку потрібно змінити
Постумови	Інформація про змагання змінена
Основний розвиток	Адміністратор натискає на відповідну кнопку, та змінює інформацію про відповідне змагання
Альтернативні розвитку	-
Виняткові ситуації	-

3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1.

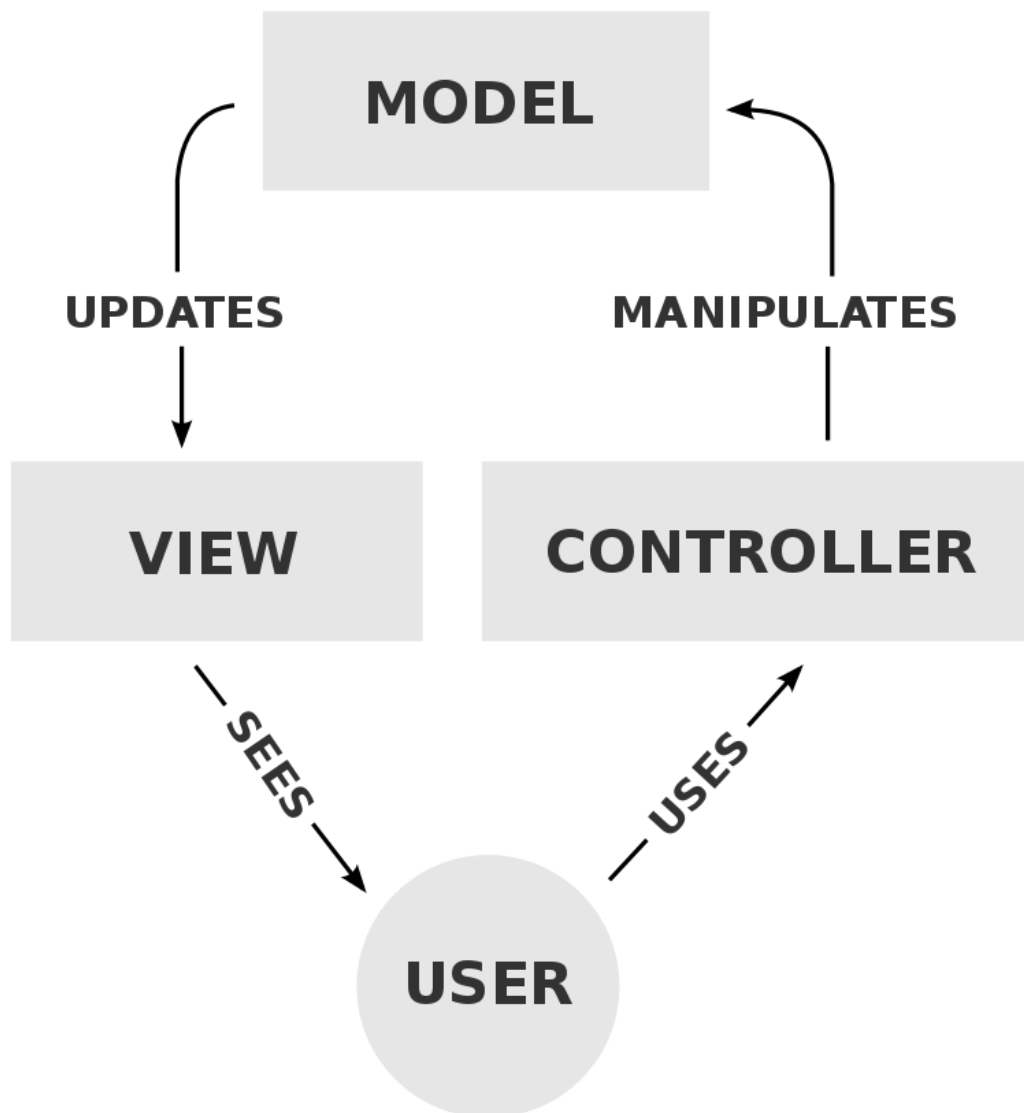


Рисунок 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

- візуальний інтерфейс;
- контролер;
- модель.

Візуальний, або графічний інтерфейс використовується для зовнішньої взаємодії користувача з застосунком. Тобто це те, що бачить користувач коли

користується ним. Далі його дії, або запити надходять до контролера, який маніпулює цими даними та направляє запити до серверної частини, або “моделі”. На серверній частині обробляється основна інформація, та виконуються основні дії, які потім переходять або оновлюються на стороні користувача. Тобто серверна частина обробляє ті запити які були опрацьовані та направлені контролером, та передає їх на графічний інтерфейс. Також модель обробляє дані з бази даних, та передає їх далі. База даних також зберігає в собі дані, які були отримані від попередніх етапів, а також дані які вже були задані.

І так циклічно обробляються запити. Від користувача дані переходять до контролеру. З контролеру дані надсилаються до моделі. З моделі вони переходять до графічного інтерфейсу. І вже у графічному інтерфейсі користувач може бачити результат своїх запитів. Загальна структура системи та її роботи працює наче “по колу”. Це забезпечує стабільність та зрозумілість роботи проекту.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Загальна структура проекту

Загальна структура проекту представлено на рисунку 4.1.

