МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

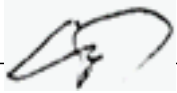
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  _ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку для автоматизації обробки результатів кінних скачок

Виконав: студент групи 4157ст3

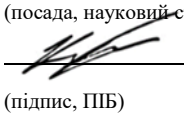
Михайлов О.В.

(підпис, ПІБ)

Керівник роботи:

доцент, к.т.н

(посада, науковий ступень вчене звання)

 Каіров В.О.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Михайлову Олександру Вікторовичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебзастосунку для автоматизації обробки результатів кінних скачок_____

Керівник роботи Каіров В.О. _____

Затверджені наказом ректора №153 від 08.04.2025 р.

2. Термін подання роботи: 02.06.2025 р._____

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів Титульний слайд, Аналоги програмного забезпечення, Аналіз вимог до програмного забезпечення, Архітектура програмного забезпечення, Ескізний проєкт програмного забезпечення, Технічний проєкт програмного забезпечення, Робочий проєкт програмного забезпечення, Результати розробки, Висновки, Заключний слайд _____

_____

_____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Михайлов О.В.
(підпис) (ПІБ)

Керівник роботи _____ Каіров В.О.
(підпис) (ПІБ)

АНОТАЦІЯ

Дана кваліфікаційна робота присвячена розв’язанню практичної задачі програмної інженерії, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку для автоматизації обробки результатів кінних скачок.

Комплексність розробки пов'язана з необхідністю створення інтуїтивного інтерфейсу для двох різних категорій користувачів, забезпечення безпеки даних та розробки механізмів для внесення, перевірки та збереження інформації в реальному часі. Невизначеність умов полягає у тому, що умови задачі можуть бути нечіткими або недостатньо визначеними, середовище може бути динамічним та вимоги користувача можуть бути змінені.

Робота включає в себе: опис та аналіз предметної галузі вебзастосунку для автоматизації обробки результатів кінних скачок, обґрунтування необхідності розробки програмного забезпечення, постановку задачі на розробку програмного забезпечення, проєкт програмного забезпечення, розділ з охорони праці та додатки.

Робота виконана на 130 сторінках друкованого тексту, містить 20 рисунків, 12 таблиць, 5 додатків та список використаних джерел з 9 найменувань.

ABSTRACT

This qualification work is dedicated to solving a practical problem of software engineering, characterized by the complexity and uncertainty of the conditions, namely the development of a web application for automating the processing of horse racing results.

The complexity of the development is associated with the need to create an intuitive interface for two different categories of users, ensuring data security and developing mechanisms for entering, checking and saving information in real time. The uncertainty of the conditions lies in the fact that the conditions of the problem may be vague or insufficiently defined, the environment may be dynamic and user requirements may change.

The work includes: description and analysis of the subject area of the web application for automating the processing of horse racing results, justification of the need for software development, statement of the task for software development, software project, section on labor protection and appendix.

The work is performed on 130 pages of printed text, contains 20 figures, 12 tables, 5 appendices and a list of used sources of 9 names.

ЗМІСТ

ВСТУП 7

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 10

1.1 Аналіз практичної задачі інженерії програмного забезпечення 10

1.2 Аналіз існуючих аналогів 11

1.3 Постановка задачі на розробку вебзастосунку для автоматизації обробки результатів кінних скачок 15

2 РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 19

2.1 Аналіз вимог до вебзастосунку для автоматизації обробки результатів кінних скачок. 19

2.1.1 Розробка моделі варіантів використання вебзастосунку для автоматизації обробки результатів кінних скачок 19

2.1.2 Розробка специфікацій прецедентів вебзастосунку для автоматизації обробки результатів кінних скачок 24

2.2 Розробка архітектури вебзастосунку для автоматизації обробки результатів кінних скачок 32

2.3 Проєкт вебзастосунку для автоматизації обробки результатів кінних скачок 35

2.3.1 Ескізний проєкт вебзастосунку для автоматизації обробки результатів кінних скачок 35

2.3.2 Технічний проєкт вебзастосунку для автоматизації обробки результатів кінних скачок 44

2.3.3 Робочий проєкт вебзастосунку для автоматизації обробки результатів кінних скачок 62

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 81

4 ОХОРОНА ПРАЦІ 83

4.1 Вступ 83

4.2 Основні законодавчі та нормативно-правові акти про охорону праці в Україні 83

4.3 Аналіз небезпечних і шкідливих факторів при роботі з комп’ютером 84

4.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів при роботі з комп’ютером 86

4.5 Санітарні норми мікроклімату виробничих приміщень 89

4.6 Пожежна безпека 89

4.7 Висновки 90
ВИСНОВКИ 91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 92
ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ 93
ДОДАТОК Б – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 99
ДОДАТОК В – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 103
ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 106
ДОДАТОК Д – ТЕКСТ ПРОГРАМИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК 116

ВСТУП

Кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку для автоматизації обробки результатів кінних скачок.

Дана тема є ініціативною, оскільки в Україні кінні скачки не є широко розповсюдженим видом спорту. Наразі більшість кінно-спортивних клубів зосереджені на навчанні верховій їзді, організації прогулянок та проведенні змагань з виїздки та конкуру.

Програмне забезпечення, яке використовують кінно-спортивні заклади, має низку обмежень, які потребують виправлення.

Наприклад, ПЗ клубу «МАГНАТ» дозволяє керувати заняттями з верхової їзди та організацією шоу, але не передбачає функцій для планування або проведення кінних скачок.

Система, яку використовує клуб «Олімп», підтримує облік прогулянок, уроків і фотосесій, проте також не включає модулів для управління скачками.

У школі верхової їзди «Спорт-Еліт» програмне забезпечення орієнтоване на підготовку до виїздки, конкуру та вольтижування, однак воно не пристосоване для організації або обліку кінних перегонів.

Враховуючи відсутність широкої практики проведення кінних скачок в Україні, розробка вебзастосунку для автоматизації обробки їх результатів може сприяти розвитку цього виду спорту в країні.

Комплексність розробки пов'язана з необхідністю створення інтуїтивного інтерфейсу для двох різних категорій користувачів, забезпечення безпеки даних та розробки механізмів для внесення, перевірки та збереження інформації в реальному часі.

Окрім того, враховувалися вимоги до масштабованості системи, її гнучкості та адаптивності. Вебзастосунок повинен забезпечувати надійну роботу з великими обсягами даних та можливість подальшого розширення функціоналу відповідно до потреб користувачів. Це рішення покликане підвищити ефективність роботи у сфері організації та проведення скачок, спрощуючи процеси для обох типів користувачів – адміністраторів та суддів.

Функціонал вебзастосунку буде розроблено таким чином, що адміністратори повинні мати можливість вносити і коригувати дані, необхідні для обліку результатів, тоді як судді повинні мати можливість вносити свої оцінки, коментарі та результати без необхідності прямої взаємодії з адміністраторами.

Метою роботи є створення вебзастосунку для автоматизації обробки результатів кінних скачок, що забезпечить зручність та ефективність у роботі адміністративного та суддівського складу.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Провести аналіз практичної задачі інженерії програмного забезпечення та опис предметної галузі програмного забезпечення.
2) Сформувати вимоги до розроблюваного вебзастосунку та розробити технічне завдання.
3) Проведення аналізу практичної задачі інженерії програмного забезпечення.
4) Аналіз існуючих аналогів.
5) Постановка задачі на розробку вебзастосунку для автоматизації обробки результатів кінних скачок.
6) Розробити архітектуру програмного забезпечення.
7) Провести розробку ескізного, технічного та робочого проєкту вебзастосунку для автоматизації обробки результатів кінних скачок.
8) Розробити документацію до вебзастосунку.

Об’єктом кваліфікаційної роботи є процес розробки програмного забезпечення для автоматизації обробки результатів кінних скачок.

У результаті вирішення цієї практичної задачі інженерії програмного забезпечення ми повинні отримати робочий вебзастосунок для автоматизації обробки результатів кінних скачок.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Вебзастосунок «Кінні скачки» – є застосунком, спрямованим на автоматизацію процесів, пов’язаних із роботою іподромів для проведення кінних скачок. Вебзастосунок охоплює низку процесів, необхідних для повноцінного функціонування іподрому: зберігання та обробка даних про змагання та їх результати, коней, їх власників та жокеїв.

Для реалізації цього вебзастосунку буде використано такі основні технології:

– Java – як основна мова програмування для реалізації серверної логіки.

– Spring Boot – фреймворк для створення бекенду, який дозволяє швидко реалізувати RESTful API та легко інтегрується з іншими компонентами.

– MySQL – реляційна система керування базами даних для зберігання інформації про коней, змагання, жокеїв, власників, суддів та інші об'єкти програмного забезпечення.

– Thymeleaf для створення користувацького інтерфейсу.

– Spring Security – для реалізації системи доступу з можливістю аутентифікації та авторизації різних типів користувачів.

Сервер додатку може бути запущений на локальному хості з використанням Tomcat (вбудованого в Spring Boot) або виділеного сервера.

База даних MySQL буде встановлена на тому ж сервері або у локальній мережі.

Доступ до вебінтерфейсу здійснюватиметься через браузер за внутрішньою IP-адресою.

Додаткові аспекти, які будуть враховуватися під час розробки вебзастосунку:

– Модульність та масштабованість – система має бути розширюваною (наприклад, додавання нових функцій, що необхідні для окремого іподрому).

– Валідація даних – перевірка на коректність заявок, унеможливлення подання неповної чи некоректної інформації.

– Резервне копіювання – оскільки система локальна, передбачено регулярне резервне копіювання бази даних.

– Інтерфейс користувача – має бути інтуїтивним і адаптованим під потреби персоналу іподрому.

– Журналювання подій – для відслідковування дій користувачів та забезпечення прозорості роботи системи.

Таким чином, створення вебзастосунку «Кінні скачки» є комплексною інженерною задачею, яка охоплює аналіз предметної області, проєктування системи, вибір відповідних технологій, реалізацію, тестування та локальне розгортання. Це рішення дозволить підвищити ефективність управління змаганнями та прозорість обліку їх результатів.

1.2 Аналіз існуючих аналогів

Перед початком розробки вебзастосунку проведемо аналіз існуючих аналогів, які надають інформацію щодо автоматизації кінних скачок. Метою аналізу є змога визначитись з функціоналом власного вебзастосунку та виявити успішні рішення, недоліки і відмінності в роботі існуючих аналогів.

Для початку, розглянемо типовий процес організації кінних скачок. У визначений іподромом термін власники коней подають заявки на участь у

змаганні, вказуючи дані про коня (кличка, порода, стать, вік), власника (ПІБ, адреса, телефон) та жокея (ПІБ, дата народження). Власником може бути фізична чи юридична особа. Жокей є партнером коня у змаганнях.

Фіксується інформація про змагання: дата, назва, місце проведення (іподром, адреса, довжина доріжки), результати (бали забігів, штрафні бали, зайняті місця).

Для оцінювання створюється суддівська колегія, до складу якої входить головний суддя та інші судді. Власники коней та близькі родичі жокеїв і тренерів не можуть бути суддями. Результати визначаються за сумою балів, враховуючи штрафи.

Жокеї мають особистий рейтинг, який розраховується за сумарною кількістю набраних балів.

Розуміння цих нюансів обробки інформації у сфері кінних скачок дозволяє краще оцінити, як подібна інформація представлена у вже реалізованих аналогах.

Першим із таких було розглянуто вебсайт кінно-спортивного клубу «МАГНАТ» [1]. Вебсторінка клубу «МАГНАТ» показує, що пропонує клуб, а саме: навчання верховій їзді, організацію кінних шоу та заходів розважального характеру.

Головну сторінку вебсайту представлено на рисунку 1.1.

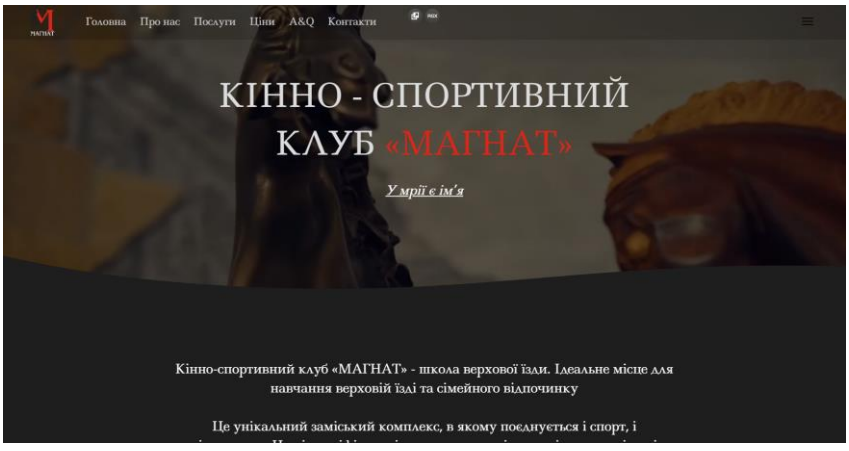


Рисунок 1.1 – Головна сторінка кінно-спортивного клубу «МАГНАТ»

Переваги:

- широкий спектр послуг у сфері навчання та шоу;
- підходить як для новачків, так і для досвідчених вершників;
- організація розважальних заходів.

Недоліки:

- відсутня інформація про проведення кінних скачок;
- обмежена деталізація щодо форматів навчання.

Також було розглянуто вебсайт кінного клубу «Олімп» [2]. «Олімп» – кінний клуб, що спеціалізується на прогулянках верхи, індивідуальних уроках верхової їзди та організації фотосесій з кіньми.

Головну сторінку представлено на рисунку 1.2.

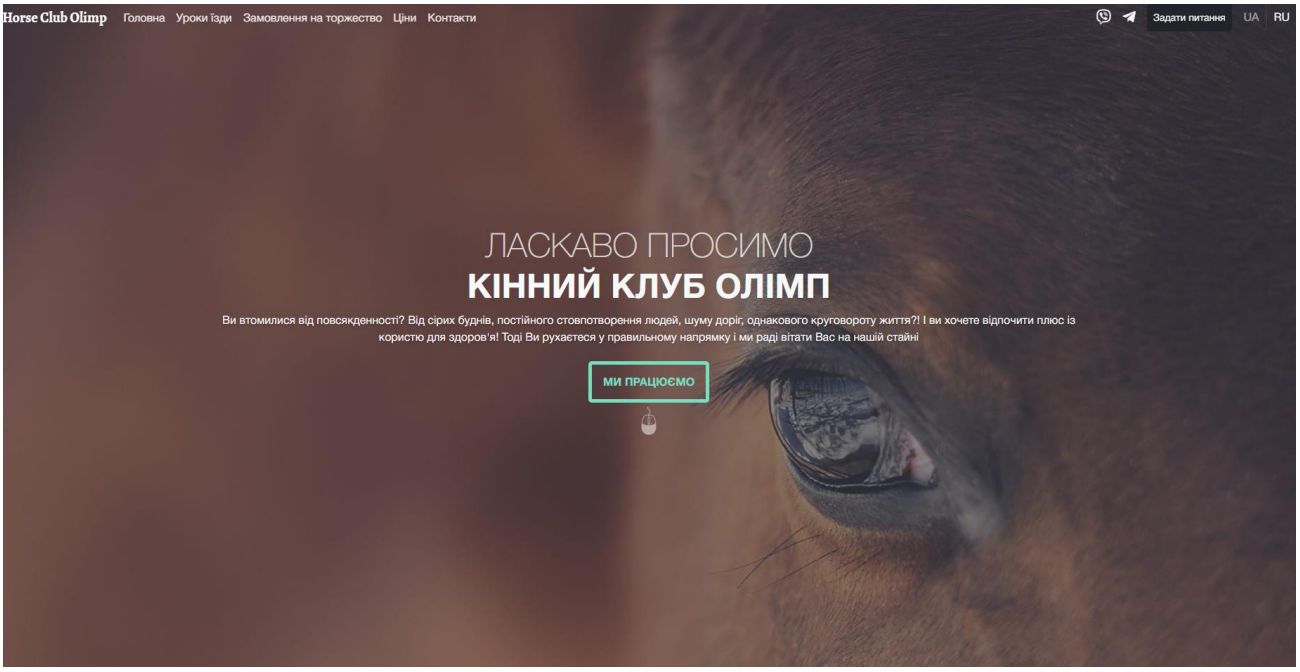


Рисунок 1.2 – Головна сторінка кінного клубу «Олімп»

Переваги:

- можливість спокійного відпочинку на природі;

- фотосесії з кіньми;
- індивідуальний підхід до клієнтів.

Недоліки:

- відсутня інформація про кінні змагання чи скачки;
- вузька спеціалізація на відпочинкових послугах.

Останнім було розглянуто вебсайт школи верхової їзди «Спорт-Еліт» [3]. «Спорт-Еліт» – школа верхової їзди, що фокусується на професійному навчанні, зокрема підготовці до змагань з виїздки, конкуру та вольтижування.

Головну сторінку представлено на рисунку 1.3.

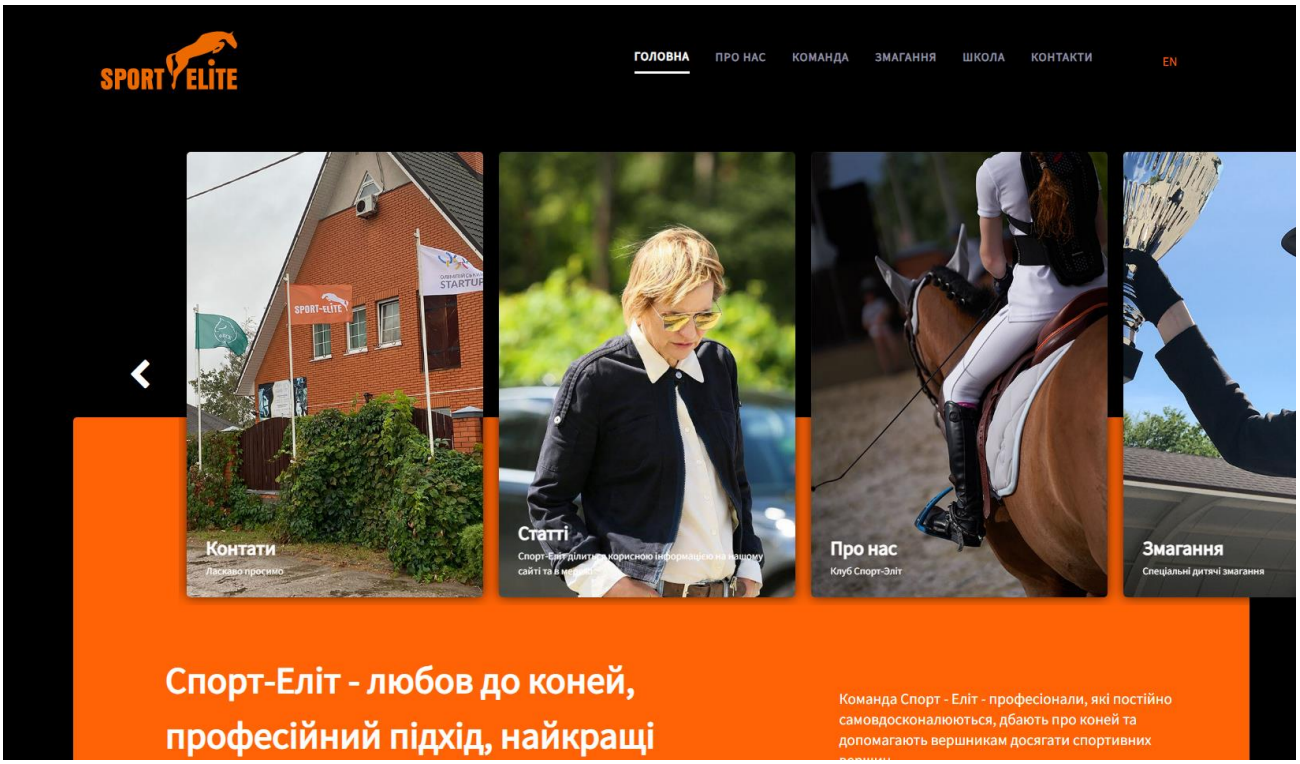


Рисунок 1.3 – Головна сторінка школи верхової їзди «Спорт-Еліт»

Переваги:

- акцент на спортивну підготовку;
- наявність програм для різних рівнів підготовки;
- спрямованість на участь у змаганнях.

Недоліки:

- відсутня інформація про скачки;
- обмежена інформація про розважальні чи аматорські послуги.

Виконавши аналіз існуючих аналогів, було виявлено їхні недоліки, а саме: відсутність кінних скачок у спектрі послуг, обмежена інформація про додаткові можливості клубів та вузька спеціалізація кожного закладу. Це свідчить про необхідність створення програмного забезпечення, що сприяло би поширенню кінних скачок.

1.3 Постановка задачі на розробку вебзастосунку для автоматизації обробки результатів кінних скачок.

Є необхідність розробити програмне забезпечення, яке було б універсальним та зручним для використання будь-якою із компаній, що займаються кінними скачками. Програмне забезпечення повинно відповідати наступним вимогам.

Вимоги до складу виконуваних функцій

- додавання, редагування та видалення інформації про іподроми;
- додавання, редагування та видалення інформації про змагання;
- додавання, редагування та видалення інформації про забіги;
- додавання, редагування та видалення інформації про результати;
- додавання, редагування інформації про порушення;
- додавання, редагування та видалення інформації про коней;
- додавання, редагування та видалення інформації про жокеїв;
- додавання, редагування та видалення інформації про власників;
- підрахунок результатів змагання;
- розрахунок рейтингу жокеїв;
- формування звітів з інформацією про скачки, дату їх проведення та

– результати;

– формування звітів з інформацією про загальну кількість жокеїв, коней,

– їх власників та змагань;

– формування звітів з відображення результатів змагань за заданим періодом (дата та час, іподром).

Вимоги до організації вхідних та вихідних даних

Введення даних відбувається за допомогою клавіатури та миші.

Взаємодія йде за допомогою вебзастосунку. Під час редагування та внесення інформації в базу даних вносяться такі дані:

– інформація про іподром (назва, адреса, довжина доріжки);

– інформація про змагання (назва змагання, дата змагання, іподром);

– інформація про забіг (назва забігу, оцінка, змагання);

– інформація про порушення (назва порушення, кількість знятих балів);

– інформація про коня (кличка, вік, порода, власник, стать);

– інформація про власника (ПІБ, адреса, телефон);

– інформація про жокея (ПІБ, дата народження, рейтинг);

– інформація про результат (місце, кількість балів, забіг, кінь, жокей).

Виведення даних відбувається за допомогою екрана ноутбуку або комп’ютера, інформація отримується у вигляді повідомлень та звітів.

Вихідними даними будуть повідомлення та звіти з інформацією про:

– додавання, редагування та видалення інформації про іподроми;

– додавання, редагування та видалення інформації про змагання;

– додавання, редагування та видалення інформації про забіги;

– додавання, редагування та видалення інформації про результати;

– додавання, редагування інформації про порушення;

– додавання, редагування та видалення інформації про коней;

– додавання, редагування та видалення інформації про жокеїв;

– додавання, редагування та видалення інформації про власників;

– підрахунок результатів змагання;

– розрахунок рейтингу жокеїв;

– формування звітів з інформацією про скачки, дату їх проведення та результати;

– формування звітів з інформацією про загальну кількість жокеїв, коней, їх власників та змагань;

– формування звітів з відображення результатів змагань за заданим періодом (дата та час, іподром).

Вимоги до інформаційної й програмної сумісності

Для роботи з програмним забезпеченням потрібно мати сервер, що матиме:

- операційну системи, що не є нижчою за Windows 10;

- систему управління базами даних MySQL.

Клієнтський комп’ютер повинен мати:

- операційну системи, що не є нижчою за Windows 10.

- веббраузер Chrome версії не нижче 119.0.6045.159

Вимоги до складу й параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на сервері наступної мінімальної комплекції:

- процесор: AMD Ryzen 5 5500 3.6 Ghz або аналогічний йому Intel Core i5-11400F;

- 4 ГБ оперативної пам’яті;

- 48 ГБ вільного місця на SSD.

- з’єднання з локальною мережею.

Також є клієнтський комп’ютер, що повинен задовольняти наступну мінімальну конфігурацію:

- AMD Ryzen 5 3600 3.6 Ghz або аналогічний йому Intel Core i5-9400F;

- 2 ГБ оперативної пам’яті;

- з’єднання з локальною мережею.

Розробка застосунку для автоматизації обробки результатів кінних скачок значно спростить та пришвидшить обробку інформації.

Більш детально постановка задачі сформульована у технічному завданні, що наведено у додатку А.

2 РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

Після виконання аналізу предметної галузі та з урахуванням виявлених вимог до програмного забезпечення, можна переходити на етап розробки та реалізації завдань.

В роботі використовується об’єктно-орієнтована методологія та уніфікована мова моделювання UML. В якості CASE-засобу обрані DrawIO та Erwin Data Modeler [4].

2.1 Аналіз вимог до вебзастосунку для автоматизації обробки результатів кінних скачок.

2.1.1 Розробка моделі варіантів використання вебзастосунку для автоматизації обробки результатів кінних скачок

Діаграма варіантів використання UML представляє собою візуальне відображення, яке демонструє різні можливі сценарії або способи застосування системи чи програмного продукту з точки зору користувача. Вона сприяє розумінню того, як користувачі взаємодіють із системою та які функції вони виконують.

Процес розробки діаграми варіантів використання включає чотири основні етапи: визначення та опис акторів, ідентифікація та опис прецедентів, створення діаграми варіантів використання та розробка специфікацій основних сценаріїв використання.

На першому етапі визначають акторів системи. Після проведення аналізу предметної області можна виділити дві групи акторів: Адміністратор та Суддя. Акторів системи представлено на рисунку 2.1

Рисунок 2.1 – Актори системи.

Короткий опис акторів представлено у таблиці 2.1

Таблиця 2.1 – Опис виявлених акторів системи

Актор	Короткий опис
Адміністратор	Займається введенням основних даних про змагання, має доступ до бази даних та всіх її таблиць, контролює роботу.
Суддя	Займається виставленням оцінок, заповнює інформацію про результати забігів, призначенням порушень під час забігу, не має доступу до бази даних, редагує дані у вебдодатку.

Виявимо та опишемо прецеденти системи. Їх представлено у таблиці 2.2

Таблиця 2.2 – Виявлені прецеденти системи

Код	Основний актор	Назва	Формулювання
1	2	3	4
A1	Адміністратор	Авторизація	Цей варіант використання дозволяє ввійти у профіль адміністратора в системі.
A2	Адміністратор	Обробити інформацію про іподроми.	Цей варіант використання дозволяє адміністратору обробити інформацію про іподроми.
A3	Адміністратор	Обробити інформацію про змагання.	Цей варіант використання дозволяє адміністратору обробити інформацію про змагання.
A4	Адміністратор	Обробити інформацію про власників.	Цей варіант використання дозволяє адміністратору обробити інформацію про власників.
A5	Адміністратор	Обробити інформацію про жокеїв.	Цей варіант використання дозволяє адміністратору обробити інформацію про жокеїв.
A6	Адміністратор	Обробити інформацію про коней.	Цей варіант використання дозволяє адміністратору обробити інформацію про коней.

Кінець таблиці 2.2

1	2	3	4
A7	Адміністратор	Обробити інформацію про результати.	Цей варіант використання дозволяє адміністратору обробити інформацію про результати.
A8	Адміністратор	Обробити інформацію про забіги.	Цей варіант використання дозволяє адміністратору обробити інформацію про забіги.
A9	Адміністратор	Обробити інформацію про клуби.	Цей варіант використання дозволяє адміністратору обробити інформацію про клуби.
J1	Суддя	Авторизація	Цей варіант використання дозволяє ввійти у профіль судді в системі.
J2	Суддя	Обробити інформацію про порушення	Цей варіант використання дозволяє судді обробити інформацію про порушення.
J3	Суддя	Обробити інформацію про результати	Цей варіант використання дозволяє судді обробити інформацію про результати.

Побудуємо діаграму варіантів використання системи у середовищі DrawIO. На рисунку 2.2 представлено діаграму варіантів використання вебзастосунку для автоматизації обробки результатів кінних скачок.

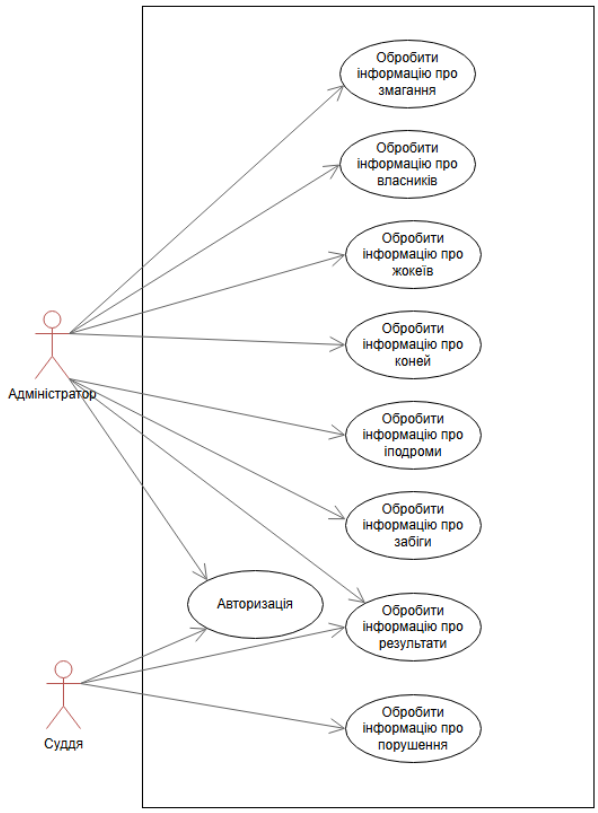


Рисунок 2.2 – Діаграма варіантів використання вебзастосунку для автоматизації обробки результатів кінних скачок

Прецедент «Обробити інформацію про змагання» передбачає роботу із такими таблицями: змагання, іподроми.

Прецедент «Обробити інформацію про власників» передбачає роботу із такими таблицями: власники.

Прецедент «Обробити інформацію про жокеїв» передбачає роботу із такими таблицями: жокеї.

Прецедент «Обробити інформацію про коней» передбачає роботу із такими таблицями: коні, власники.

Прецедент «Обробити інформацію про іподроми» передбачає роботу із такими таблицями: іподроми.

Прецедент «Обробити інформацію про забіги» передбачає роботу із такими таблицями: забіги, результати.

Прецедент «Обробити інформацію про результати» передбачає роботу із такими таблицями: результати, жокеї, коні, порушення, змагання.

Прецедент «Обробити інформацію про порушення» передбачає роботу із такими таблицями: порушення, результати.

2.1.2 Розробка специфікацій прецедентів вебзастосунку для автоматизації обробки результатів кінних скачок

Для детального опису варіантів використання, зображених на рисунку 2.2, розробляються їхні специфікації. Специфікація варіанту використання включає такі елементи: «Опис прецеденту», «Діючі особи прецеденту», «Зв'язок з іншими варіантами», «Передумови», «Базовий потік», «Альтернативні потоки», «Постумови» та «Особливості». Специфікації варіантів використання створеного програмного забезпечення наведені в таблицях 2.3 – 2.10.

Таблиця 2.3 – Специфікація варіанту використання «Авторизація»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу авторизуватись у системі.
Дійові особи прецеденту	Адміністратор, Суддя.
Передумови	Адміністратор та суддя повинні мати ідентифікатор (логін та пароль) для доступу в систему.
Основний потік подій	Користувач вводить ім’я та пароль. Система перевіряє дані на коректність. В залежності від прав доступу, завантажується той чи інший модуль. В разі некоректних даних виконується альтернативний потік подій.
Альтернативний потік подій	Невірний пароль: ввели невірний пароль. Дійовій особі пропонується можливість повторити введення або завершити варіант використання.
Постумови	Не визначені

Таблиця 2.4 – Специфікація варіанту використання «Обробити інформацію про змагання»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо змагань.
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними змагань.. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (назва, дата) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (назва, дата) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.5 – Специфікація варіанту використання «Обробити інформацію про власників»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо власників.
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними власників. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (ПІБ, адреса, телефон) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (ПІБ, адреса, телефон) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.6 – Специфікація варіанту використання «Обробити інформацію про жокеїв»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо жокеїв.
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними жокеїв. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (ПІБ, дата народження, рейтинг) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (ПІБ, дата народження, рейтинг) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.7 – Специфікація варіанту використання «Обробити інформацію про коней»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо коней.
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними коней.. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (прізвисько, стать, порода, вік) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (прізвисько, стать, порода, вік) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.8 – Специфікація варіанту використання «Обробити інформацію про іподроми»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо іподромів.

Кінець таблиці 2.8

1	2
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними іподромів. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (назва, адреса, довжина доріжки) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (назва, адреса, довжина доріжки) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.9 – Специфікація варіанту використання «Обробити інформацію про забіги»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо забігів.

Кінець таблиці 2.9

1	2
Дійові особи прецеденту	Адміністратор
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними забігів. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (назва забігу, кінь, місце у забігу, кількість балів) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (назва забігу, кінь, місце у забігу, кількість балів) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.10 – Специфікація варіанту використання «Обробити інформацію про результати»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо результатів.

Кінець таблиці 2.10

1	2
Дійові особи прецеденту	Суддя
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Адміністратор ініціює управління даними результатів.. Система дає можливість обрати варіант обробки (додавання, видалення, модифікація, перегляд). Адміністратор обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі три варіанти основного потоку: 1.Додавання даних Система надає можливість внесення даних. Адміністратор вводить необхідні дані (кінь, змагання, жокей, місце, результат) та надає системі команду зберегти їх. Система зберігає дані. 2.Видалення даних Система надає можливість адміністратору спозиціонуватись на запис для видалення. Адміністратор позиціонується на необхідний запис та надає команду на видалення. Система видаляє запис. 3.Модифікація даних Система надає можливість адміністратору спозиціонуватись на запис для модифікації. Адміністратор позиціонується на необхідний запис. Система надає можливість модифікації даних. Адміністратор робить необхідні зміни в даних (кінь, змагання, жокей, місце, результат) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.11 – Специфікація варіанту використання «Обробити інформацію про порушення»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору обробити інформацію щодо порушень.

Кінець таблиці 2.11

1	2
Дійові особи прецеденту	Суддя
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Суддя ініціює управління даними забігів. Суддя обирає варіант обробки. Система виводить список усіх змагань. При цьому можливі варіанти основного потоку (додавання, модифікація, перегляд).: 1.Додавання даних Система надає можливість внесення даних. Суддя вводить необхідні дані (назва, кількість знятих балів) та надає системі команду зберегти їх. Система зберігає дані. 2.Модифікація даних Система надає можливість судді спозиціонуватись на запис для модифікації. Суддя позиціонується на необхідний запис. Система надає можливість модифікації даних. Суддя робить необхідні зміни в даних (назва, кількість знятих балів) та надає системі команду зберегти їх. Система зберігає дані.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

2.2 Розробка архітектури вебзастосунку для автоматизації обробки результатів кінних скачок

Відповідно до вимог до вебзастосунку для автоматизації обробки результатів кінних скачок, було обрано найбільш підходящий архітектурний стиль – MVC (Model-view-controller).

Патерн MVC (Model-View-Controller) використовується для проєктування програмного забезпечення:

– Модель (Model): відповідає за управління даними та бізнес-логікою. Вона обробляє бізнес-правила й надає доступ до даних іншим компонентам.

– Представлення (View): відображає дані з Моделі користувачу та

передає введену інформацію Контролеру. Представлення пасивне, тобто не взаємодіє напряму з Моделлю.

– Контролер (Controller): виступає посередником між Моделлю та Представленням. Він обробляє дані, отримані від користувача, та оновлює Модель і Представлення відповідно до змін. Контролер також виконує перевірку та перетворення введених даних.

На рисунку 2.3 представлено діаграму взаємодії використання патерну з користувачем.