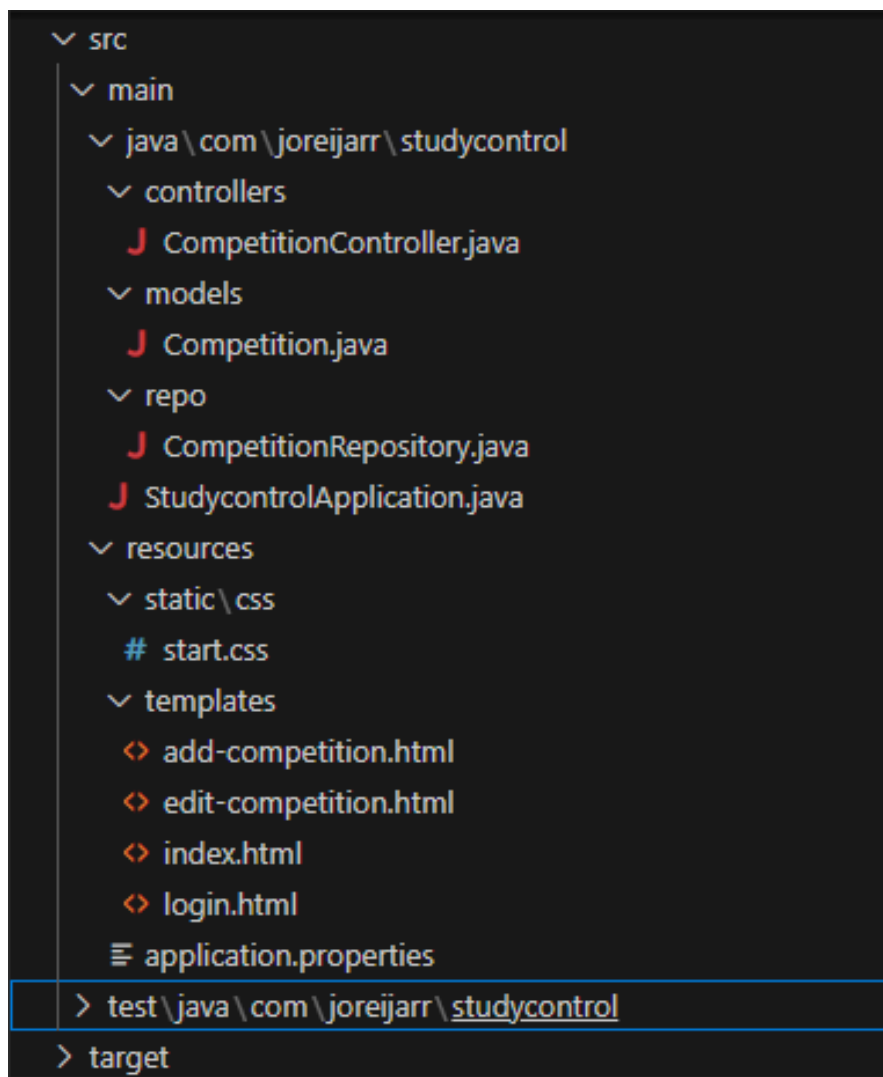


Рисунок 4.1 — Загальна структура проекту

Проект виглядає як ієрархічна система. Код об'єднаний в папці main. В папці controllers знаходиться файл CompetitionController.java, який використовується для загального мапінгу проекту. Далі, в папці models знаходиться файл Competition.java, в якому який зберігає поля та методи для роботи з базою даних. В папці repo, знаходиться CompetitionRepository.java, що відповідає за взаємодію з базою даних.

Конфігурується та запускається це все `StudycontrolApplication.java`. Далі йде тека `resources`, в якій зберігаються `css`-файл, та декілька `HTML`-сторінок: `add-competition` - сторінка для додавання змагання, `edit-competition` - сторінка для редагування змагання, `index` - головна сторінка, `login` - сторінка для входу.

4.2 Структура бази даних

База даних системи складається з однієї таблиці, її схема представлена на рисунку 4.2.

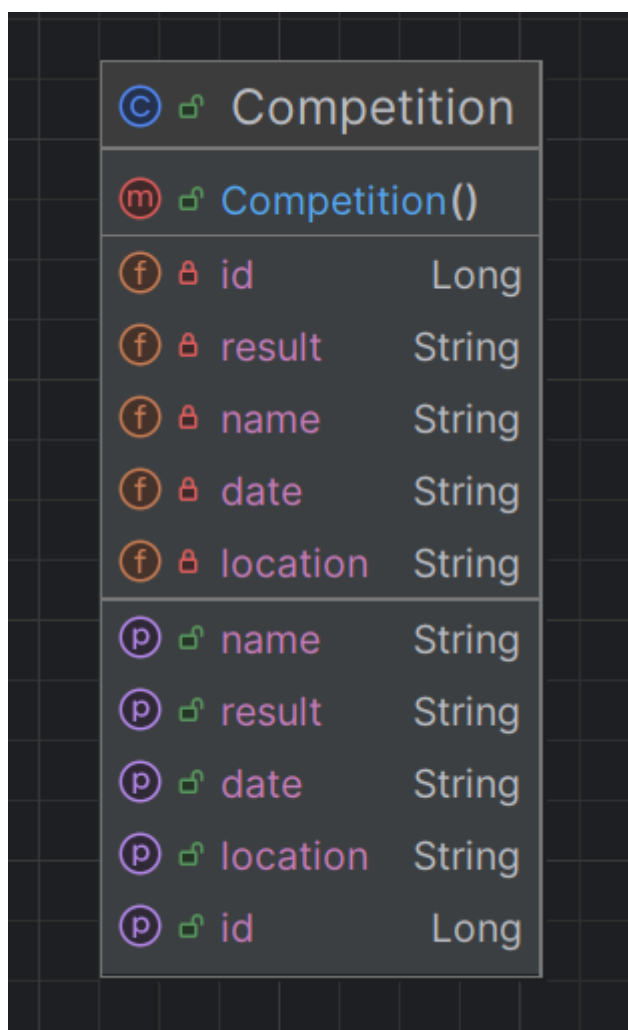


Рисунок 4.2 — Схема бази даних

Клас `Competition` у схемі бази даних має наступні поля та методи:

Поля класу:

– `id`: тип `Long` - унікальний ідентифікатор змагання.

- result: тип String - результати змагання.
- name: тип String - назва змагання.
- date: тип String - дата проведення змагання.
- location: тип String - місце проведення змагання.

Методи класу:

- Competition(): конструктор за замовчуванням.

Цей клас може використовуватися в проєкті на Java Spring для управління даними про змагання.

4.3 Структура класів системи

Діаграма класів системи представлена на рисунку 4.3.