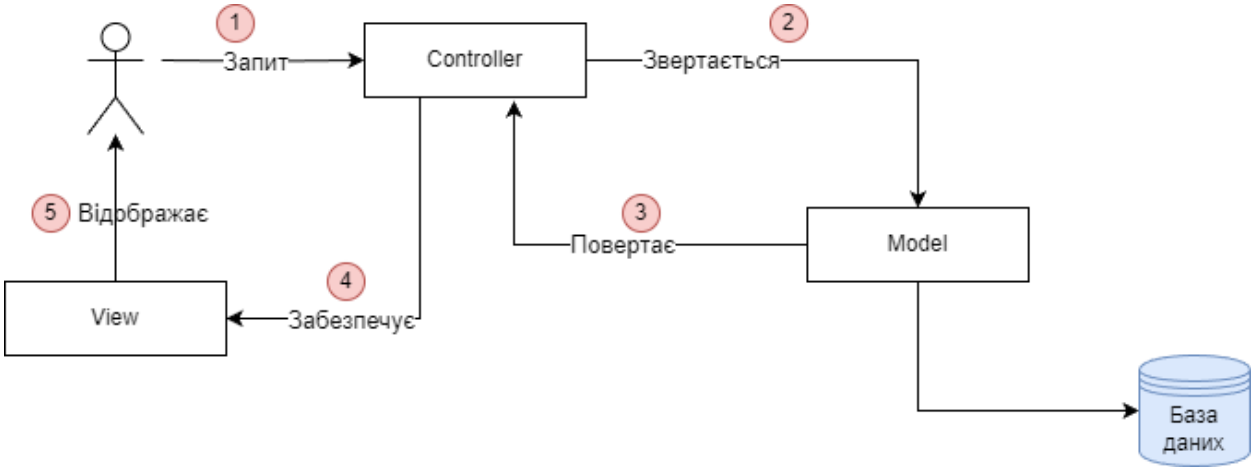


Рисунок 2.3 – Діаграма взаємодії використання патерну MVC

Розробка діаграми компонентів вебзастосунку для автоматизації обробки результатів кінних скачок.

Діаграма компонентів є статичною структурною діаграмою, що відображає поділ програмної системи на окремі структурні елементи та зв’язки (залежності) між ними. У ролі фізичних компонентів можуть виступати файли, бібліотеки, модулі, виконувані файли та пакети.

Діаграма компонентів вебзастосунку для автоматизації обробки результатів кінних скачок представлена на рисунку 2.4

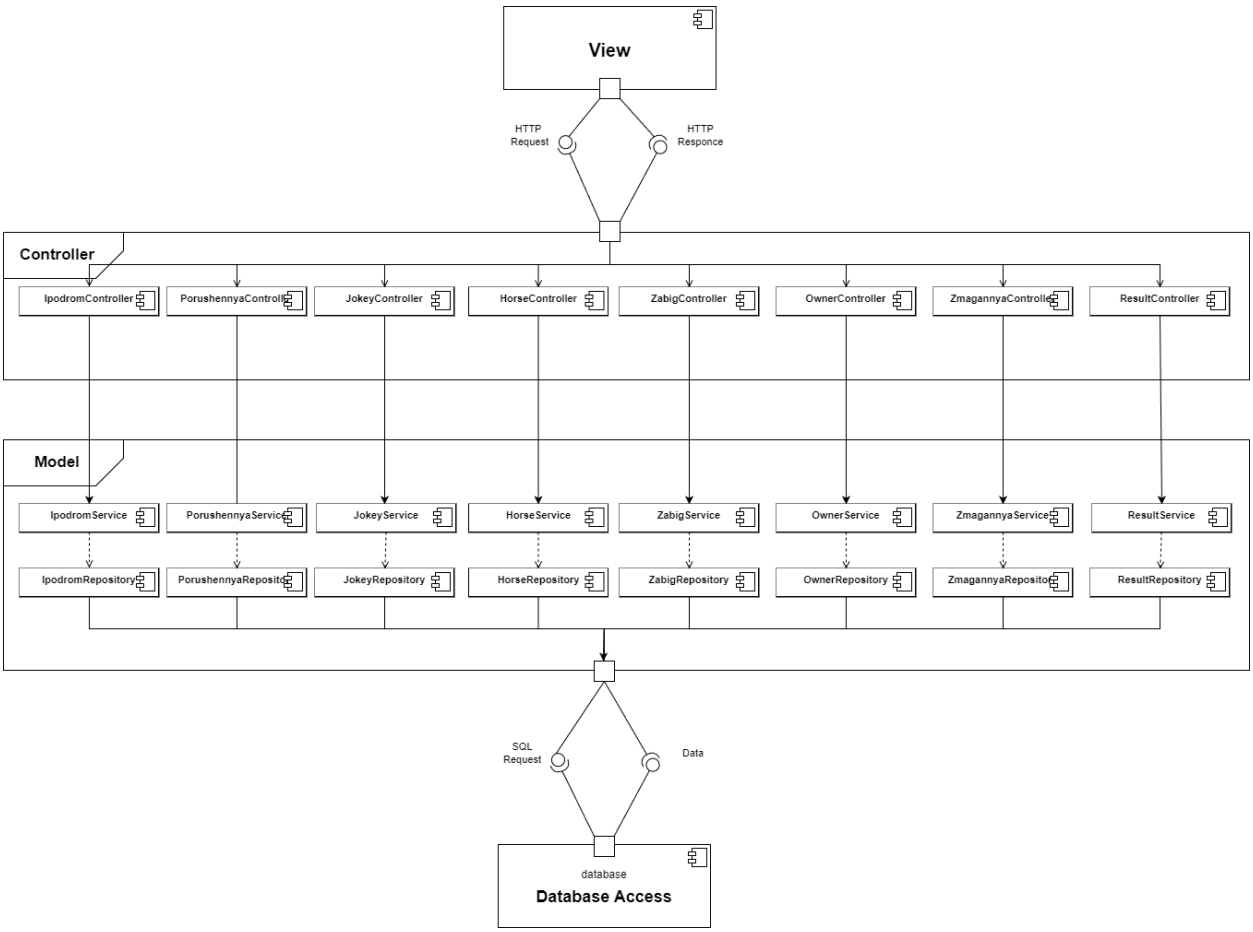


Рисунок 2.4 – Діаграма компонентів вебзастосунку для автоматизації обробки результатів кінних скачок

Розробка діаграми розгортання вебзастосунку для автоматизації обробки результатів кінних скачок

Діаграма розгортання є діаграмою, що відображає архітектурну структуру системи, що складається з різних обчислювальних вузлів та зв’язків між ними. Діаграма розгортання використовується для візуалізації фізичного розміщення системних компонентів на різних вузлах та зв’язків між ними.

Діаграму розгортання вебзастосунку для автоматизації обробки результатів кінних скачок представлено на рисунку 2.5

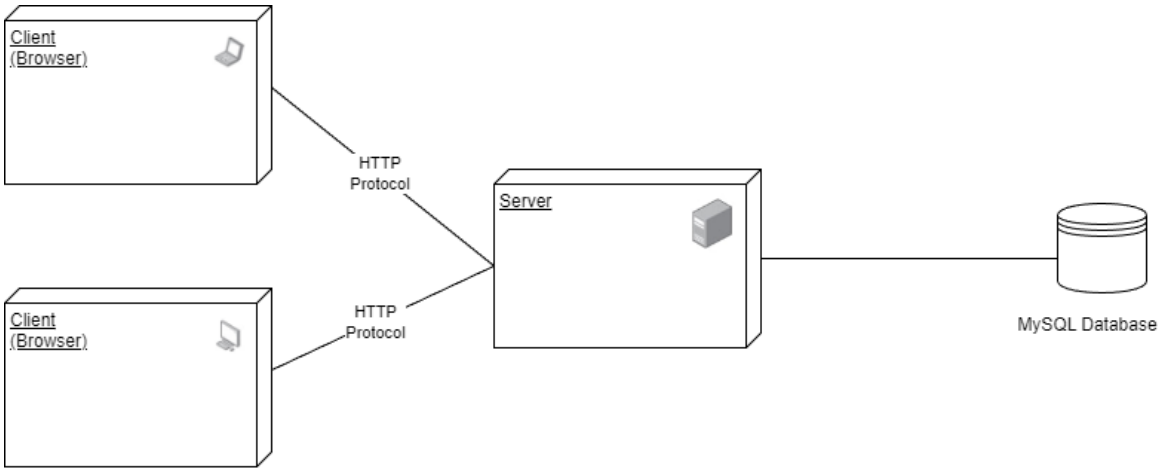


Рисунок 2.5 – Діаграма розгортання вебзастосунку для автоматизації обробки результатів кінних скачок

2.3 Проєкт вебзастосунку для автоматизації обробки результатів кінних скачок

При розробці проєкту вебзастосунку були виконані роботи з ескізного, технічного та робочого проєктування.

2.3.1 Ескізний проєкт вебзастосунку для автоматизації обробки результатів кінних скачок

Для систематизації представлення сценаріїв варіантів використання побудовано діаграми діяльності для усіх основних прецедентів системи, відповідно до яких створено проєкт користувацького інтерфейсу, а також розроблено інформаційну модель у вигляді концептуальної моделі даних.

Розробка діаграм діяльності основних прецедентів вебзастосунку для автоматизації обробки результатів кінних скачок

UML-діаграма діяльності є візуальною формою представлення графа дій.

Такий граф можна розглядати як різновид графа станів скінченного автомата, де вершини відповідають певним діям, а переходи між ними відбуваються після завершення цих дій.

Цей тип діаграми ілюструє, як система змінюється в динаміці, показуючи, як керуючі потоки переміщуються від однієї дії до іншої. Крім того, вона дозволяє зображати паралельне виконання дій та альтернативні шляхи розвитку подій. Основні складові елементи діаграми діяльності включають: «Дія», «Перехід», «Умова вибору», «Початок», «Завершення» і «Лінії синхронізації».

На рисунках 2.6 – 2.7 наведені приклади діаграм діяльності для таких процесів, як «Авторизація» та «Обробка інформації про коня».

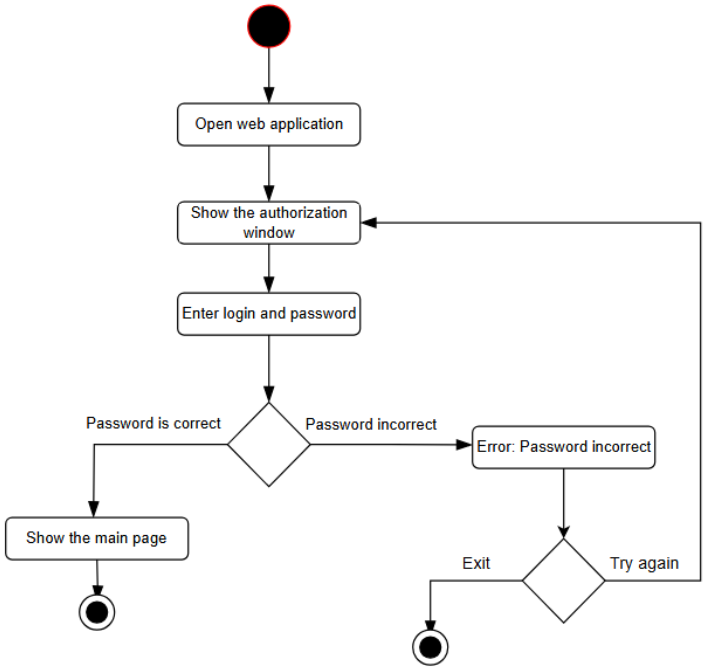


Рисунок 2.6 – Діаграма діяльності варіанту використання «Авторизація»

На рисунку 2.7 представлено діаграму діяльності варіанта використання «Обробити інформацію про коня».

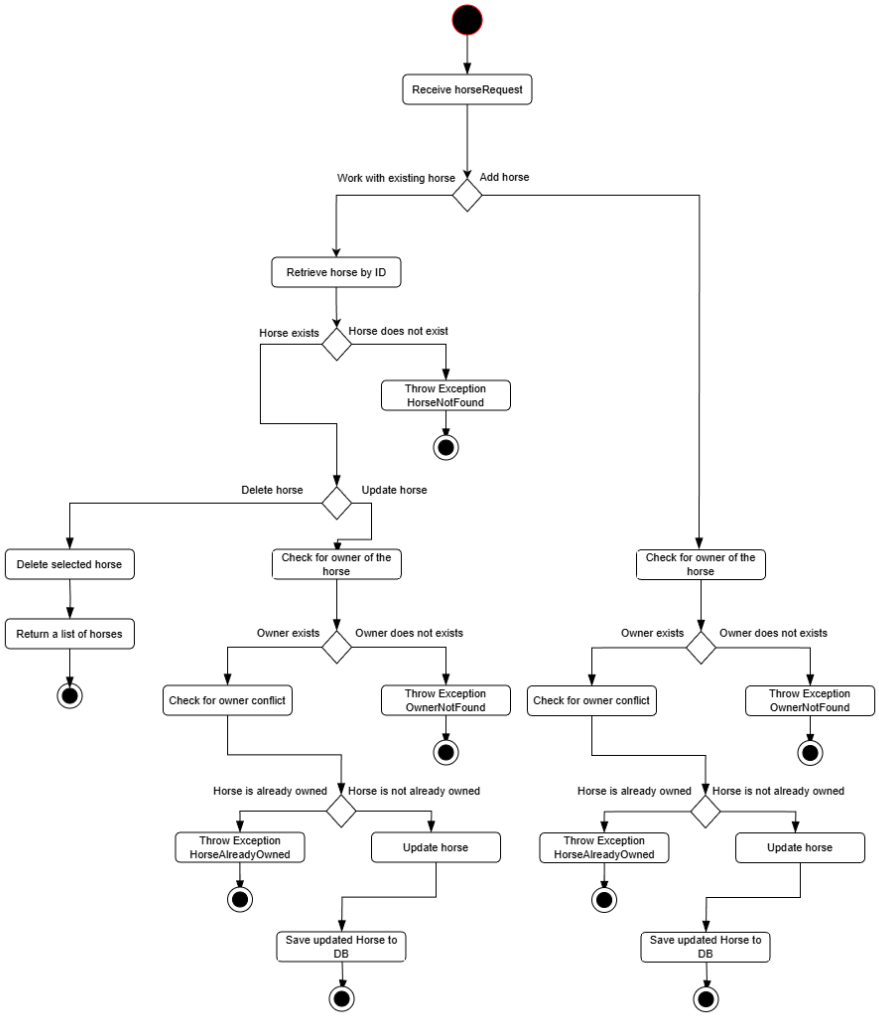


Рисунок 2.7 – Діаграма діяльності варіанту використання «Обробка інформації про коня»

Розробка концептуальної моделі даних вебзастосунку для автоматизації обробки результатів кінних скачок

Концептуальне проєктування – це процес створення уявлення (схеми чи моделі) бази даних на концептуальному рівні, що передбачає визначення сутностей, їхніх атрибутів і зв’язків між ними. Цей процес не залежить від типу обраної моделі даних і не враховує аспектів фізичної реалізації.

Одним із методів відображення концептуальної моделі є діаграма типу

«сутність–зв’язок» (Entity-Relationship, ER-модель).

Визначимо сутності системи.

Сутність – це реальний або уявний об’єкт, який можна розпізнати й ідентифікувати. Вона представляє ключові об’єкти чи поняття, що стосуються розроблюваної системи. Кожна сутність має свої атрибути (властивості або характеристики), які дозволяють створити її детальний опис.

Нижче наведено перелік сутностей вебзастосунку для автоматизації обробки результатів кінних скачок:

- Забіг;
- Порушення;
- Результат;
- Іподром;
- Змагання;
- Кінь;
- Власник;
- Жокей.

Визначимо атрибути сутностей.

Список атрибутів сутності Забіг:

- Ідентифікатор забігу;
- Оцінка;
- Назва забігу;
- Змагання.

Список атрибутів сутності Порушення:

- Ідентифікатор порушення;
- Назва порушення;
- Кількість знятих балів.

Список атрибутів сутності Результат:

- Ідентифікатор результату;

- Кількість балів;
- Кінь;
- Жокей;
- Забіг;
- Місце.

Список атрибутів сутності Іподром:

- Ідентифікатор іподрому;
- Назва;
- Довжина доріжки;
- Адреса.

Список атрибутів сутності Змагання:

- Ідентифікатор змагання;
- Дата змагання;
- Назва змагання;
- Іподром.

Список атрибутів сутності Кінь:

- Ідентифікатор коня;
- Порода;
- Кличка;
- Вік;
- Стать;
- Власник.

Список атрибутів сутності Власник:

- Ідентифікатор власника;
- ПІБ;
- Телефон;
- Адреса;

Список атрибутів сутності Жокей:

– Ідентифікатор жокею;
– ПІБ;
– Дата народження;
– Рейтинг.

Побудуємо модель «сутність-зв’язок»

Зв'язок – це взаємозв'язок між сутностями, який має значення в певній предметній області. Такий зв'язок може мати власні атрибути та існувати у різних формах: між двома окремими сутностями (бінарний зв’язок), між кількома сутностями (n-арний зв’язок) або навіть між однією і тією ж сутністю (рекурсивний зв’язок). Ступінь зв’язку визначає, скільки екземплярів однієї сутності може бути пов’язано з екземплярами іншої.

У таблиці 2.12 наведено приклад таблиці сутностей і зв’язків між ними.

Таблиця 2.12 – Зв’язки між сутностями

Сутність 1	Сутність 2	Зв’язок
Забіг	Результат	Один-до-багатьох
Порушення	Результат	Багато-до-багатьох
Власник	Кінь	Один-до-багатьох
Кінь	Результат	Один-до-багатьох
Іподром	Змагання	Один-до-багатьох
Змагання	Забіг	Один-до-багатьох
Жокей	Результат	Один-до-багатьох

На рисунку 2.9 представлено діаграму «сутність-зв’язок».