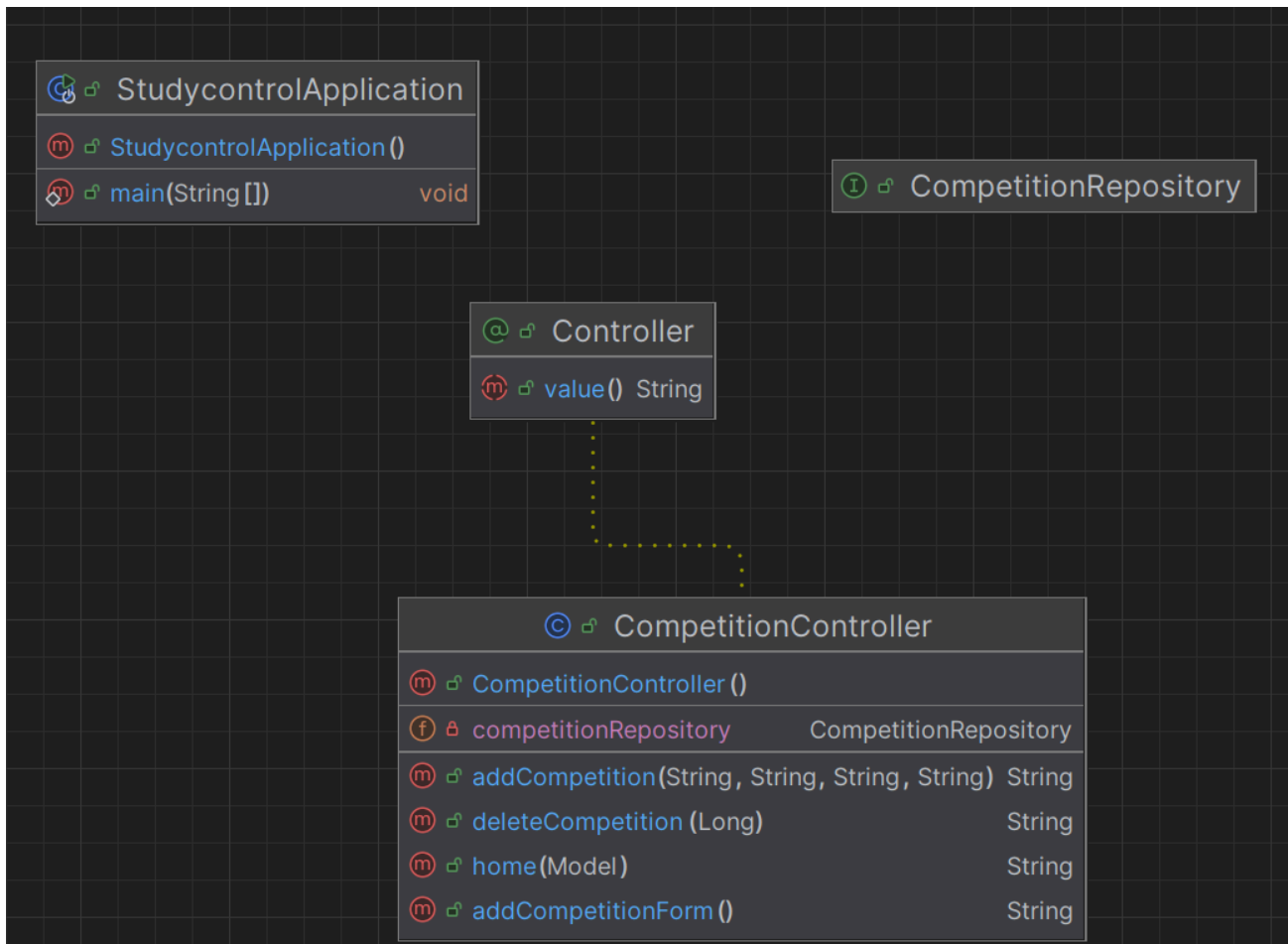


Рисунок 4.3 — Діаграма класів системи

Діаграма класів показує клас `CompetitionController`, який є частиною архітектури MVC у проекті на Java Spring.

`CompetitionController`

- Позначений анотацією `@Controller`, що ідентифікує його як компонент контролера у Spring MVC.
- Включає наступні методи та поля:

Поля:

- `competitionRepository`: тип `CompetitionRepository` - залежність, ін'єктована для доступу до операцій бази даних стосовно змагань.

Методи:

- `CompetitionController()`: конструктор класу.
- `addCompetition(String, String, String, String)`: приймає чотири параметри типу `String` (можливо, ім'я, дата, місце та результат змагання) і повертає `String`. Метод для додавання нового змагання.
- `deleteCompetition(Long)`: приймає параметр типу `Long` (ідентифікатор змагання) і повертає `String`. Метод для видалення змагання.
- `home(Model)`: приймає модель та повертає `String`. Можливо використовується для відображення головної сторінки або певного вигляду.
- `addCompetitionForm()`: метод, який повертає `String` та може відкривати форму для додавання нового змагання.
- `editCompetitionForm()`: метод, який повертає `String` та може відкривати форму для редагування змагання.
- `editCompetition(): (String, String, String, String)`: приймає чотири параметри типу `String` (можливо, ім'я, дата, місце та результат змагання) і повертає `String`. Метод для редагування змагання.

Анотація: `value()`

- Полягає в значенні `String`, можливо вказує на маршрут чи базовий шлях, який контролер обробляє.

StudycontrolApplication

- `StudycontrolApplication()`: конструктор класу.
- `main(String[])`: головний метод, який запускає додаток. Параметр `String[]` приймає аргументи командного рядка.

CompetitionRepository

- Інтерфейс, який призначений для взаємодії з базою даних. Він може містити методи для збереження, видалення, оновлення та отримання змагань.

Зв'язок між `CompetitionController` і `CompetitionRepository` вказує на використання репозиторія всередині контролера для доступу до бази даних.

`StudycontrolApplication` є вхідною точкою додатка, яка відповідає за його запуск і конфігурацію.

Залежності між класами представлено на рисунку 4.4.

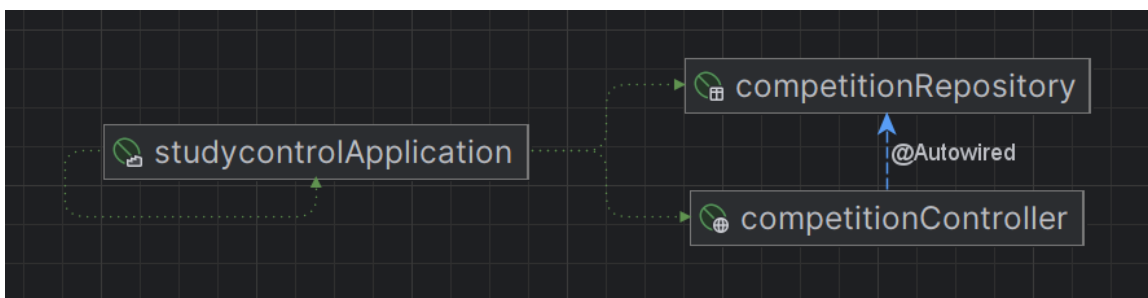


Рисунок 4.4 — Залежності проекту

На діаграмі представлені залежності між основними компонентами додатка Java Spring:

1. `studycontrolApplication` - головний клас додатка, що містить метод `main`. Він відповідає за запуск і конфігурацію Spring додатку. Цей клас з'єднується з іншими компонентами додатка через контекст Spring.
2. `competitionRepository` - це інтерфейс, який визначає методи для доступу до бази даних для об'єктів типу "Competition". Цей компонент містить анотацію `@Autowired`, що вказує на автоматичне впровадження

залежностей у Spring. Це означає, що Spring контейнер автоматично внедрить екземпляр `CompetitionRepository` до `competitionController`.