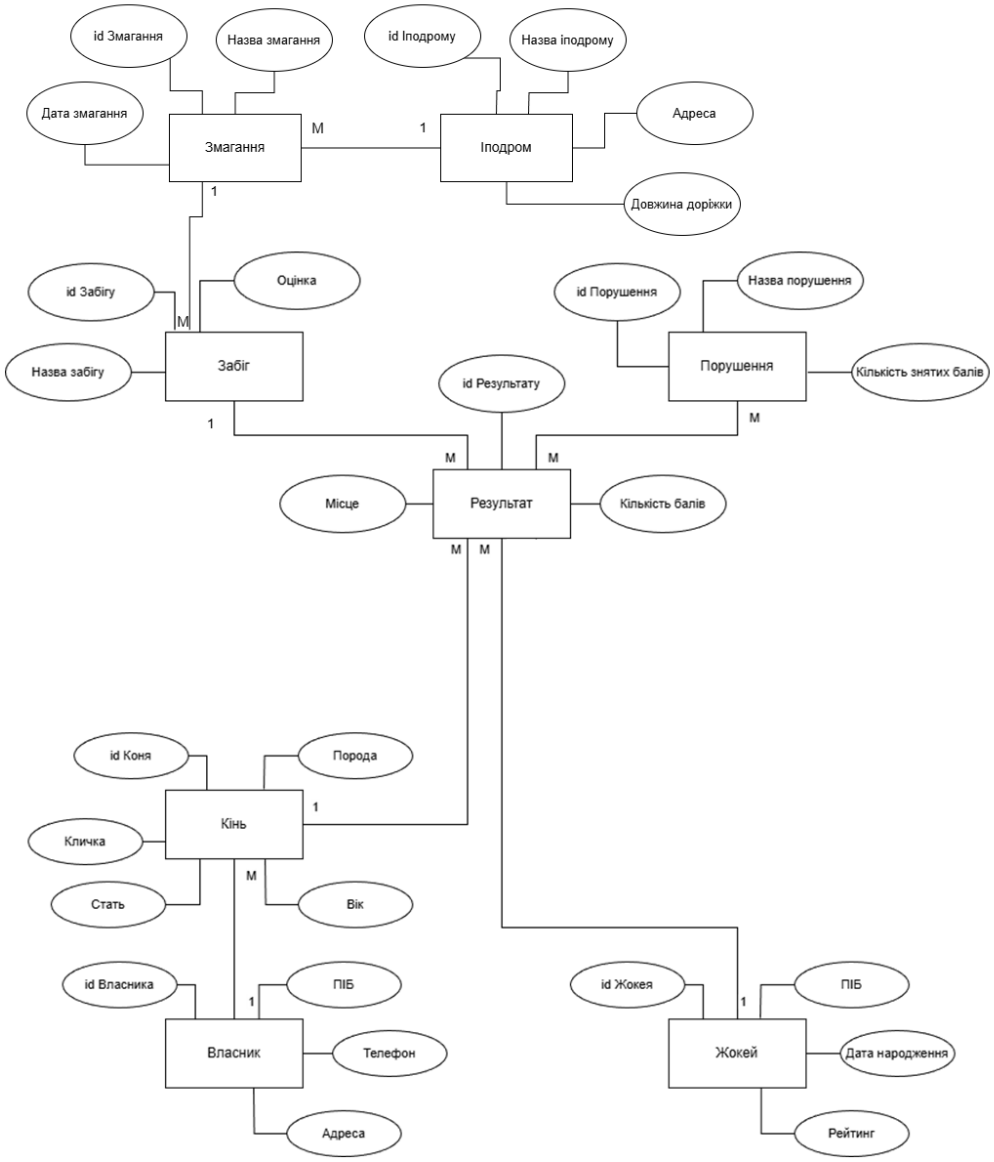


Рисунок 2.9 – Діаграма «сутність-зв’язок» вебзастосунку для автоматизації обробки результатів кінних скачок

Розробка проєкту користувацького інтерфейсу вебзастосунку для автоматизації обробки результатів кінних скачок

Важливим етапом проєктування програмного забезпечення є розробка його інтерфейсу. Відповідно до умов програмного забезпечення необхідно створити

простий та достатньо функціональний інтерфейс користувача. На рисунках 2.10 – 2.14 представлено ескізи основних форм та сторінок вебзастосунку.

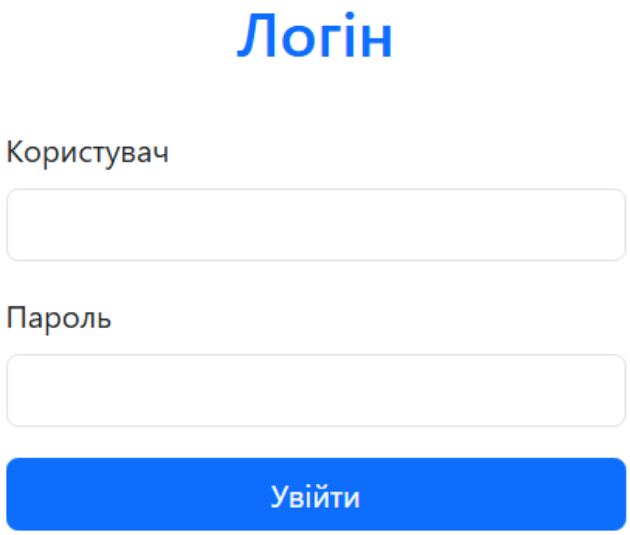


Рисунок 2.10 – Ескіз вікна авторизації вебзастосунку

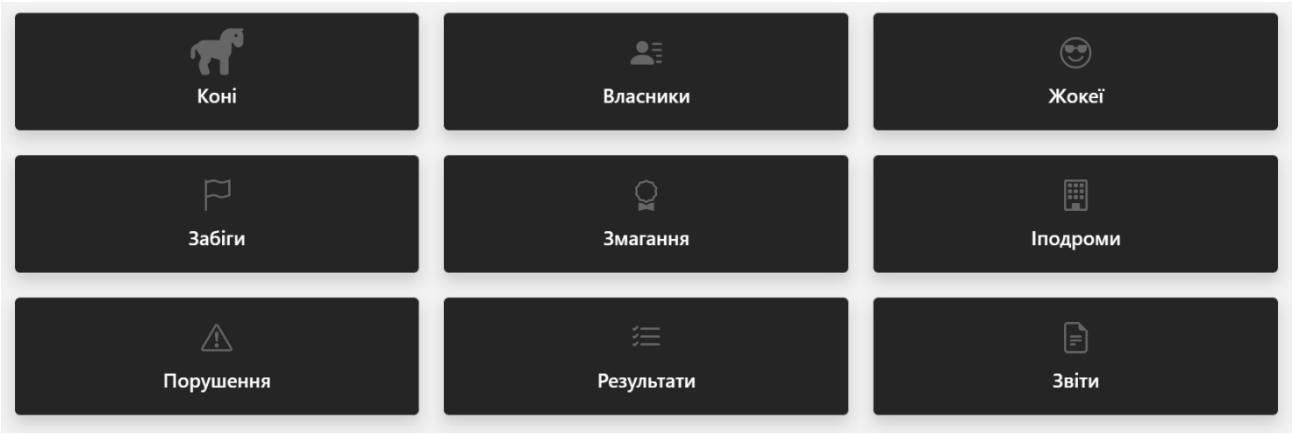


Рисунок 2.11 – Ескіз головної сторінки вебзастосунку

Рисунок 2.12 – Ескіз сторінки формування звітів

Рисунок 2.13 – Ескіз таблиці вебзастосунку

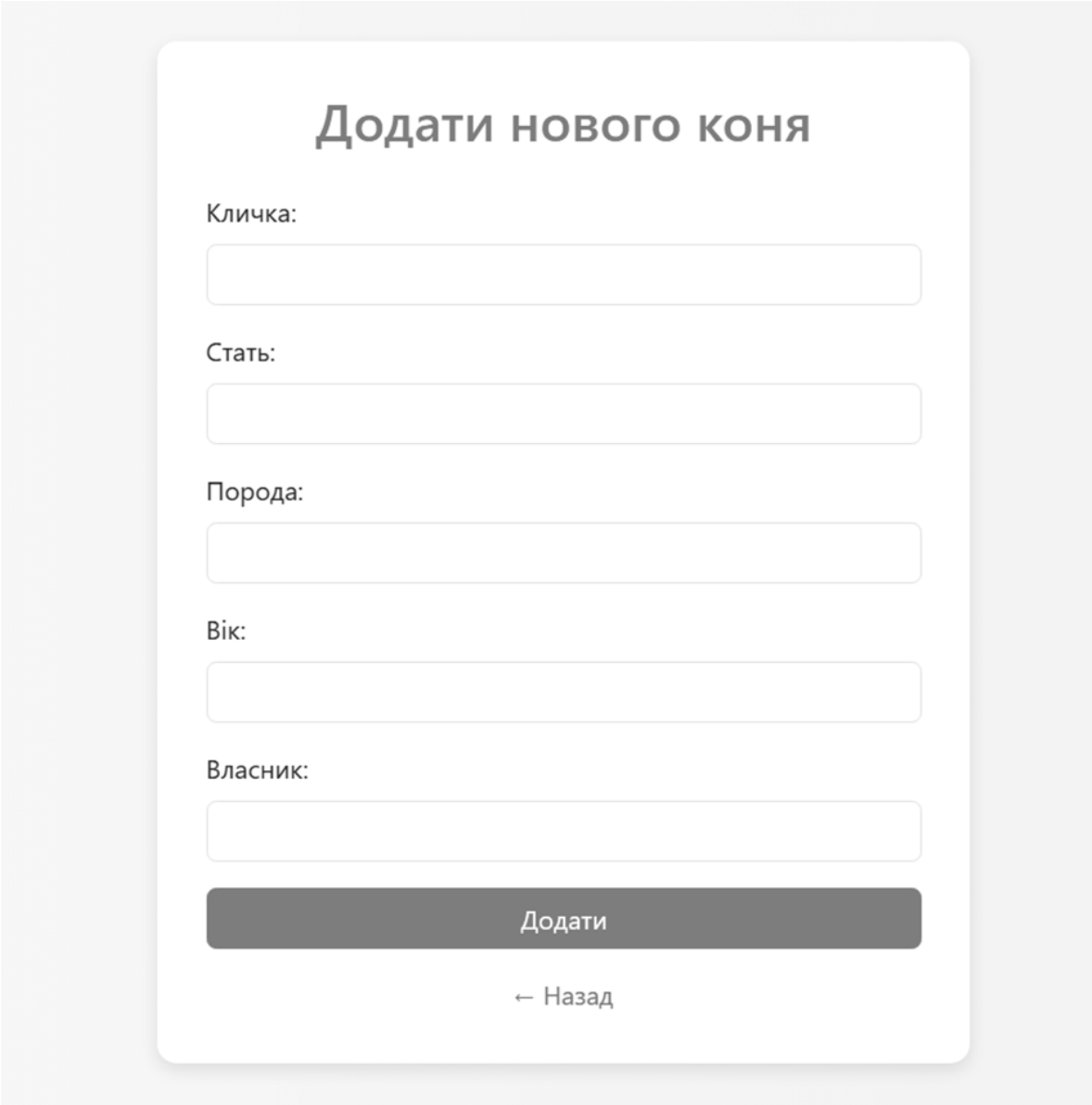


Рисунок 2.14 – Ескіз форми додавання нового коня

2.3.2 Технічний проєкт вебзастосунку для автоматизації обробки результатів кінних скачок

Ґрунтуючись на розробленому ескізному проєкті, проведемо розробку технічного проєкту вебзастосунку для автоматизації обробки результатів кінних скачок. Побудуємо статичну модель, представлену діаграмою класів, та динамічну модель, представлену діаграмою послідовності.

Побудова статичної моделі вебзастосунку для автоматизації обробки результатів кінних скачок

Статичну структуру програмного забезпечення можна зобразити за допомогою діаграми класів. Вона є структурною схемою, яка відображає склад системи або ПЗ, показуючи класи, їхні властивості (атрибути) і зв’язки між ними. Класи на діаграмі зображуються у вигляді прямокутників, де вказані атрибути та методи – відповідно до їхніх функцій та характеристик.

Взаємозв’язки між класами позначаються стрілками, що відображають напрямок зв’язку між класами. Такі діаграми допомагають краще зрозуміти будову системи та зв’язки між її елементами.

На рисунку 2.15 представлено діаграму класів вебзастосунку для автоматизації обробки результатів кінних скачок.

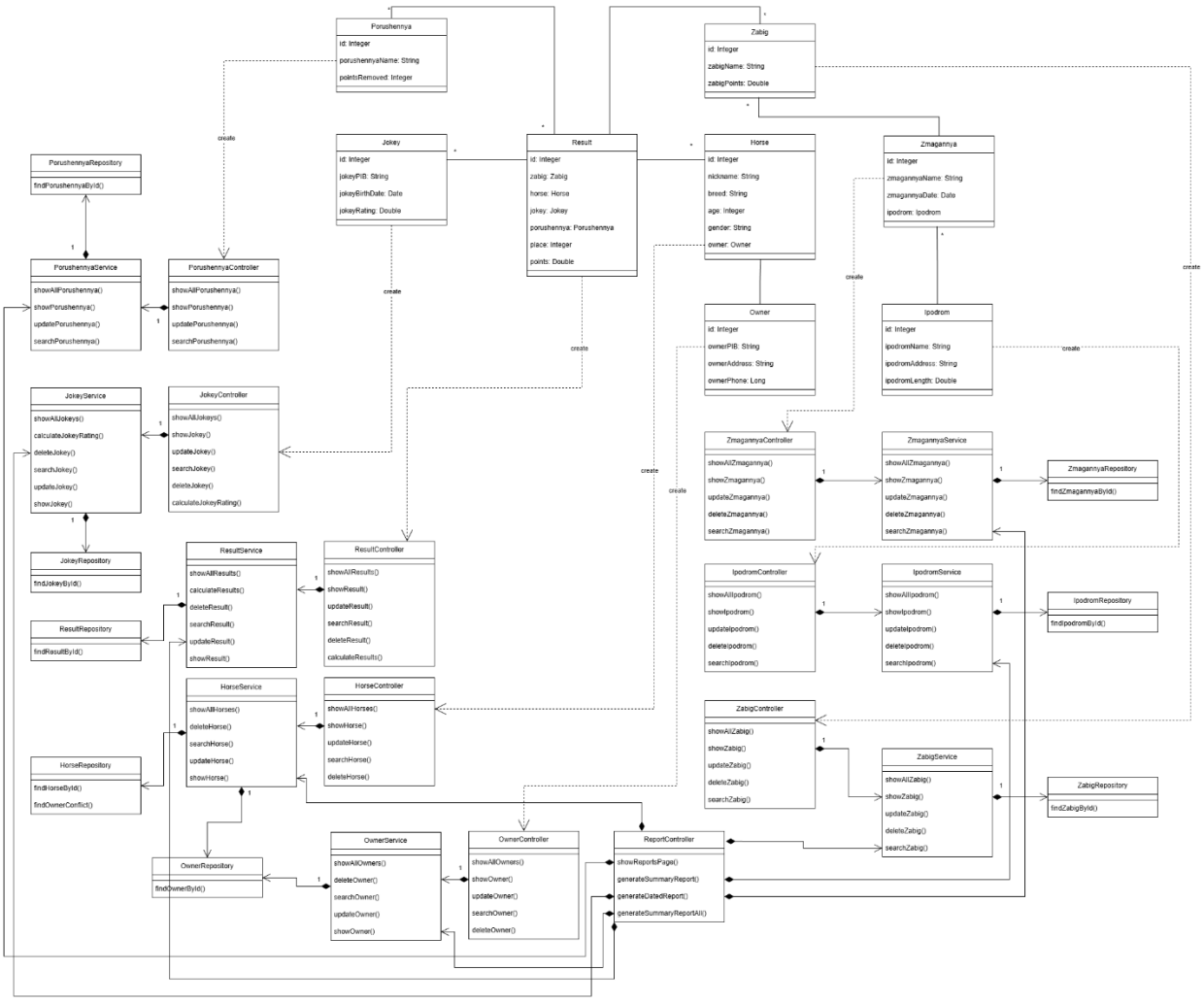


Рисунок 2.15 – Діаграма класів вебзастосунку для автоматизації обробки результатів кінних скачок

Специфікації класів вебзастосунку для автоматизації обробки результатів кінних скачок

Нижче наведено специфікації класів вебзастосунку для автоматизації обробки результатів кінних скачок, де зазначено їх назву, відповідальність, атрибути та методи.

Специфікація класу “Porushennya”

Назва: Porushennya.

Відповідальність: Клас представляє порушення, що стосуються певних результатів.

Атрибути:

– id: Integer – унікальний ідентифікатор порушення;

– porushennyaName: String – назва порушення;

– pointsRemoved: Integer – кількість знятих балів за порушення.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Jokey”

Назва: Jokey.

Відповідальність: Клас представляє жокея, який бере участь у змаганнях.

Атрибути:

– id: Integer – унікальний ідентифікатор жокея;

– jokeyPIB: String – повне ім’я жокея;

– jokeyPIB: Date – дата народження жокея;

– jokeyPIB: Double – рейтинг жокея.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Result”

Назва: Result.

Відповідальність: Клас представляє результати змагання.

Атрибути:

– id: Integer – унікальний ідентифікатор результату;

– zmagannya: Zmagannya – змагання, до якого належить результат;

– horse: Horse – кінь, що брав участь у результаті;

– dateOfEvent: Date – дата події;

– place: String – місце проведення події;

– jokey: Jokey – жокей, що брав участь у змаганні;

– porushennya: Porushennya – порушення, що виникли під час результату.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Zabig”

Назва: Zabig.

Відповідальність: Клас представляє конкретний забіг на змаганнях.

Атрибути:

– id: Integer – унікальний ідентифікатор забігу;

– zabigName: String – назва забігу;

– zabigPoints: Integer – кількість балів за забіг.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Horse”

Назва: Horse.

Відповідальність: Клас представляє інформацію про коня.

Атрибути:

– id: Integer – унікальний ідентифікатор коня;

– nickname: String – кличка коня;

– breed: String – порода коня;

– age: Integer – вік коня;

– gender: String – стать коня;

– owner: Owner – власник коня.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Zmagannya”

Назва: Zmagannya.

Відповідальність: Клас представляє змагання з кінного спорту.

Атрибути:

– id: Integer – унікальний ідентифікатор змагання;

– zmagannyaName: String – назва змагання;

– zmagannyaDate: Date – дата проведення змагання.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Owner”

Назва: Owner.

Відповідальність: Клас представляє власника коня.

Атрибути:

– id: Integer – унікальний ідентифікатор власника;

– ownerPIB: String – повне ім’я власника;

– ownerAddress: String – адреса власника;

– ownerPhone: Long – номер телефону власника.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “Ipodrom”

Назва: Ipodrom.

Відповідальність: Клас представляє інформацію про іподром.

Атрибути:

– id: Integer – унікальний ідентифікатор іподрому;

– ipodromName: String – назва іподрому;

– ipodromAddress: String – адреса іподрому;

– trackLength: Double – довжина бігової доріжки.

Методи:

Клас має методи для встановлення та отримання значень усіх атрибутів.

Специфікація класу “PorushennyaController”

Назва: PorushennyaController.

Відповідальність: Обробка запитів щодо порушень.

Методи:

– showAllPorushennya() – метод для отримання звіту всіх порушень;

– showPorushennya() – метод для отримання інформації про конкретне порушення;

– searchPorushennya() – метод для пошуку порушення;

– updatePorushennya() – редагування інформації про порушення.

Специфікація класу “JokeyController”

Назва: JokeyController.

Відповідальність: Обробка запитів щодо жокеїв.

Методи:

– showAllJokeys() – метод для отримання звіту всіх жокеїв;

– searchJokeys() – метод для пошуку жокея;

– showJokey() – метод для отримання інформації про конкретного жокея;

– calculateJokeyRating() – метод для прорахунку рейтингу жокея;

– updateJokey() – редагування інформації про жокея;

– deleteJokey() – видалення жокея.

Специфікація класу “ResultController”

Назва: ResultController.

Відповідальність: Обробка запитів щодо результатів.

Методи:

– showAllResults() – метод для отримання всіх результатів;

– showResult() – отримання конкретного результату;

– searchResults () – метод для пошуку результату;

– calculateResult() – підрахунок результату;

– updateResult() – редагування результату;

– deleteResult() – видалення результату.

Специфікація класу “HorseController”

Назва: HorseController.

Відповідальність: Обробка запитів щодо інформації про коней.

Методи:

– showAllHorses() – метод для отримання звіту всіх коней;

– showHorse() – метод для отримання інформації про конкретного коня;

– searchHorses() – метод для пошуку інформації про коня;

– updateHorse() – редагування інформації про коня;

– deleteHorse() – видалення коня.

– saveHorse() – збереження коня.

Специфікація класу “OwnerController”

Назва: OwnerController.

Відповідальність: Обробка запитів щодо власників коней.

Методи:

– showAllOwners() – метод для отримання звіту всіх власників;

– showOwner() – метод для отримання інформації про конкретного власника;

– searchOwners() – метод для пошуку інформації про власника;

– updateOwner() – редагування інформації про власника;

– deleteOwner() – видалення власника.

Специфікація класу “ZmagannyaController”

Назва: ZmagannyaController.

Відповідальність: Обробка запитів щодо змагань.

Методи:

– showAllZmagannya() – метод для отримання всіх змагань;

– showZmagannya() – отримання конкретного змагання;

– searchZmagannya() – метод для пошуку інформації про змагання;

– updateZmagannya() – редагування змагання;

– deleteZmagannya() – видалення змагання.

Специфікація класу “IpodromController”

Назва: IpodromController.

Відповідальність: Обробка запитів щодо іподромів.

Методи:

– showAllIpodroms() – метод для отримання всіх іподромів;

– showIpodrom() – отримання конкретного іподрому;

– searchIpodrom() – метод для пошуку інформації про іподром;

– updateIpodrom() – редагування інформації про іподром;

– deleteIpodrom() – видалення іподрому.

Специфікація класу “ZabigController”

Назва: ZabigController.

Відповідальність: Обробка запитів щодо забігів.

Методи:

– showAllZabigs() – метод для отримання всіх забігів;

– showZabig() – отримання конкретного забігу;

– searchZabig() – метод для пошуку інформації про забіг;

– updateZabig() – редагування забігу;

– deleteZabig() – видалення забігу.

Специфікація класу “ReportController”

Назва: ReportController.

Відповідальність: Обробка запитів щодо формування звітів.

Методи:

– showReportsPage() – метод для показу варіантів звітів;

– generateSummaryReport () – метод для формування звіту з інформацією про суму усіх учасників;

– generateDatedReport () – метод для формування звіту з інформацією про забіги за датою;

– generateSummaryReportAll() – метод для формування звіту з інформацією про усі забіги;

Специфікація класу “PorushennyaService”

Назва: PorushennyaService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із порушеннями.

Методи:

– showAllPorushennya() – метод для отримання звіту про всі порушення;

– showPorushennya() – метод для отримання інформації про конкретне порушення;

– searchPorushennya() – метод для пошуку порушення;

– updatePorushennya() – метод для редагування інформації про порушення.

Специфікація класу “JokeyService”

Назва: JokeyService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із жокеями.

Методи:

– showAllJokeys() – метод для отримання звіту про всіх жокеїв;

– searchJokeys() – метод для пошуку жокея;

– calculateJokeyRating() – метод для прорахунку рейтингу жокея;

– showJokey() – метод для отримання інформації про конкретного жокея;

– updateJokey() – метод для редагування інформації про жокея;

– deleteJokey() – метод для видалення жокея.

Специфікація класу “ResultService”

Назва: ResultService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із результатами.

Методи:

– showAllResults() – метод для отримання звіту про всі результати;

– showResult() – метод для отримання інформації про конкретний результат;

– searchResults () – метод для пошуку результату;

– calculateResult() – підрахунок результату;

– updateResult() – метод для редагування інформації про результат.

Специфікація класу “HorseService”

Назва: HorseService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із конями.

Методи:

– showAllHorses() – метод для отримання звіту про всіх коней;

– showHorse() – метод для отримання інформації про конкретного коня;

– updateHorse() – метод для редагування інформації про коня;

– deleteHorse() – метод для видалення коня;

– saveHorse() – метод для збереження інформації про коня;

– isOwnerPresent() – метод для перевірки наявності власника коня у базі даних.

Специфікація класу “OwnerService”

Назва: OwnerService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із власниками коней.

Методи:

– showAllOwners() – метод для отримання звіту про всіх власників;

– showOwner() – метод для отримання інформації про конкретного власника;

– searchOwners() – метод для пошуку інформації про власника;

– updateOwner() – метод для редагування інформації про власника;

– deleteOwner() – метод для видалення власника.

Специфікація класу “ZabigService”

Назва: ZabigService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із забігами.

Методи:

– showAllZabig() – метод для отримання звіту про всі забіги;

– showZabig() – метод для отримання інформації про конкретний забіг;

– updateZabig() – метод для редагування інформації про забіг;

– searchZabig() – метод для пошуку інформації про забіг;

– deleteZabig() – метод для видалення забігу.

Специфікація класу “ZmagannyaService”

Назва: ZmagannyaService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із змаганнями.

Методи:

– showAllZmagannya() – метод для отримання звіту про всі змагання;

– showZmagannya() – метод для отримання інформації про конкретне змагання;

– searchZmagannya() – метод для пошуку інформації про змагання;

– updateZmagannya() – метод для редагування інформації про змагання;

– deleteZmagannya() – метод для видалення змагання.

Специфікація класу “IpodromService”

Назва: IpodromService.

Відповідальність: Об’єкт сервісу для виконання операцій, пов’язаних із іподромами.

Методи:

– showAllIpodroms() – метод для отримання звіту про всі іподроми;

– showIpodrom() – метод для отримання інформації про конкретний іподром;

– searchIpodrom() – метод для пошуку інформації про іподром;

– updateIpodrom() – метод для редагування інформації про іподром;

– deleteIpodrom() – метод для видалення іподрому.

Специфікація класу “PorushennyaRepository”

Назва: PorushennyaRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про порушення.

Методи:

– findPorushennyaById(id: Integer): Porushennya – знаходить порушення за його унікальним ідентифікатором.

Специфікація класу “JokeyRepository”

Назва: JokeyRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про жокеїв.

Методи:

– findJokeyById(id: Integer): Jokey – знаходить жокея за його унікальним ідентифікатором.

Специфікація класу “ResultRepository”

Назва: ResultRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про результати.

Методи:

– findResultById(id: Integer): Result – знаходить результат за його унікальним ідентифікатором.

Специфікація класу “HorseRepository”

Назва: HorseRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про коней.

Методи:

– findHorseById(id: Integer): Horse – знаходить коня за його унікальним ідентифікатором.

– findOwnerConflict(horseid, ownerid) – знаходить конфлікт у випадку неіснуючого або неправильного введення інформації щодо власника коня.

Специфікація класу “OwnerRepository”

Назва: OwnerRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про власників коней.

Методи:

– findOwnerById(id: Integer): Owner – знаходить власника за його унікальним ідентифікатором.

Специфікація класу “ZmagannyaRepository”

Назва: ZmagannyaRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про змагання.

Методи:

– findZmagannyaById(id: Integer): Zmagannya – знаходить змагання за його унікальним ідентифікатором.

Специфікація класу “IpodromRepository”

Назва: IpodromRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про іподроми.

Методи:

– findIpodromById(id: Integer): Ipodrom – знаходить іподром за його унікальним ідентифікатором.

Специфікація класу “ZabigRepository”

Назва: ZabigRepository.

Відповідальність: Інтерфейс для взаємодії з базою даних щодо інформації про забіги.

Методи:

– findZabigById(id: Integer): Zabig – знаходить забіг за його унікальним ідентифікатором.

Розробка динамічної моделі вебзастосунку для автоматизації обробки результатів кінних скачок

Динамічну модель програмного забезпечення можна зобразити у вигляді діаграми послідовності. Діаграми послідовності (sequence diagrams) є одним із різновидів UML-діаграм (Unified Modeling Language), призначених для моделювання динамічної поведінки системи. Вони демонструють, як об’єкти взаємодіють між собою у певному порядку. Основні складові таких діаграм – це актори, об’єкти, лінії життя, повідомлення та інтервали активності.

На рисунках 2.16 та 2.17 наведено приклади діаграм послідовності для варіантів «Обробити інформацію про коня» та «Авторизація».

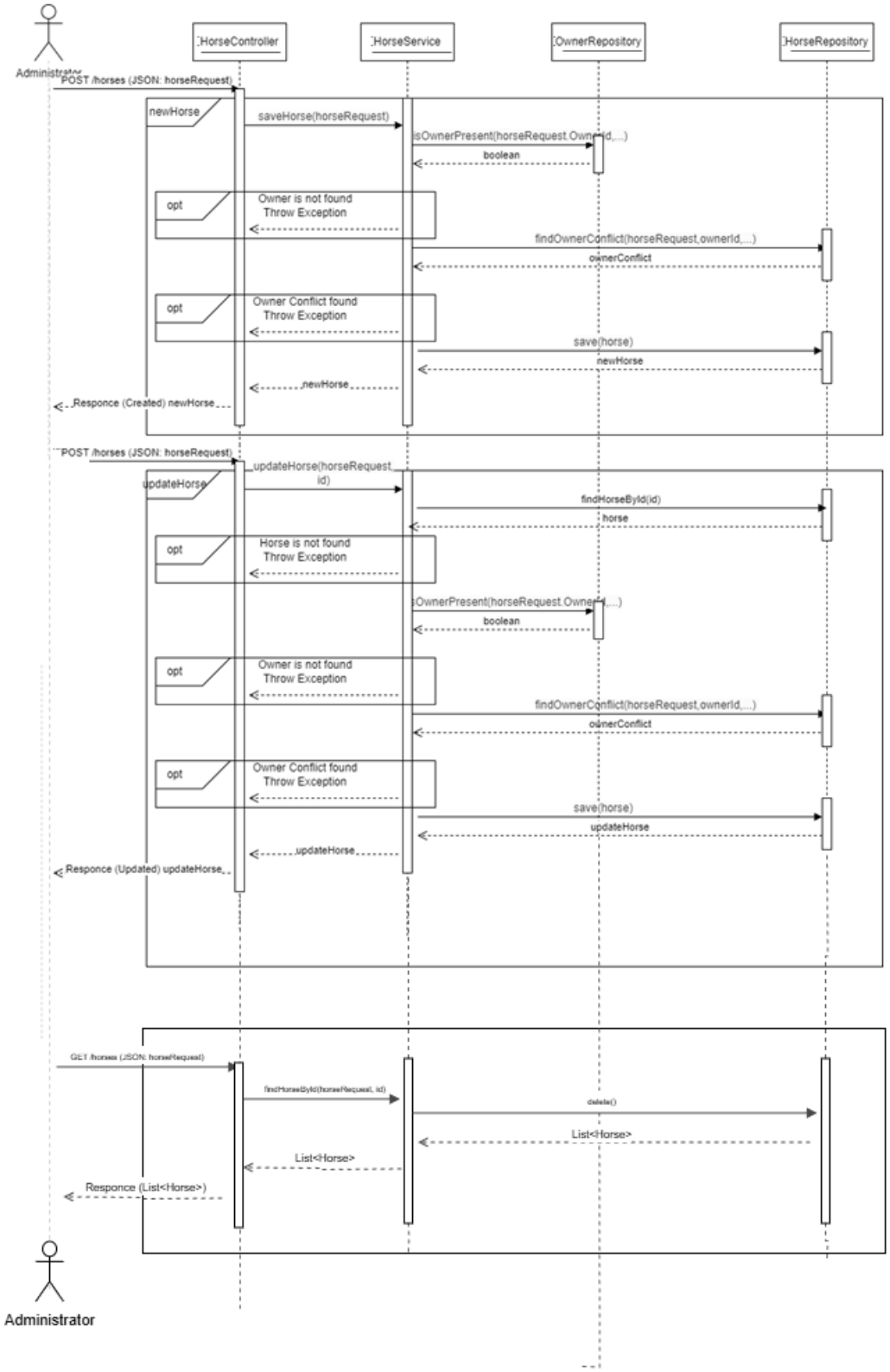


Рисунок 2.16 – Діаграма послідовності варіанту використання «Обробка інформації про коня»

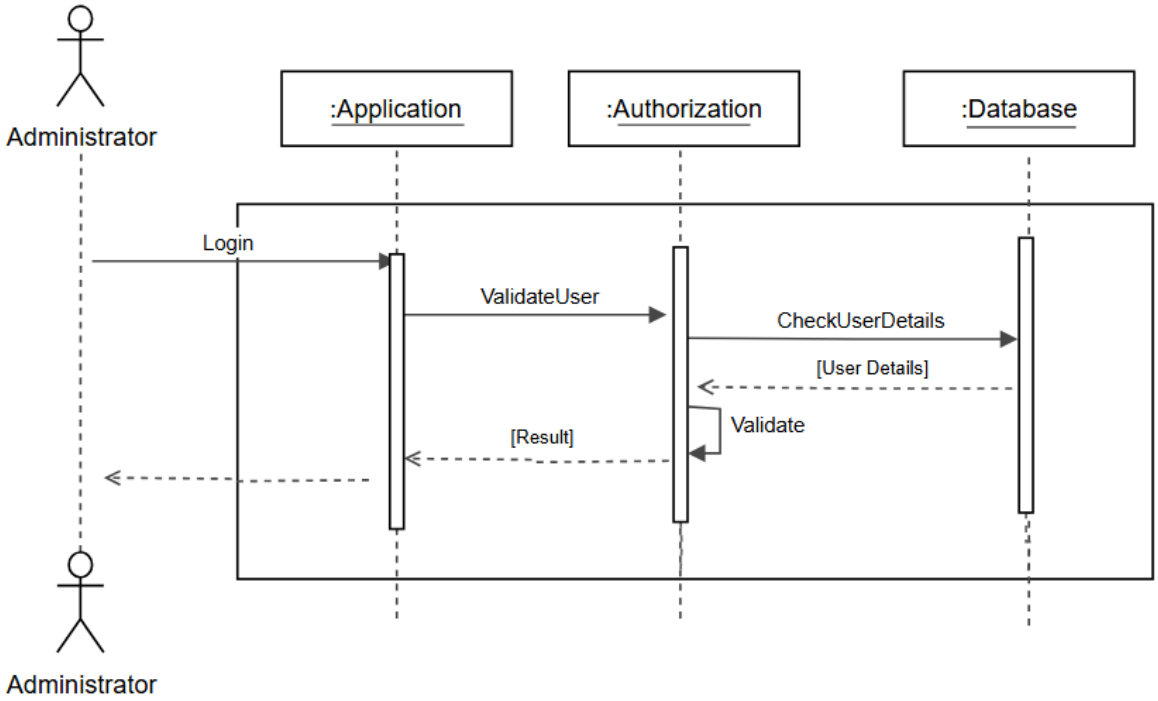


Рисунок 2.17 – Діаграма послідовності варіанту використання «Авторизація»

Розробка логічної моделі даних вебзастосунку для автоматизації обробки результатів кінних скачок

Під час створення логічної моделі реляційної бази даних ключову роль відіграє процес визначення відношень, що базується на понятті функціональної залежності (ФЗ). Завдяки встановленню ФЗ та вибору найдоцільніших з точки зору компактності і ефективності представлень, вдається сформувати оптимальну структуру бази даних. Це дозволяє забезпечити надійне зберігання та обробку інформації, використовуючи методи еквівалентного перетворення схем відношень реляційних баз даних.

Такий підхід до моделювання називається нормалізацією відношень у межах інформаційної моделі предметної області. Основна ціль логічного проєктування полягає у формуванні концептуального уявлення про базу даних з

урахуванням обраної моделі зберігання інформації (наприклад, реляційної, ієрархічної або мережевої).