3. `competitionController` - контролер, який управляє веб-запитами для змагань. Він використовує `competitionRepository` для взаємодії з базою даних, що відображено стрілкою від `competitionRepository` до `competitionController`, підкреслюючи, що репозиторій впроваджується в контролер.

Ця діаграма демонструє важливість впровадження залежностей в архітектурі додатків на Spring, де компоненти легко інтегруються та управляються через контейнер Spring.

ВИСНОВКИ

Перед початком виконання роботи, основною її метою було створення такої системи, за допомогою якої користувач зміг би відслідковувати розклад та результати змагань. Це є основною концепцією багатьох спортивних веб-ресурсів, які також мають такий функціонал. Тому я вирішив порівняти різні такі ресурси між собою, та виділити їх сильні та слабкі сторони. Почавши своє дослідження, я наптрапив на веб-сайт з результатами змагань до різних матчів та команд. Проте цей ресурс мав один великий недолік: там були лише результати змагань. Перевіливши ще декілька ресурсів, я вирішив виділив у них ще декілька помітних недоліків: непривабливий інтерфейс, незручний спосіб подачі розкладу та результатів, неможливість редагувати інформацію, та інші недоліки які б не приваблювали нових людей до ресурсу. Перевіливши декілька інших застосунків, я прийшов до певного висновку про їх сильні та привабливі сторони. Тому зібравши їх до купи, я отримав певне уявлення про те які мають бути переваги у веб-ресурсу такого типу: привабливий та зручний для користування інтерфейс, зрозумілий формат подачі інформації яка потрібна користувачеві, можливість для адміністратора з легкістю додавати, редагувати, видаляти та змінювати інформацію про змагання, їх розклад та результати. Отримавши уявлення про те якою повинна бути моя система, я почав підготовку для безпосередньої розробки. Мені потрібно було уявити як буде виглядати мій застосунок. Тому окремо я написав невеликий сценарій для нього, де записав вищезазначені сильні сторони з певними корекціями.

Мовою розробки яку я використовував стала Java, з її потужним фреймворком Spring. Для більш зручного процесу розробки, я використовував середовище програмування VS Code від Microsoft, оскільки я вже доволі давно використовую його для написання коду та в своїх проєктах. Для створення графічного інтерфейсу я вирішив використовувати всім відому та базову мову розмітки HTML, з CSS та Thymeleaf. Ці засоби стали доволі потужними в написанні цього проєкту.

Перш за все я створив клас, який відповідав за запуск і конфігурацію всього додатку. Далі, певним чином структурувавши мій проєкт я почав розробку. Весь

“маппінг” я створював за допомогою контролера, в якому прописав окремі сторінки та методи для них, кожен з яким мав конкретну мету, в залежності від головної мети сторінки. Цей контролер мав головну мету - керувати веб-запитами. Створивши клас для основної інформації про змагання, через який з цими даними могли інші методи додатку, я також додав інтерфейс, який дозволяв методам це робити. Для оформлення інтерфейсу я створив декілька HTML файлів, які візуально структурували інформацію на сторінках. Для встановлення взаємозв'язку з базою даних я використовував Hibernate.

Об'єднавши всі вищезазначені елементи, структурувавши їх я отримав результат своєї роботи. Наслідком всього став застосунок, який допомагає фанатам спортивних ігор переглядати розклад та результати змагань своїх улюблених команд. Ресурс має певне графічне оформлення, та певну структуру подачі цієї інформації. Також для адміністратора ресурсу є можливість змінювати, додавати та видаляти інформацію про змагання та їх результати. Система є доволі корисною та цікавою з точки зору бізнесу, адже вона точно може зацікавити людей.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт спортивний подій Sport.ua [Електронний ресурс] – Режим доступу до ресурсу: <https://sport.ua/>
2. Веб-ресурс з результатами спортивних подій FlashScore [Електронний ресурс] – Режим доступу до ресурсу: <https://www.flashscore.com/>
3. Веб-ресурс з результатами спортивних подій Терикон [Електронний ресурс] – Режим доступу до ресурсу: <https://terrikon.com/uk/football/online>
4. Веб-ресурс з результатами спортивних подій Футбол24 [Електронний ресурс] – Режим доступу до ресурсу: <https://football24.ua/calendar/>
5. Веб-ресурс з результатами спортивних подій ФутболUA [Електронний ресурс] – Режим доступу до ресурсу: <https://football.ua/scoreboard/>
6. Baeldung - Spring Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.baeldung.com/spring-tutorial>
7. Building an Application with Spring Boot [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/guides/gs/spring-boot>
8. Learn how to build an application with minimal configuration. [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/spring-guides/gs-spring-boot>
9. HTML Tutorial [Електронний ресурс]. Режим доступу: <https://www.w3schools.com/html/>
10. MDN Web Docs: HTML Tutorial. [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
11. HTML.com: Learn HTML. [Електронний ресурс]. Режим доступу: <https://html.com/>
12. MDN Web Docs: CSS Tutorial [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
13. CSS-Tricks [Електронний ресурс]. Режим доступу: <https://css-tricks.com/>
14. CSS Basics [Електронний ресурс]. Режим доступу: <https://www.cssbasics.com/>
15. Spring Framework Documentation – [Електронний ресурс]. Режим

доступу: <https://spring.io/docs>

16. Spring Data JPA Documentation [Електронний ресурс]. Режим доступу: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

17. Spring Data REST Reference Guide [Електронний ресурс]. Режим доступу: <https://docs.spring.io/spring-data/rest/docs/current/reference/html/>

18. Сайт спортивний подій iSport.ua [Електронний ресурс]. Режим доступу: <https://isport.ua/ua/football/ukraine/1197250-matchi-sbornoj-ukrainy-raspisanie-i-rezultaty>

19. XSPORT - новини спорту онлайн [Електронний ресурс]. Режим доступу: <https://xsport.ua/ua/>

20. Офіційна документація Visual Studio Code з Java: <https://code.visualstudio.com/docs/languages/java>

21. Java in Visual Studio Code [Електронний ресурс]. Режим доступу: <https://code.visualstudio.com/docs/java/java-tutorial>

22. MDN Web Docs [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/en-US/docs/Web>

23. Спорт - Вікіпедія [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%80%D1%82>

24. Today's Top Sports Scores and Games [Електронний ресурс]. Режим доступу: <https://www.foxsports.com/scores>

25. Sport results today - Football, cycling, tennis scores [Електронний ресурс]. Режим доступу: <https://www.eurosport.com/score-center.shtml>

26. Scores & Fixtures - Football - BBC Sport [Електронний ресурс]. Режим доступу: <https://www.bbc.co.uk/sport/football/scores-fixtures>