Розроблену логічну модель представлено на рисунку 2.18

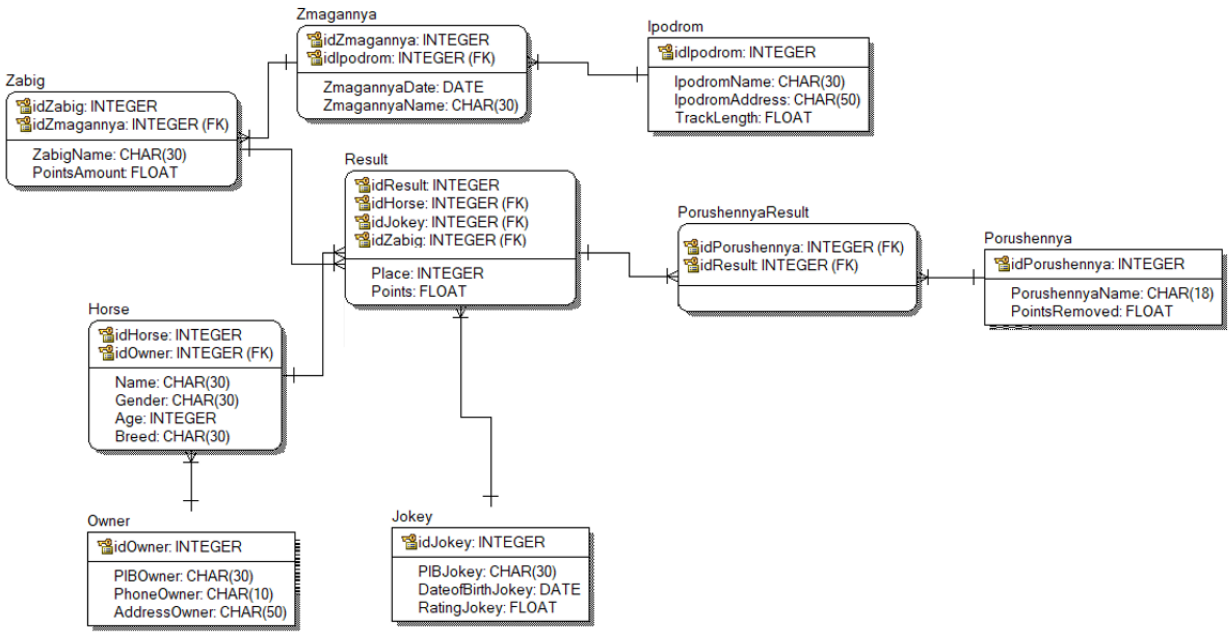


Рисунок 2.18 – Логічна модель вебзастосунку для автоматизації обробки результатів кінних скачок

2.3.3 Робочий проєкт вебзастосунку для автоматизації обробки результатів кінних скачок

Обгрунтування вибору засобів розробки вебзастосунку для автоматизації обробки результатів кінних скачок

Для створення вебзастосунку для автоматизації обробки результатів кінних скачок були обрані такі засоби розробки:

– IntelliJ IDEA: Для написання коду вебзастосунку використовувався редактор коду IntelliJ IDEA від компанії JetBrains. Цей редактор є одним із найзручніших середовищ для розробки на Java. Використано безкоштовну версію Community Edition, яка містить всі необхідні функції для розробки серверних додатків. Програма підтримується на платформах Windows, Linux та macOS і має потужні засоби автодоповнення, рефакторингу та інтеграції з системами контролю версій.

– Java: Для серверної частини було обрано мову Java, яка є потужною, об'єктно-орієнтованою і добре підходить для розробки надійних вебзастосунків.

– Spring Boot: В якості фреймворку використано Spring Boot, що спрощує створення вебзастосунків за архітектурною моделлю MVC (Model-View-Controller). Spring Boot автоматизує конфігурації, дозволяє легко працювати з формами, валідацією, базами даних, а також забезпечує гнучку маршрутизацію, інтеграцію з шаблонами (Thymeleaf) та обробку помилок.

– HTML: Високорівнева мова програмування, що застосовується для створення інтерактивних вебсторінок і вебдодатків. Вона дає можливість динамічно взаємодіяти з HTML і CSS, що дозволяє змінювати контент сторінки без перезавантаження. Завдяки інтеграції з шаблонізатором Thymeleaf HTML дозволяє також відображати помилки валідації, умовно показувати блоки та керувати значеннями форм без використання JavaScript.

– CSS: CSS використовується для стилізації сторінок: визначення кольорів, шрифтів, позиціонування елементів, адаптивного дизайну тощо.

Для покращеної структури стилів може використовуватись SCSS, що є препроцесором до CSS і дозволяє застосовувати змінні, вкладені селектори та функції, які спрощують підтримку великих проєктів.

– Thymeleaf: Як шаблонізатор для формування HTML-сторінок на сервері обрано Thymeleaf. Він інтегрується зі Spring Boot "з коробки", дозволяє ефективно працювати з формами, локалізацією та динамічним відображенням даних (наприклад, списків ролей, повідомлень про помилки тощо).

– Bootstrap: Для покращення візуального вигляду форм та таблиць можна використовувати Bootstrap або бібліотеку jQuery для обробки подій, анімацій та AJAX-запитів.

– MySQL: Для зберігання зареєстрованих користувачів, ролей та іншої інформації обрана реляційна база даних MySQL, яка є стабільною, масштабованою та популярною у веброзробці.

– phpMyAdmin: Для адміністрування бази використовується phpMyAdmin – вебінтерфейс, що дозволяє зручно створювати таблиці, переглядати записи, виконувати запити та резервне копіювання даних без необхідності використання CLI.

Реалізація основних класів вебзастосунку для автоматизації обробки результатів кінних скачок

Вебзастосунок написаний мовою Java із застосуванням Spring Boot та СКБД MySQL. Для управління залежностями забезпечення архітектури плагінів у проєкту, структурованності, використано інструмент управління проєктами та автоматизованими засобами збірки Maven. Нижче наведено код для запуску програми та клас роботи з таблицею Horse бази даних.

Повний код програми представлено в додатку Д.

HorseRacing.java

```
package org.example.horseracing;

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

@SpringBootApplication
public class HorseRacing {

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

}
```

Horse.java

```
package org.example.labb.entity;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "horse", schema = "labs_java")
public class Horse {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idhorse", nullable = false)
    private Integer id;

    @Column(name = "nickname", length = 45)
    private String nickname;

    @Column(name = "gender", length = 45)
    private String gender;

    @Column(name = "breed", length = 45)
    private String breed;
```

```
    @Column(name = "age")
    private Integer age;

    @ManyToOne
    @JoinColumn(name = "owner_idowner", referencedColumnName = "idowner")
    private Owner owner;

    @OneToMany(mappedBy = "horse")
    private List<Result> results = new ArrayList<>();

    public Owner getOwner() {
        return owner;
}

    public void setOwner(Owner owner) {
        this.owner = owner;
    }
    public List<Result> get Result() {
        return results;
    }

    public void set Result(List<Result> results) {
        this.results = results;
    }
}
```

У наведеному коді використано JPA для відображення класу Horse із таблиці бази даних horse та анотації, що описують структуру таблиці і зв’язки між сутностями. Також для позначення головного класу, з якого починається запуск вебзастосунку, було використано анотацію із Spring Boot.

Тестування та випробовування вебзастосунку для автоматизації обробки результатів кінних скачок

Тестування програмного забезпечення – це процес, коли проводиться

дослідження та випробування програмного продукту, щоб виявити розбіжності між реальною поведінкою програми та її очікуваними результатами, базуючись на заздалегідь визначеному наборі тестових випадків. Цей процес включає в себе перевірку функціональності, продуктивності програми та інших аспектів заради забезпечення правильності та відповідності вимогам та гарантування стабільної роботи у реальних умовах використання програмного забезпечення.

Далі наведено тестування наступних прецедентів: «Обробка інформації про коней», «Обробка інформації про порушення» та «Обробка інформації про іподроми».

Під час тестування використовувалося тестування «чорної скриньки» за такими методами:

– Еквівалентне розбиття – для кожного тестового випадку визначалися класи еквівалентності для вхідних даних, поділяючи їх на валідні та невалідні.

– Аналіз граничних значень – деякі з тестових випадків включали перевірку граничних значень для числових і текстових параметрів, а також значень на межах допустимих діапазонів.

Тестування прецеденту «Обробка інформації про коней» методами еквівалентного розбиття та аналізу граничних значень.

Додавання інформації

1) Перевірити, що форма додавання інформації (наприклад, про коня) відображається коректно та доступна адміністратору.

Ідентифікатор: TC-ADMIN-ADD-01

Предмети тестування: Форма додавання коня

Специфікація вхідних даних: Відкриття сторінки додавання коня у веббраузері.

Специфікація результатів: Форма коректно відображається, усі необхідні поля присутні: Кличка, Порода, Стать, Вік, Власник.

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як адміністратор.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити вебзастосунок у браузері за адресою http://races:8080.
- Увійти як адміністратор.
- Перейти до розділу "Коні" → "Додати коня".
- Перевірити, чи відображаються поля: Кличка, Порода, Стать, Вік, Власник.

Методи тестування:

Чорна скринька: Перевірка функціональності форми без знання її внутрішньої структури.

Класи еквівалентності:

- Доступ до форми для авторизованого адміністратора (валідний клас).
- Доступ до форми для неавторизованого користувача (невалідний клас, очікування: відмова в доступі).

Тести:

- Перевірка відображення всіх полів форми для авторизованого адміністратора.
- Спроба доступу до форми без авторизації (очікування: помилка доступу).

2) Перевірити, що при введенні коректних даних інформація про коня успішно додається, а користувач отримує повідомлення про успіх.

Ідентифікатор: TC-ADMIN-ADD-02

Предмети тестування: Форма додавання коня

Специфікація вхідних даних:

- Кличка: Буря
- Порода: Арабська
- Стать: Жеребець
- Вік: 5
- Власник: Олександр Михайлов

Специфікація результатів: Коня успішно додано до бази даних, з’являється повідомлення "Кінь успішно доданий", кінь відображається у списку коней.

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як адміністратор.

Залежність з іншими тестами: TC-ADMIN-ADD-01

Дії користувача:

- Відкрити вебзастосунок у браузері за адресою http://races:8080.
- Увійти як адміністратор.
- Перейти до розділу "Коні" → "Додати коня".
- Заповнити поля:
  - Кличка: Буря
  - Порода: Арабська
  - Стать: Жеребець
  - Вік: 5
  - Власник: Олександр Михайлов
- Натиснути кнопку "Зберегти".
- Перевірити, чи з’являється повідомлення "Кінь успішно доданий".
- Перейти до списку коней і перевірити, чи кінь "Буря" відображається.

Методи тестування:

Чорна скринька: Перевірка збереження даних без знання внутрішньої логіки.

Класи еквівалентності:

- Валідні дані: Вік (1–30), Кличка (1–50 символів), Порода (вибрана зі списку), Стать (Жеребець/Кобила), Власник (1–100 символів).
- Невалідні дані: Порожні поля, перевищення довжини рядків, некоректні символи.

Аналіз граничних умов:

- Вік: 1 (мін.), 30 (макс.), 0 (невалідний), 31 (невалідний).
- Довжина Клички: 1 символ (мін.), 50 символів (макс.), 51 символ (невалідний).

Тести:

- Введення валідних даних (як у специфікації).
- Введення граничних значень для віку (1, 30).
- Введення невалідних даних: вік 0, вік 31, порожня кличка.

3) Перевірити, що при введенні некоректних даних (наприклад, від’ємний вік коня) валідація спрацьовує, і відображається повідомлення про помилку.

- Ідентифікатор: TC-ADMIN-ADD-03
- Предмети тестування: Форма додавання коня
- Специфікація вхідних даних:
  - Кличка: Буря
  - Порода: Арабська
  - Стать: Жеребець
  - Вік: -1
  - Власник: Олександр Михайлов

Специфікація результатів: Дані не зберігаються, відображається повідомлення "Помилка: Вік має бути додатним числом".

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як адміністратор.

Залежність з іншими тестами: TC-ADMIN-ADD-01

Дії користувача:

- Відкрити вебзастосунок у браузері за адресою http://races:8080.
- Увійти як адміністратор.
- Перейти до розділу "Коні" → "Додати коня".
- Заповнити поля:
  - Кличка: Буря
  - Порода: Арабська
  - Стать: Жеребець
  - Вік: -1
  - Власник: Олександр Михайлов
- Натиснути кнопку "Зберегти".
- Перевірити, чи з’являється повідомлення "Помилка: Вік має бути додатним числом".

Методи тестування:

Чорна скринька: Перевірка валідації введення.

Класи еквівалентності:

- Валідний вік: 1–30.
- Невалідний вік: <1, >30, нечислові значення.

Аналіз граничних умов:

- Вік: 0, -1, 31, нечислові значення (наприклад, "abc").

Тести:

- Введення від’ємного віку (-1).
- Введення нульового віку (0).

- Введення нечислового значення ("abc").
- Введення віку, що перевищує максимум (31).

Редагування інформації

4) Перевірити, що при спробі редагування вдруге протягом 24 годин відображається повідомлення про обмеження.

Ідентифікатор: TC-ADMIN-EDIT-01

Предмети тестування: Форма редагування коня

Специфікація вхідних даних:

- ID коня: 1 (раніше відредагований менше ніж 24 години тому).
- Нові дані: Кличка: Гром

Специфікація результатів: Зміни не зберігаються, відображається повідомлення "Зміни дозволені лише раз на 24 години".

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як адміністратор. У базі даних є запис про коня з ID 1, відредагований недавно.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити вебзастосунок у браузері за адресою http://races:8080.
- Увійти як адміністратор.
- Перейти до розділу "Коні" → вибрати коня з ID 1 → "Редагувати".
- Змінити поле Кличка на "Гром".
- Натиснути кнопку "Зберегти".
- Перевірити, чи з’являється повідомлення "Зміни дозволені лише раз на 24 години".

Методи тестування:

Чорна скринька: Перевірка обмеження редагування.

Класи еквівалентності:

– Редагування до 24 годин після попередньої зміни (невалідний клас).
– Редагування після 24 годин (валідний клас).

Аналіз граничних умов:

– Час редагування: 23:59 год (невалідний), 24:00 год (валідний).

Тести:

– Спроба редагування через 23:59 год після попередньої зміни.
– Спроба редагування через 24:00 год після попередньої зміни.

Тестування прецеденту «Обробка інформації про порушення» методами еквівалентного розбиття та аналізу граничних значень.

Додавання інформації про порушення

1) Перевірити, що форма додавання порушення доступна для активного забігу.

– Ідентифікатор: TC-JUDGE-VIOLATION-01
– Предмети тестування: Форма додавання порушення

Специфікація вхідних даних: Відкриття сторінки додавання порушення для забігу з ID 201.

Специфікація результатів: Форма коректно відображається, містить поля: ID забігу, Опис порушення, Пов’язаний кінь/жокей.

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як суддя. У базі даних є активний забіг з ID 201.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– Відкрити вебзастосунок у браузері за адресою http://races:8080.
– Увійти як суддя.

- Перейти до розділу "Забіги" → вибрати забіг з ID 201 → "Додати порушення".
- Перевірити, чи відображаються поля: ID забігу, Опис порушення, Пов’язаний кінь/жокей.

Методи тестування:

Чорна скринька: Перевірка доступності форми.

Класи еквівалентності:

- Активний забіг (валідний клас).
- Неактивний/завершений забіг (невалідний клас).

Тести:

- Доступ до форми для активного забігу.
- Спроба доступу до форми для завершеного забігу (очікування: помилка).

2) Перевірити, що при введенні порожнього опису порушення валідація спрацьовує.

Ідентифікатор: TC-JUDGE-VIOLATION-02

Предмети тестування: Форма додавання порушення

Специфікація вхідних даних:

- ID забігу: 201
- Опис порушення: [порожнє]
- Пов’язаний жокей: Петро Сидоренко

Специфікація результатів: Порушення не додається, відображається повідомлення "Помилка: Опис порушення не може бути порожнім".

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як суддя. У базі даних є активний забіг з ID 201.

Залежність з іншими тестами: TC-JUDGE-VIOLATION-01

Дії користувача:

- Відкрити вебзастосунок у браузері за адресою http://races:8080.
- Увійти як суддя.
- Перейти до розділу "Забіги" → вибрати забіг з ID 201 → "Додати порушення".
- Заповнити поля:
  - ID забігу: 201
  - Опис порушення: [залишити порожнім]
  - Пов’язаний жокей: Петро Сидоренко
- Натиснути кнопку "Зберегти".
- Перевірити, чи з’являється повідомлення "Помилка: Опис порушення не може бути порожнім".

Методи тестування:

Чорна скринька: Перевірка валідації.

Класи еквівалентності:

- Валідний опис: 1–500 символів.
- Невалідний опис: порожній, перевищення 500 символів.

Аналіз граничних умов:

- Довжина опису: 0 символів (невалідний), 1 символ (валідний), 500 символів (валідний), 501 символ (невалідний).

Тести:

- Введення порожнього опису.
- Введення опису з 500 символів.
- Введення опису з 501 символу.

Тестування прецеденту «Обробка інформації про іподроми» методом еквівалентного розбиття.

Перегляд інформації

1) Перевірити, що при застосуванні фільтрів за назвою іподрому відображаються лише відповідні об’єкти.

Ідентифікатор: TC-JUDGE-VIEW-01

Предмети тестування: Функція перегляду іподромів

Специфікація вхідних даних:

– Фільтр: Назва іподрому = "Київський іподром"

Специфікація результатів: Відображається список іподромів, що містить лише "Київський іподром" із даними: назва, адреса, довжина доріжки.

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як суддя. У базі даних є іподром "Київський іподром".

Залежність з іншими тестами: Відсутні.

Дії користувача:

– Відкрити вебзастосунок у браузері за адресою http://races:8080.
– Увійти як суддя.
– Перейти до розділу "Іподроми".
– У полі фільтра ввести "Київський іподром".
– Натиснути кнопку "Застосувати фільтр".
– Перевірити, чи відображається лише "Київський іподром" із коректними даними.

Методи тестування:

Чорна скринька: Перевірка фільтрації.

Класи еквівалентності:

– Існуюча назва іподрому (валідний клас).
– Неіснуюча назва іподрому (невалідний клас).

Тести:

– Фільтрація за існуючою назвою.