27. Creating a Web Application With Spring Boot [Електронний ресурс]. Режим доступу: <https://dzone.com/articles/creating-a-web-application-with-spring-boot>

28. Building Modern Web Applications With Spring Boot and... [Електронний ресурс]. Режим доступу: <https://vaadin.com/docs/v14/flow/tutorial/overview>

29. Basic application in Java Spring Boot [Електронний ресурс]. Режим доступу: <https://www.geeksforgeeks.org/how-to-create-a-basic-application-in-java-spring-boot/>

30. Running a Java application on a webpage [Электронный ресурс]. Режим доступа:

<https://stackoverflow.com/questions/8606008/running-a-java-application-on-a-webpage>

31. How can we run a Java program on an HTML-coded... [Электронный ресурс]. Режим доступа:

<https://medium.com/@brajagopal.tripathi/how-can-we-run-a-java-program-on-an-html-coded-website-eaac7c600d2e>

ДОДАТОК А

Лістинг програми

Посилання на проект на Github <https://github.com/VasyletsVitalii/coursework>

Код програми:

CompetitionController.java:

```
package com.joreijarr.studycontrol.controllers;
import com.joreijarr.studycontrol.models.Competition;
import com.joreijarr.studycontrol.repo.CompetitionRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import javax.servlet.http.HttpSession;

@Controller
public class CompetitionController {

    @Autowired
    private CompetitionRepository competitionRepository;

    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute("competitions", competitionRepository.findAll());
        return "index";
    }

    @GetMapping("/add")
    public String addCompetitionForm(HttpSession session) {
        if (Boolean.TRUE.equals(session.getAttribute("loggedIn"))) {
            return "add-competition";
        }
        return "redirect:/login";
    }

    @PostMapping("/add")
```

```

    public String addCompetition(@RequestParam String name, @RequestParam String
location, @RequestParam String date, @RequestParam String result, HttpSession
session) {
        if (Boolean.TRUE.equals(session.getAttribute("loggedIn"))) {
            Competition competition = new Competition();
            competition.setName(name);
            competition.setLocation(location);
            competition.setDate(date);
            competition.setResult(result);
            competitionRepository.save(competition);
            return "redirect:/";
        }
        return "redirect:/login";
    }

@PostMapping("/delete")
    public String deleteCompetition(@RequestParam Long id, HttpSession session)
{
    if (Boolean.TRUE.equals(session.getAttribute("loggedIn"))) {
        competitionRepository.deleteById(id);
        return "redirect:/";
    }
    return "redirect:/login";
}

@GetMapping("/edit")
    public String editCompetitionForm(@RequestParam Long id, Model model,
HttpSession session) {
        if (Boolean.TRUE.equals(session.getAttribute("loggedIn"))) {
            Competition competition =
competitionRepository.findById(id).orElse(null);
            if (competition != null) {
                model.addAttribute("competition", competition);
                return "edit-competition";
            }
            return "redirect:/";
        }
        return "redirect:/login";
    }

@PostMapping("/edit")
    public String editCompetition(@RequestParam Long id, @RequestParam String
name, @RequestParam String location, @RequestParam String date, @RequestParam
String result, HttpSession session) {
        if (Boolean.TRUE.equals(session.getAttribute("loggedIn"))) {

```

```

                                Competition    competition    =
competitionRepository.findById(id).orElse(null);
    if (competition != null) {
        competition.setName(name);
        competition.setLocation(location);
        competition.setDate(date);
        competition.setResult(result);
        competitionRepository.save(competition);
    }
    return "redirect:/";
}
return "redirect:/login";
}

@GetMapping("/login")
public String loginForm() {
    return "login";
}

@PostMapping("/login")
public String login(@RequestParam String username, @RequestParam String
password, HttpSession session) {
    if ("admin".equals(username) && "admin".equals(password)) {
        session.setAttribute("loggedIn", true);
        return "redirect:/";
    }
    return "redirect:/login?error";
}

@GetMapping("/logout")
public String logout(HttpSession session) {
    session.invalidate();
    return "redirect:/login";
}
}

```

CompetitionRepository.java:

```

package com.joreijarr.studycontrol.repo;

import com.joreijarr.studycontrol.models.Competition;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository

```

```
public interface CompetitionRepository extends JpaRepository<Competition, Long>
{ }
```

Competition.java:

```
package com.joreijarr.studycontrol.models;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Competition {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String location;
    private String date;
    private String result;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }
}
```

```

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getResult() {
        return result;
    }

    public void setResult(String result) {
        this.result = result;
    }
}

```

StudycontrolApplication.java:

```

package com.joreijarr.studycontrol;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class StudycontrolApplication {
    public static void main(String[] args) {
        SpringApplication.run(StudycontrolApplication.class, args);
    }
}

```

start.css:

```

* {
    box-sizing: border-box;
}

body {
    background-color: #6576F0;
}

.clear {
    clear: both;
}

.log {

```

```
width: 400px;
margin: 5% auto;
background-color: #d0d8d9;
padding: 30px 0;
}
.log h2 {
  text-align: center;
  color: #474959;
  font-weight: bold;
  font-size: 26px;
  margin-bottom: 50px;
}
.log .input-cont {
  position: relative;
  margin: 0 50px 60px;
}
.log .input-cont:last-of-type {
  margin-bottom: 30px;
}
.log .input-cont input {
  position: relative;
  z-index: 1;
  width: 100%;
  height: 40px;
  outline: none;
  color: #474959;
  font-size: 22px;
  font-weight: 400;
  background: none;
  border: none;
}
.log .input-cont input:focus {
  outline: none;
}
.log .input-cont label {
  position: absolute;
  color: #474959;
  top: 0;
  left: 0;
  line-height: 40px;
  -webkit-transition: .3s;
  -moz-transition: .3s;
  -o-transition: .3s;
  transition: .3s;
}
```

```

.log .input-cont input:focus + label {
  margin-top: -30px;
  -webkit-transform: scale(.8);
  -moz-transform: scale(.8);
  -o-transform: scale(.8);
  transform: scale(.8);
  color: #bdbdbd;
}
.log .border1,
.log .border2 {
  position: absolute;
  height: 1px;
  background-color: #9E9E9E;
  left: 0;
  bottom: 0;
  width: 100%;
}
.log .border1::after,
.log .border1::before,
.log .border2::after,
.log .border2::before {
  content: "";
  position: absolute;
  bottom: 0;
  width: 0;
  height: 2px;
  -webkit-transition: .5s;
  -moz-transition: .5s;
  -o-transition: .5s;
  transition: .5s;
}
.log .border1::after,
.log .border2::after {
  right: 50%;
  background-color: #474959;
}
.log .border1::before,
.log .border2::before {
  left: 50%;
  background-color: #474959;
}
.log .input-cont input:focus ~ .border1::after,
.log .input-cont input:focus ~ .border1::before,
.log .input-cont input:focus ~ .border2::after,