– Фільтрація за неіснуючою назвою (очікування: порожній список).

2) Перевірити, що при відсутності результатів за фільтром відображається повідомлення.

Ідентифікатор: TC-JUDGE-VIEW-02

Предмети тестування: Функція перегляду іподромів

Специфікація вхідних даних:

– Фільтр: Назва іподрому = "Невідомий іподром"

Специфікація результатів: Список іподромів порожній, відображається повідомлення "Результатів не знайдено".

Тестове оточення: Веббраузер (Chrome, версія 120 або вище), локальний сервер Java, підключення до MySQL.

Спеціальні процедурні вимоги: Користувач авторизований як суддя.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– Відкрити вебзастосунок у браузері за адресою http://races:8080.
– Увійти як суддя.
– Перейти до розділу "Іподроми".
– У полі фільтра ввести "Невідомий іподром".
– Натиснути кнопку "Застосувати фільтр".
– Перевірити, чи відображається повідомлення "Результатів не знайдено".

Методи тестування:

Чорна скринька: Перевірка поведінки при відсутності даних.

Класи еквівалентності:

– Неіснуюча назва іподрому (невалідний клас).

Тести:

– Фільтрація за неіснуючою назвою.

Випробування вебзастосунку для автоматизації обробки результатів кінних скачок

Випробування вебзастосунку проводиться відповідно до програми та методики випробувань, що представлена у додатку В.

Під час випробування вебзастосунку було перевірено функціонал: додавання сутності Кінь до таблиці коней, редагування сутності Кінь у таблиці коней, видалення сутності Кінь у таблиці коней, видалення власника, що має коней, перегляд інформації про власників, пошук коня за його інформацією.

1) Додавання сутності Кінь до таблиці коней

Після авторизації користувача необхідно обрати таблицю Коней, що представлена на головній сторінці для переходу до таблиці коней. Після цього необхідно натиснути на кнопку «Додати нового коня», що розміщена під таблицею.

Після відкриття форми додавання коня вводимо правильні дані про коня, такі як:

Кличка: Солас;

Вік: 5;

Порода: Кавказька;

Стать: Самка;

Власник: 1 – обираємо із випадаючого списку необхідного власника.

Натискаємо «Додати». Форма перенаправляє користувача на сторінку із таблицею «Коні», де можна побачити доданого коня. Випробування пройшло успішно.

2) Редагування сутності Кінь у таблиці коней

Після авторизації користувача обираємо таблицю Коні та переходимо на неї, натискаючи на відповідну кнопку. Після цього наводимося на необхідного коня та натискаємо на кнопку «Редагувати», що присутня поряд із обраним

конем. Обираємо для випробування коня, що був доданий у випробуванні 1.

Після відкриття форми редагування змінюємо дані про коня. У цьому випробуванні змінимо кличку коня та його власника.

Змінюємо відповідні дані:

Кличка: Перчинка;

Власник: 4 – обираємо із випадаючого списку необхідного власника.

Натискаємо «Редагувати». Форма оновлює інформацію та перенаправляє до таблиці коней, де можна побачити, що ім’я коня та його власник змінився. Випробування успішно пройдено.

3) Видалення сутності Кінь у таблиці коней

Переходимо до таблиці «Коні», натискаючи на відповідну кнопку. У таблиці бачимо записи із конями та наводимося на коня, якого необхідно видалити.

Для цього випробування було додано коня із ID 8 із такими даними:

Кличка: Видалений

Вік: 7;

Порода: Мустанг;

Стать: Самка;

Власник: 4 – обираємо із випадаючого списку необхідного власника.

Наводимо на кнопку «Видалити», що розміщена у стовпчику «Дії» навпроти необхідного коня.

Натискаємо на кнопку «Видалити» та підтверджуємо видалення у вікні підтвердження.

Після підтвердження кінь видаляється із бази даних та сторінка перезавантажується із оновленим списком. Випробування пройшло успішно.

4) Видалення власника, що має коней

Переходимо до таблиці «Власники», натискаючи на відповідну кнопку. У таблиці бачимо записи із власниками та наводимося на власника, якого необхідно

видалити.

Під час цього випробування видалимо власника із ID 4, що був задіяний у випробуваннях раніше.

Наведемо курсор на власника та натиснемо на кнопку «Видалити». Система видає повідомлення, що у випадку видалення власника, усі його коні також будуть видалені. Погоджуємося із повідомленням та система видаляє власника, перевантажуючи таблицю власників із оновленими даними.

Натискаємо кнопку «Назад» та переходимо до таблиці «Коні», перевіряємо, чи було видалено відповідних коней.

Коні власника 4 були видалені, тому можна зробити висновок, що випробування було успішно пройдено.

5) Перегляд інформації про власників

Після авторизації користувача необхідно обрати таблицю Власників, що представлена на головній сторінці для переходу до таблиці власників.

На сторінці ми можемо побачити таблицю із власниками, кнопку їх додавання, окремі кнопки редагування та видалення власників. Над таблицею присутні поля, що дозволяють використовувати пошук по присутнім власникам за їх параметрами.

Необхідно перевірити, що інформація власників береться та передається із бази даних коректно.

Відкриваємо базу даних MySQL та бачимо наявні записи власників. У таблиці присутні власники із ID 1, 2, 3 та 5, кожен з яких має правильно внесені дані. Переходимо до вебзастосунку та перевіряємо дані на відповідність.

Усі дані власників було перевірено на правильність і відповідність. Отже, випробування було успішно пройдено.

6) Пошук коня за його інформацією

Після авторизації користувача необхідно обрати таблицю Коней, що представлена на головній сторінці для переходу до таблиці коней. На сторінці ми

можемо побачити таблицю із конями, кнопку їх додавання, окремі кнопки редагування та видалення власників. Над таблицею присутні поля, що дозволяють використовувати пошук по присутнім власникам за їх параметрами: кличка, вік, стать, порода, власник та кнопка пошуку.

Використаємо поля для знаходження коня.

Введемо такі дані у поля пошуку:

Кличка: Test

Вік: 13

Стать: Самка

Порода: Negro

Власник: Test

Натискаємо на кнопку пошуку.

Система оновлює таблицю та показує коня, що відповідає усім параметрам пошуку та має такі дані:

Кличка: Test Horse

Вік: 13

Стать: Самка

Порода: Negro

Власник: Test Owner

Можна зробити висновок, що система коректно шукає інформацію, тож випробування було пройдено успішно.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

У результаті виконання кваліфікаційної роботи було розроблено вебзастосунок для автоматизації обробки результатів кінних скачок.

На першому етапі було проведено аналіз практичної задачі, що необхідно розв’язати. Після цього було виконано пошук та дослідження аналогічних продуктів з метою визначення доцільності створення власного програмного забезпечення.

За результатами аналізу предметної галузі сформовано постановку задачі на розробку власного програмного забезпечення, було визначено функціонал та користувачів власної системи автоматизації. Визначено архітектуру, що найкраще відповідає вимогам розроблюваного програмного забезпечення.

На етапі ескізного проєктування було створено діаграми діяльності для визначення основних сценаріїв використання, розроблено концептуальну модель даних майбутнього програмного забезпечення та виконано моделювання інтерфейсу користувача.

У рамках технічного проєктування створено статичну модель, представлену у вигляді діаграми класів, а також побудовано динамічну модель роботи програмного продукту та логічну модель даних.

Під час робочого проєктування було обрано інструменти й засоби розробки програмного забезпечення. Основними з них стали мова Java, Spring Boot і СКБД MariaDB. Середовищем для розробки слугувала IntelliJ IDEA. Для кращого розуміння структури ПЗ, його залежностей і зв’язків між компонентами була створена відповідна діаграма. Після цього розпочалося безпосереднє програмування модулів системи.

Фінальним етапом стало тестування та налагодження ПЗ з метою виявлення та виправлення помилок. Для цього було підготовлено програму

тестування та методику, за якою виконувались усі випробування (додаток В).

Завершальним кроком була розробка докладної, зручної для користувача інструкції з використання, в якій наочно описано функціональні можливості ПЗ.

Технічне завдання представлено у додатку А, опис програмного забезпечення у додатку Б, програму та методику випробувань у додатку В, інструкцію користувача у додатку Г, текст програми у додатку Д.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Головною задачею охорони праці на підприємстві є відповідність умов праці чинному законодавству.

4.2 Основні законодавчі та нормативно-правові акти про охорону праці в Україні

Охорона праці в Україні регулюється низкою законодавчих і нормативно-правових актів, які спрямовані на забезпечення безпечних і здорових умов праці. До основних належать:

- Конституція України – встановлює право громадян на належні, безпечні та здорові умови праці (стаття 43) [5].
- Кодекс законів про працю України – регулює трудові відносини, включаючи обов’язки роботодавців щодо створення безпечних умов праці.
- Закон України "Про охорону праці" – основний документ, що визначає права та обов’язки працівників і роботодавців у сфері охорони праці, принципи державної політики та механізми контролю.
- Закон України "Про загальнообов’язкове державне соціальне страхування" – регулює страхування від нещасних випадків на виробництві та професійних захворювань.
- Нормативно-правові акти з охорони праці (НПАОП) – включають правила, норми, інструкції та стандарти, затверджені центральними органами виконавчої влади, зокрема Держпраці.

- Міжнародні стандарти та конвенції МОП, ратифіковані Україною, наприклад, Конвенція № 155 про безпеку, здоров'я працівників та виробниче середовище.

Ці акти створюють правову основу для реалізації заходів з охорони праці, спрямованих на збереження життя, здоров'я та працездатності працівників [6].

4.3 Аналіз небезпечних і шкідливих факторів при роботі з комп'ютером

Робота з комп'ютером є невід'ємною частиною багатьох професій, але вона пов'язана з низкою небезпечних і шкідливих факторів, які можуть негативно впливати на здоров'я працівників. Ці фактори умовно поділяються на фізичні, хімічні, біологічні, психофізіологічні та організаційні. Нижче проведено детальний аналіз основних небезпечних і шкідливих факторів, характерних для роботи з комп'ютером.

Фізичні фактори

- Електромагнітне випромінювання: Монітори, системні блоки та інші компоненти комп'ютера генерують електромагнітні поля низької інтенсивності. Хоча сучасні пристрої відповідають стандартам безпеки, тривалий вплив може викликати дискомфорт, головний біль або порушення сну.

- Недостатнє або надмірне освітлення: Неправильне освітлення робочого місця (яскраве світло, відблиски на екрані, низька освітленість) призводить до напруження очей, погіршення зору та втоми.

- Шум: Постійний шум від роботи вентиляторів системного блоку, клавіатури чи принтерів може створювати дискомфорт, знижувати концентрацію та викликати подразнення.

- Температурний режим: Недостатня вентиляція або перегрівання обладнання може спричинити підвищення температури в приміщенні, що негативно впливає на самопочуття працівника.

Ергономічні фактори

- Незручна поза: Тривале сидіння в статичній позі, неправильне розташування крісла, столу чи монітора призводить до болю в спині, шиї, зап’ястях (синдром карпального каналу), а також до розвитку остеохондрозу чи інших захворювань опорно-рухового апарату.

- Неправильне розташування обладнання: Занадто високе або низьке розташування монітора, клавіатури чи миші змушує працівника напружувати м’язи, що сприяє швидкій втомлюваності.

- Тривале статичне навантаження: Постійна робота за комп’ютером без перерв викликає перенапруження м’язів, порушення кровообігу та зниження працездатності.

Психофізіологічні фактори

- Зорова напруга: Тривала концентрація на екрані, миготіння зображення, дрібний шрифт або низька контрастність викликають синдром комп’ютерного зору (CVS), який проявляється сухістю очей, почервонінням, різзю та погіршенням зору.

- Психоемоційне навантаження: Висока відповідальність, багатозадачність, терміни виконання завдань або монотонність роботи можуть спричинити стрес, тривожність, емоційне вигорання.

- Гіподинамія: Малорухливий спосіб життя, пов’язаний із тривалим сидінням, призводить до порушення обміну речовин, ожиріння, проблем із серцево-судинною системою.

Хімічні фактори

- Виділення шкідливих речовин: Пластикові корпуси комп’ютерів, принтери чи інші офісні пристрої за нагрівання можуть виділяти леткі органічні сполуки (наприклад, формальдегід), які подразнюють дихальні шляхи та очі.

- Пил: Накопичення пилу всередині комп’ютерного обладнання може спричинити алергічні реакції або подразнення шкіри.

Біологічні фактори

- Мікроорганізми: На клавіатурі, миші та інших поверхнях накопичуються бактерії та віруси, що підвищує ризик інфекційних захворювань, особливо в офісах із великою кількістю працівників.

- Статична електрика: Вона може сприяти накопиченню пилу та мікроорганізмів, а також викликати дискомфорт при контакті з обладнанням.

Організаційні фактори

- Нераціональний режим праці: Відсутність регламентованих перерв, тривала робота без відпочинку сприяють накопиченню втоми та зниженню продуктивності.

- Недостатня вентиляція: Погана циркуляція повітря в приміщенні призводить до накопичення вуглекислого газу, зниження концентрації кисню, що викликає сонливість і зниження працездатності.

4.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів при роботі з комп’ютером

Для мінімізації впливу небезпечних і шкідливих факторів при роботі з комп’ютером необхідно впроваджувати комплексні заходи, спрямовані на покращення умов праці, оптимізацію робочого місця та підвищення культури безпеки. Нижче наведено детально розроблені заходи, які охоплюють організаційні, технічні, ергономічні та профілактичні аспекти.

Організаційні заходи

- Регламентація режиму праці та відпочинку: Встановлення перерв через кожні 45–60 хвилин роботи (5–10 хвилин для відпочинку очей, легкої фізичної активності). Наприклад, виконання вправ для очей (мигання, переведення погляду на далекі об’єкти) або коротка розминка для шиї та спини.

- Навчання та інструктажі: Проведення регулярних тренінгів з ергономіки, правил безпечної роботи за комп’ютером і профілактики професійних захворювань. Працівники мають знати про ризики синдрому комп’ютерного зору, гіподинамії та стресу.

- Контроль тривалості роботи: Обмеження безперервної роботи за комп’ютером до 6 годин на день із обов’язковими перервами, особливо для працівників із високим зоровим навантаженням.

Ергономічні заходи

Облаштування робочого місця:

- Висота столу та крісла повинна регулюватися так, щоб ноги працівника стояли на підлозі, а коліна були зігнуті під кутом 90°.

- Монітор розташовується на відстані 50–70 см від очей, верхній край екрана – на рівні очей або трохи нижче.

- Клавіатура та миша розміщуються так, щоб лікті залишалися зігнутими під кутом 90°, а зап’ястя не перенапружувалися.

- Використання ергономічного обладнання: Застосування крісел із підтримкою попереку, підставок для моніторів, ергономічних клавіатур і мишей із підтримкою зап’ястя.

- Запобігання статичним позам: Надання можливості періодично змінювати положення тіла (наприклад, використання столів із регулюванням висоти для роботи стоячи).

Технічні заходи

- Контроль електромагнітного випромінювання: Використання сучасних моніторів із низьким рівнем випромінювання (стандарти TCO або Blue Light Filter). Регулярна перевірка обладнання на відповідність нормам безпеки.

- Оптимізація освітлення: Забезпечення рівномірного освітлення робочого місця (300–500 люкс), уникнення відблисків на екрані шляхом встановлення матових екранів або жалюзі. Використання ламп із нейтральним спектром світла.

- Зниження рівня шуму: Встановлення безшумних вентиляторів, використання шумопоглинаючих матеріалів у приміщенні, ізоляція джерел шуму (наприклад, принтерів).

- Забезпечення вентиляції: Регулярне провітрювання приміщення, використання систем кондиціонування для підтримання температури 20–24°C і вологості 40–60%.

Профілактичні заходи

- Медичні огляди: Організація періодичних оглядів у офтальмолога та ортопеда для працівників, які тривалий час працюють за комп'ютером, для раннього виявлення проблем із зором або опорно-руховим апаратом.

- Фізична активність: Впровадження корпоративних програм із фізкультури (наприклад, щоденні 10-хвилинні розминки в офісі, доступ до тренажерних залів). Рекомендується виконання вправ для розтягування м'язів спини, шиї та кистей.

- Профілактика синдрому комп'ютерного зору: Використання захисних окулярів із фільтрами синього світла, застосування зволожуючих крапель для очей, налаштування екрана (збільшення шрифту, зниження яскравості, підвищення частоти оновлення).

Гігієнічні заходи

- Регулярне прибирання робочого місця: Очищення клавіатури, миші та монітора дезінфікуючими засобами для зниження ризику накопичення бактерій.

- Контроль чистоти обладнання: Періодичне чищення внутрішніх компонентів комп'ютера (вентиляторів, системного блоку) для зменшення накопичення пилу та шкідливих речовин.

- Забезпечення чистого повітря: Використання очищувачів повітря або іонізаторів для зменшення концентрації пилу та алергенів.

4.5 Санітарні норми мікроклімату виробничих приміщень

Гігієнічні норми стосуються параметрів мікроклімату в межах робочих зон виробничих об’єктів, установ, організацій та підприємств — незалежно від форми їх власності чи підпорядкування [7].

У нормативному документі наведено допустимі та оптимальні межі мікрокліматичних показників, а також визначено порядок проведення вимірювань і критерії оцінювання мікроклімату.

Основні вимоги до мікроклімату

Мікроклімат у приміщеннях виробничого призначення визначається такими параметрами:

– температура повітря,

– відносна вологість,

– швидкість повітряного потоку,

– рівень теплового (інфрачервоного) випромінювання,

– температура поверхонь.

Для робочої зони передбачені як оптимальні, так і допустимі значення мікрокліматичних характеристик. Вони встановлюються залежно від пори року та фізичної складності виконуваної роботи. Якщо в межах одного простору виконуються роботи з різним ступенем навантаження, орієнтуватися слід на категорію працівників, яка є найчисельнішою [8].

4.6 Пожежна безпека

В приміщенні, де здійснювалася розробка вебдзастосунку, використовуються тільки негорючі речовини та матеріали у холодному стані, тому за ступенем вибухопожежної та пожежної небезпеки приміщення відноситься до категорії «Д». Пожежну небезпеку несуть у собі лише кабельні електропроводки до обладнання, що є припустимим для даної категорії

приміщень.