```

```

.log .input-cont input:focus ~ .border2::before {
  width: 50%;
}
.log .check,
.log a {
  float: left;
  width: calc(50% - 50px);
  display: block;
  font-size: 12px;
  margin-bottom: 30px;
}
.log .check {
  margin-left: 50px;
}
.log a {
  text-align: right;
  text-decoration: none;
  color: #474959;
}
.log a:hover {
  text-decoration: underline;
  color: #F00;
}
.log form input[type="submit"] {
  display: block;
  margin: 0 auto 20px;
  border: 2px solid transparent;
  padding: 5px 20px;
  font-size: 22px;
  cursor: pointer;
  color: #474959;
  -webkit-transition: .5s;
  -moz-transition: .5s;
  -o-transition: .5s;
  transition: .5s;
}
.log form input[type="submit"]:focus {
  outline: none;
}
.log form input[type="submit"]:hover {
  border: 2px solid #474959;
}

```

add-competition.html:


```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Add Competition</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet"
/>
</head>
<body>
<header class="bg-primary text-white text-center py-3 mb-4">
    <div class="container">
        <h1>Sports Competition</h1>
        <nav class="nav justify-content-center">
            <a class="nav-link text-white" th:href="@{/}">Home</a>
            <a class="nav-link text-white" th:href="@{/add}">Add Competition</a>
        </nav>
    </div>
</header>
<div class="container">
    <h2 class="text-center my-4">Add New Competition</h2>
    <form th:action="@{/add}" method="post" class="w-50 mx-auto">
        <div class="form-group">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" class="form-control">
        </div>
        <div class="form-group">
            <label for="location">Location:</label>
            <input type="text" id="location" name="location"
class="form-control">
        </div>
        <div class="form-group">
            <label for="date">Date:</label>
            <input type="text" id="date" name="date" class="form-control">
        </div>
        <div class="form-group">
            <label for="result">Result:</label>
            <input type="text" id="result" name="result" class="form-control">
        </div>
        <button type="submit" class="btn btn-primary">Add Competition</button>
    </form>
</div>
<footer class="footer bg-light text-center py-3 mt-4">
    <div class="container">
        <p>&copy; 2024 Sports Competition</p>
    </div>

```

```

</footer>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js">
</script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></s
cript>
</body>
</html>

```

edit-competition.html:

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Edit Competition</title>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet"
/>
</head>
<body>
<header class="bg-primary text-white text-center py-3 mb-4">
  <div class="container">
    <h1>Sports Competition</h1>
    <nav class="nav justify-content-center">
      <a class="nav-link text-white" th:href="@{/}">Home</a>
      <a class="nav-link text-white" th:href="@{/add}">Add Competition</a>
      <a class="nav-link text-white" th:href="@{/logout}">Logout</a>
    </nav>
  </div>
</header>
<div class="container">
  <h2 class="text-center my-4">Edit Competition</h2>
  <form th:action="@{/edit}" method="post" class="w-50 mx-auto">
    <input type="hidden" name="id" th:value="${competition.id}">
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" class="form-control"
th:value="${competition.name}">
    </div>
    <div class="form-group">
      <label for="location">Location:</label>

```

```

        <input type="text" id="location" name="location" class="form-control"
th:value="${competition.location}">
    </div>
    <div class="form-group">
        <label for="date">Date:</label>
        <input type="text" id="date" name="date" class="form-control"
th:value="${competition.date}">
    </div>
    <div class="form-group">
        <label for="result">Result:</label>
        <input type="text" id="result" name="result" class="form-control"
th:value="${competition.result}">
    </div>
    <button type="submit" class="btn btn-primary">Save Changes</button>
</form>
</div>
<footer class="footer bg-light text-center py-3 mt-4">
    <div class="container">
        <p>&copy; 2024 Sports Competition</p>
    </div>
</footer>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js">
</script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></s
cript>
</body>
</html>

```

login.html:

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>

```

```

<div class="container">
  <h2 class="text-center my-4">Admin Login</h2>
  <form th:action="@{/login}" method="post" class="w-50 mx-auto">
    <div class="form-group">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username"
class="form-control">
    </div>
    <div class="form-group">
      <label for="password">Password:</label>
      <input type="password" id="password" name="password"
class="form-control">
    </div>
    <button type="submit" class="btn btn-primary">Login</button>
  </form>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js">
</script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></s
cript>
</body>
</html>

```

index.html:

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Competitions</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
<header class="bg-primary text-white text-center py-3 mb-4">
  <div class="container">
    <h1>Sports Competition</h1>
    <nav class="nav justify-content-center">
      <a class="nav-link text-white" th:href="@{/}">Home</a>
      <span th:if="${session.loggedIn != null}">

```

```

        <a class="nav-link text-white" th:href="@{/add}">Add
Competition</a>

        <a class="nav-link text-white"
th:href="@{/logout}">Logout</a>
    </span>
    <span th:if="${session.loggedIn == null}">
        <a class="nav-link text-white" th:href="@{/login}">Login</a>
    </span>
</nav>
</div>
</header>
<div class="container">
    <h2 class="text-center my-4">Competitions</h2>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Name</th>
                <th>Location</th>
                <th>Date</th>
                <th>Result</th>
                <th th:if="${session.loggedIn != null}">Actions</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="competition : ${competitions}">
                <td th:text="${competition.name}"></td>
                <td th:text="${competition.location}"></td>
                <td th:text="${competition.date}"></td>
                <td th:text="${competition.result}"></td>
                <td th:if="${session.loggedIn != null}">
                    <form th:action="@{/delete}" method="post" style="display:
inline;">
                        <input type="hidden" name="id" th:value="${competition.id}">
                        <button type="submit" class="btn btn-danger">Delete</button>
                    </form>
                    <a th:href="@{'/edit?id=' + ${competition.id}}" class="btn
btn-warning">Edit</a>
                </td>
            </tr>
        </tbody>
    </table>
</div>
<footer class="footer bg-light text-center py-3 mt-4">
    <div class="container">
        <p>&copy; 2024 Sports Competition</p>

```

```

    </div>
</footer>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js">
</script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></s
cript>
</body>
</html>

```

Вигляд бази даних:

+ Options

			id	date	location	name	result
<input type="checkbox"/>				1	12/06/05	Santiago Bernabeu	Barcelona vs Real
<input type="checkbox"/>				2	12/07/24	Kyiv	Shakhtar vs Dynamo 2:1

☐ Check all With selected: Edit Copy Delete Export

Сторінки веб-інтерфейсу:

На рисунку 5.1 зображений вигляд головної сторінки з точки зору звичайного користувача:



Рис. 5.1 - Сторінка звичайного користувача

На рисунку 5.2 зображено сторінка входу до панелі адміністратора:

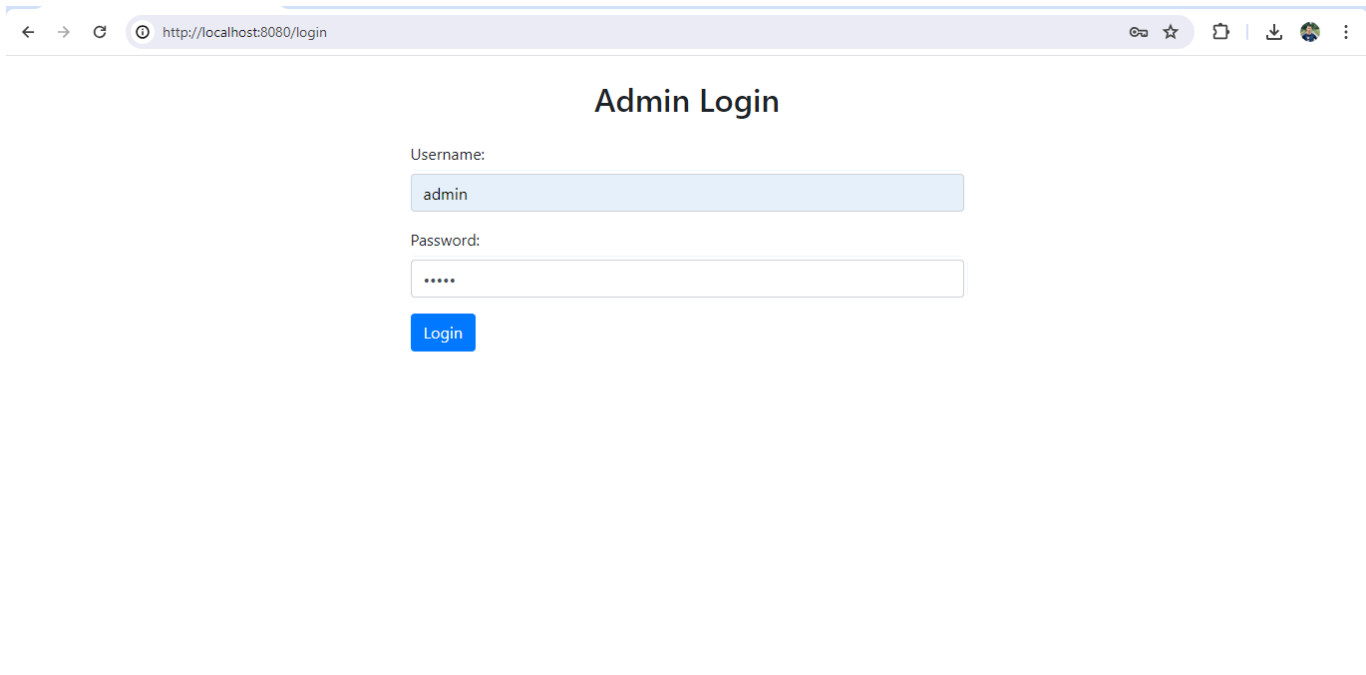


Рис. 5.2 - Сторінка входу

На рисунку 5.3 зображено головну сторінку з точки зору адміністратора:

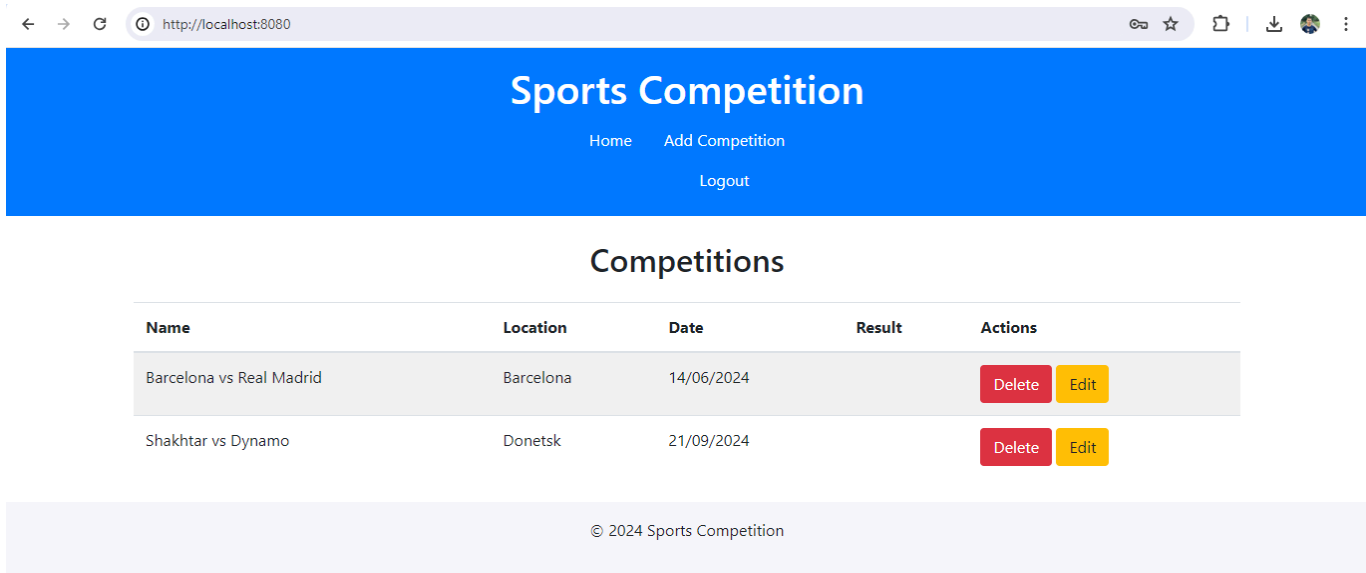
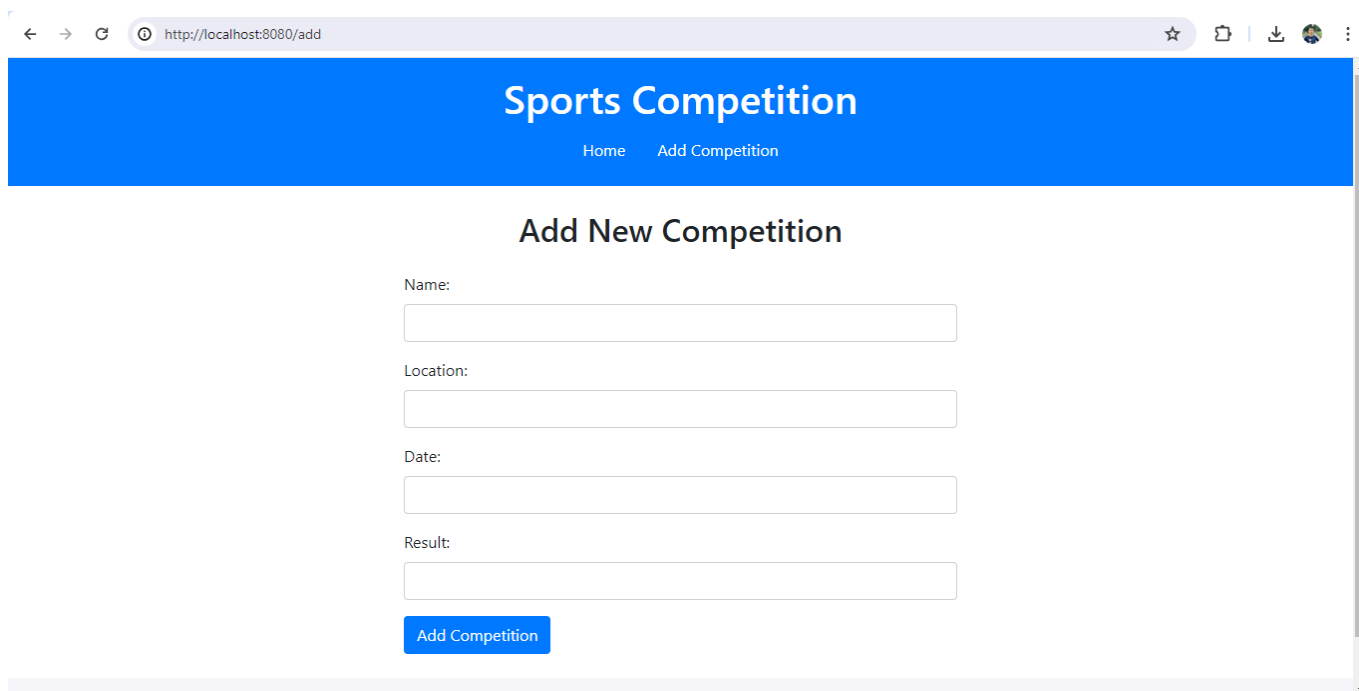


Рис. 5.3 - Сторінка Адміністратора

На рисунку 5.4 зображено сторінку з якої додається нове змагання:



← → ↻ http://localhost:8080/add ☆ 📄 ⬇️ 🌐 ⋮

Sports Competition

Home Add Competition

Add New Competition

Name:

Location:

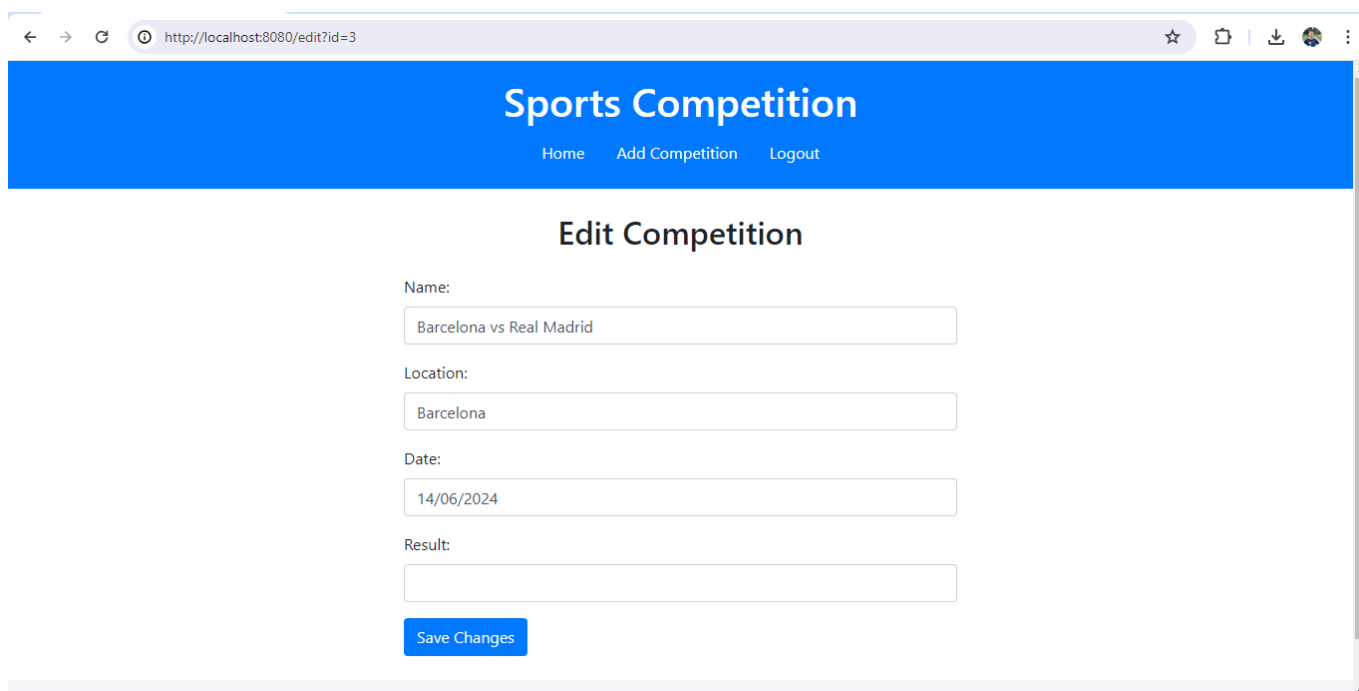
Date:

Result:

Add Competition

Рис. 5.4 - Сторінка додавання нового змагання

На рисунку 5.5 зображено сторінку редагування змагання:



← → ↻ http://localhost:8080/edit?id=3 ☆ 📄 ⬇️ 🌐 ⋮

Sports Competition

Home Add Competition Logout

Edit Competition

Name:

Location:

Date:

Result:

Save Changes

Рис. 5.5 - Сторінка редагування змагання