За вогнестійкістю приміщення відноситься до другої категорії згідно з ДБН В.1.1.7-2002. Робоча зона дослідника відноситься до класу вибухонебезпечності В-IIа та пожежонебезпечності П-IIа, оскільки вибухонебезпечна концентрація пилу і волокон може утворюватися лише внаслідок аварії або несправності [9].

4.7 Висновки

В цілому приміщення по категорії протипожежної безпеки відповідає нормам, але особливу увагу потрібно звернути на утримання в справному стані засобів протипожежного захисту та своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного захисту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи мета, яка ставилася в роботі, була повністю досягнута, а саме було розроблено вебзастосунок для автоматизації обробки результатів кінних скачок, що забезпечить зручність та ефективність у роботі адміністративного та суддівського складу.

Для досягнення поставленої мети було вирішено наступні задачі:

– проведено аналіз практичної задачі інженерії програмного забезпечення та опис предметної галузі програмного забезпечення.

– сформульовано вимоги до розроблюваного вебзастосунку та розроблено технічне завдання.

– проведено аналіз практичної задачі інженерії програмного забезпечення.

– проведено аналіз існуючих аналогів.

– зроблено постановку задачі на розробку вебзастосунку для автоматизації обробки результатів кінних скачок.

– розроблено архітектуру програмного забезпечення.

– проведено розробку ескізного, технічного та робочого проєктів вебзастосунку для автоматизації обробки результатів кінних скачок.

– розроблено документацію до вебзастосунку.

У результаті було створено вебзастосунок, що забезпечить зручність та ефективність у роботі адміністративного та суддівського складу завдяки автоматизації процесів роботи іподромів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МАГНАТ. URL: https://magnat.in.ua (дата звернення 25.03.2025).

2. КІННИЙ КЛУБ «ОЛІМП». URL: https://horse-club.com.ua/ua (дата звернення 25.03.2025).

3. Sport Elite. URL: https://www.sport-elite.com.ua/index.php/uk/ (дата звернення 25.03.2025).

4. Erwin Data Modeler. URL: https://www.erwin.com/products/erwin-data-modeler/ (дата звернення 25.03.2025).

5. Верховна Рада України. Законодавство України. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 30.05.2025).

6. Гороховатський В.О., Дубницький В.Ю., Кобилін А.М., Лукін В.О. та ін. Визначення трудомісткості при розробленні програмних комплексів / Системи обробки інформації. Харків: Харківський, 2014. с. 92-98.

7. ДБН В.1.1.7-2016 Пожежна безпека об’єктів будівництва. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=68456 (дата звернення 30.05.2022).

8. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення 30.05.2022).

9. Основи охорони праці: Підручник / За ред. проф. В.В.Березуцького. Харків: Факт, 2005. 480 с.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Повна назва продукту: Вебзастосунок для автоматизації обробки результатів кінних скачок.

Галузь застосування: автоматизація процесів, пов’язаних із роботою іподромів для проведення кінних скачок.

1 Підстави для розробки

Підставами для розробки вебзастосунку є завдання на кваліфікаційну роботу на здобуття ступеня вищої освіти «бакалавр».

2 Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням розробки є автоматизація роботи персоналу та суддівської колегії іподромів для проведення кінних скачок та покращення ефективності їх роботи, зниження складності роботи із програмним забезпеченням, збільшення зручності, зменшення кількості помилок, пришвидшення обробки інформації.

2.2 Функціональне призначення

Функціональне призначення полягає у автоматизації дій, котрі роблять адміністрація та суддівська колегія.

Адміністрація повинна мати можливість: авторизуватися; додавати інформацію про іподроми, змагання, коней, власників, жокеїв; переглядати будь-яку інформацію, що зберігається; редагувати інформацію про іподроми,

змагання, коней, власників, жокеїв; видаляти інформацію про іподроми, змагання, коней, власників, жокеїв; формувати звіти з результатами змагань.

Суддя повинен мати можливість: авторизуватися; додавати інформацію про порушення; редагувати інформацію про порушення; додавати інформацію про оцінки забігів; редагувати інформацію про оцінки забігів; видаляти інформацію про оцінки забігів; переглядати інформацію про іподроми; переглядати інформацію про змагання; переглядати інформацію про забіги; переглядати інформацію про коней; переглядати інформацію про жокеїв.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Розроблюване програмне забезпечення повинно реалізовувати наступний функціонал:

- додавання, редагування та видалення інформації про іподроми;
- додавання, редагування та видалення інформації про змагання;
- додавання, редагування та видалення інформації про забіги;
- додавання, редагування та видалення інформації про результати;
- додавання, редагування інформації про порушення;
- додавання, редагування та видалення інформації про коней;
- додавання, редагування та видалення інформації про жокеїв;
- додавання, редагування та видалення інформації про власників;
- підрахунок результатів змагання;
- розрахунок рейтингу жокеїв;
- формування звітів з інформацією про скачки, дату їх проведення та результати;
- формування звітів з інформацією про загальну кількість жокеїв, коней,

їх власників та змагань;

- формування звітів з відображення результатів змагань за заданим періодом (дата та час, іподром).

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення даних відбувається за допомогою клавіатури та миші.

Взаємодія йде за допомогою вебзастосунку. Під час редагування та внесення інформації в базу даних вносяться такі дані:

- інформація про іподром (назва, адреса, довжина доріжки);
- інформація про змагання (назва змагання, дата змагання, іподром);
- інформація про забіг (назва забігу, оцінка, змагання);
- інформація про порушення (назва порушення, кількість знятих балів);
- інформація про коня (кличка, вік, порода, власник, стать);
- інформація про власника (ПІБ, адреса, телефон);
- інформація про жокея (ПІБ, дата народження, рейтинг);
- інформація про результат (місце, кількість балів, забіг, кінь, жокей).

Виведення даних відбувається за допомогою екрана ноутбуку або комп’ютера, інформація отримується у вигляді повідомлень та звітів.

Вихідними даними будуть повідомлення та звіти:

- повідомлення про результати обробки іподромів;
- повідомлення про результати обробки змагань;
- повідомлення про результати обробки забігів;
- повідомлення про результати обробки результатів;
- повідомлення про результати обробки порушень;
- повідомлення про результати обробки коней;
- повідомлення про результати обробки власників;
- повідомлення про результати обробки жокеїв;
- звіт з інформацією про скачки, дату їх проведення та результати;

- звіт з інформацією про загальну кількість жокеїв, коней, їх власників та змагань;

- звіт з відображенням результатів змагань за заданим періодом (дата та час, іподром).

3.2 Вимоги до надійності

Програма повинна надійно працювати за допомогою використання ряду організаційно-технічних заходів, які перераховані нижче:

- резервне копіювання: повинна створюватись резервна копія бази даних і файлів конфігурації, забезпечуватись їх зберігання для відновлення роботи у випадку втрати інформації.

- захист даних від несанкціонованого доступу.

3.3 Вимоги до складу й параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на сервері наступної мінімальної комплекції:

- процесор: AMD Ryzen 5 5500 3.6 Ghz або аналогічний йому Intel Core i5-11400F;

- 4 ГБ оперативної пам’яті;

- 48 ГБ вільного місця на SSD.

- з’єднання з локальною мережею.

Також є клієнтський комп’ютер, що повинен задовольняти наступну мінімальну конфігурацію:

- AMD Ryzen 5 3600 3.6 Ghz або аналогічний йому Intel Core i5-9400F;

- 2 ГБ оперативної пам’яті;

- з’єднання з локальною мережею.

3.4 Вимоги до інформаційної й програмної сумісності

Для роботи з програмним забезпеченням потрібно мати сервер, що матиме:

- операційну систему, що є не нижчою за Windows 10;
- наявність системи управління базами даних MySQL.

Клієнтський комп’ютер повинен мати:

- операційну систему, що не є нижчою за Windows 10;
- веббраузер Chrome версії не нижче 119.0.6045.159.

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програму та методику випробувань;
- керівництво користувача;
- опис програми;
- код програми.

5 Техніко-економічні показники

Техніко-економічні показники не розраховувались.

6 Стадії та етапи розробки

Стадії та етапи розробки ПЗ представлені у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки ПЗ

№	Стадії та етапи	Термін виконання
1	Аналіз практичної задачі інженерії ПЗ	13.02.2025
2	Аналіз вимог до ПЗ	21.02.2025
3	Розробка архітектури ПЗ	11.03.2025
4	Розробка проєкту ПЗ	14.04.2025
5	Реалізація та тестування ПЗ	24.05.2025

7 Порядок контролю й приймання

Контроль за аналізом та проєктуванням кожної окремої частини програмного забезпечення здійснюється на кожному етапі з урахуванням вимог, визначених у технічному завданні. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводяться відповідно до програми і методики випробування і документуються за допомогою протоколу проведення випробувань. У разі знаходження помилок під час приймання програмного продукту складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівником організації – замовника та організації – розробника.

Розробник повинен протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проходження перевірки.

ДОДАТОК Б – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Загальні відомості

Розробленний вебзастосунок для автоматизації обробки результатів кінних скачок представляє собою інструмент, що надає можливість керувати процесом та обліком кінних скачок.

Для розробки програмного забезпечення були обрані наступні засоби:

– Мовою програмування була обрана Java;
– Було обрано MySQL в якості бази даних;
– В якості середовища розробки було обрано IntelliJ IDEA.

Для функціонування програмного забезпечення необхідна наявність персонального комп’ютера чи ноутбука з операційною системою Windows 10 або новішої. Також необхідна наявність підключення до локального серверу для коректної роботи з базою даних MySQL.

2 Функціональне призначення

Вебзастосунок виконує наступні основні функції:

– додавання, редагування та видалення інформації про іподроми;
– додавання, редагування та видалення інформації про змагання;
– додавання, редагування та видалення інформації про забіги;
– додавання, редагування та видалення інформації про результати;
– додавання, редагування інформації про порушення;
– додавання, редагування та видалення інформації про коней;
– додавання, редагування та видалення інформації про жокеїв;
– додавання, редагування та видалення інформації про власників;
– підрахунок результатів змагання;
– розрахунок рейтингу жокеїв;

– формування звітів з інформацією про скачки, дату їх проведення та результати;

– формування звітів з інформацією про загальну кількість жокеїв, коней, їх власників та змагань;

– формування звітів з відображення результатів змагань за заданим періодом (дата та час, іподром).

3 Опис логічної структури

Програмне забезпечення розроблене на основі модульної архітектури, де кожен окремий модуль відповідає за виконання певної функції. Наприклад, один модуль здійснює додавання інформації про коней, тоді як інший забезпечує формування звітів із відображенням результатів змагань за обраний період. Такий підхід сприяє підвищенню зручності у підтримці, масштабуванні та повторному використанні компонентів системи.

Окрім цього, система реалізована за принципом клієнт-серверної архітектури: клієнтська частина представлена у вигляді вебзастосунку, який взаємодіє з серверною частиною, що працює на основі бази даних MySQL. Це забезпечує безперебійну роботу додатку як під час проведення забігів, так і у будь-який інший час.

4 Технічні засоби

Для роботи з програмним забезпеченням потрібно мати склад технічних засобів з мінімальними параметрами:

– CPU: AMD Ryzer 5 3600 3.6 МГц;

– RAM: 4 GB;

– SSD з 10гб вільного місця.

– пристрій вводу-виводу: маніпулятор типу «Миша», монітор, клавіатура.

5 Виклик та завантаження

Вебзастосунок встановлюється безпосередньо на пристрій під керування операційної системи Windows. Запуск вебзастосунку здійснюється через підключення до бази даних за допомогою Open Server Panel та відкриття локальної сторінки у браузері.

Після запуску відкривається вікно авторизації користувача, де необхідно ввести логін та пароль для входу. У випадку правильних даних відкривається головна сторінка вебзастосунку, де можна побачити список усіх наявних таблиць та звітів.

6 Вхідні та вихідні дані

Дані зберігаються в локальній базі даних MySQL

Вхідні дані:

Введення даних відбувається за допомогою клавіатури та миші.

Взаємодія йде за допомогою вебзастосунку. Під час редагування та внесення інформації в базу даних вносяться такі дані:

– інформація про іподром (назва, адреса, довжина доріжки);
– інформація про змагання (назва змагання, дата змагання, іподром);
– інформація про забіг (назва забігу, оцінка, змагання);
– інформація про порушення (назва порушення, кількість знятих балів);
– інформація про коня (кличка, вік, порода, власник, стать);
– інформація про власника (ПІБ, адреса, телефон);
– інформація про жокея (ПІБ, дата народження, рейтинг);
– інформація про результат (місце, кількість балів, забіг, кінь, жокей).

Вихідні дані:

Виведення даних відбувається за допомогою екрана ноутбуку або

комп’ютера, інформація отримується у вигляді повідомлень та звітів.

Вихідними даними будуть повідомлення та звіти з інформацією про:

– повідомлення про результати обробки іподромів;
– повідомлення про результати обробки змагань;
– повідомлення про результати обробки забігів;
– повідомлення про результати обробки результатів;
– повідомлення про результати обробки порушень;
– повідомлення про результати обробки коней;
– повідомлення про результати обробки власників;
– повідомлення про результати обробки жокеїв;
– звіт з інформацією про скачки, дату їх проведення та результати;
– звіт з інформацією про загальну кількість жокеїв, коней, їх власників та змагань;
– звіт з відображенням результатів змагань за заданим періодом (дата та час, іподром).

ДОДАТОК В – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

1 Об’єкт випробувань

Повна назва об’єкта випробувань: вебзастосунок для автоматизації обробки результатів кінних скачок.

Область застосування: Функціонал вебзастосунку буде розроблено таким чином, що адміністратори повинні мати можливість вносити і коригувати дані, необхідні для обліку результатів, тоді як судді повинні мати можливість вносити свої оцінки, коментарі та результати без необхідності прямої взаємодії з адміністраторами.

2 Мета випробувань

Метою випробувань є перевірка відповідності розробленого програмного забезпечення вимогам, які викладені в технічному завданні.

3 Вимоги до програми

Під час випробування програмного забезпечення необхідно перевірити, чи відповідають її функціональні характеристики тим вимогам, що зазначено у розділі «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

Програмна документація має містити:

- технічне завдання;
- опис програми;
- програму та методику випробувань;
- інструкцію користувача;

– тексти програмних модулів.

5 Засоби та порядок випробувань

Програмні засоби випробувань

Програмними засобами, необхідними для випробувань, є: ОС Windows версії не нижче 10, наявність встановленого IntelliJ IDEA та Open Server Panel.

Апаратні засоби випробувань

Для випробувань необхідно мати апаратні засоби, що мають компоненти персонального комп’ютера, такі як:

– CPU: AMD Ryzer 5 3600 3.6 МГц;

– RAM: 4 GB;

– SSD з 10гб вільного місця.

– пристрій вводу-виводу: маніпулятор типу «Миша», монітор, клавіатура.

Порядок проведення випробувань

Проведення випробувань програмного забезпечення має відбуватися в наступному порядку:

– виконання інструкцій до кожної функції;

– аналіз результатів і встановлення відповідності вебзастосунку вимогам і системним документам.

У випадку виявлення помилок у роботі, необхідно скласти перелік знайдених проблем і визначити терміни їх виправлення.

Після виправлення зазначених помилок, необхідно протестувати програмне забезпечення повторно у повному обсязі.

6 Методика випробувань

Випробування будуть проводитись за методикою «чорного ящика». Через велику кількість функцій буде представлено декілька результатів випробувань функцій програми. В таблиці В.1 наведено програму випробувань вебзастосунку.

Таблиця В.1 – Програма випробувань вебзастосунку

№	Дія	Очікуваний результат
1	Додавання сутності Кінь до таблиці коней	Вебзастосунок виводить повідомлення про успішне додавання Коня
2	Редагування сутності Кінь у таблиці коней	Вебзастосунок виводить повідомлення про успішне редагування Коня
3	Видалення сутності Кінь у таблиці коней	Вебзастосунок виводить повідомлення про успішне видалення Коня
4	Видалення власника, що має коней.	Вебзастосунок видає попередження про видалення власника, що має коней. У випадку погодження, видаляється власник та усі пов’язані із ним коні.
5	Перегляд інформації про власників	Вебзастосунок успішно виводить інформацію про усі іподроми.
6	Пошук коня за його інформацією	Вебзастосунок успішно виводить інформацію про знайдені іподроми.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

Призначення програми

Функціональним призначенням вебзастосунку є автоматизація обробки результатів кінних скачок для адміністрації та суддів, обробка інформації про коней, іподроми, жокеїв, власників, змагань та забігів, універсальність та зручність використання будь-якою із компаній, що займаються кінними скачками.

Експлуатаційним призначенням розробки є автоматизація роботи персоналу та суддівської колегії іподромів для проведення кінних скачок та покращення ефективності їх роботи, зниження складності роботи із програмним забезпеченням, збільшення зручності, зменшення кількості помилок, пришвидшення обробки інформації.

Вебзастосунок має можливість виконання наступних функцій:

1) Авторизація

Вхідні дані: логін, пароль;

Вихідні дані: результат авторизації;

2) Обробка інформації про коней (додавання, редагування та видалення інформації)

Вхідні дані: кличка, вік, порода, власник, стать

Вихідні дані: результат додавання, редагування, видалення.

3) Обробка інформації про змагання (додавання, редагування та видалення інформації)

Вхідні дані: назва змагання, дата змагання;

Вихідні дані: результат додавання, редагування, видалення.

4) Обробка інформації про забіги (додавання, редагування та видалення інформації)

Вхідні дані: жокей, кінь, оцінка, порушення;

Вихідні дані: результат додавання, редагування, видалення.

5) Обробка інформації про порушення (додавання та редагування інформації)

Вхідні дані: назва порушення, кількість знятих балів

Вихідні дані: результат додавання, редагування.

6) Обробка інформації про іподроми (додавання, редагування та видалення інформації)

Вхідні дані: назва, адреса, довжина доріжки

Вихідні дані: результат додавання, редагування, видалення.

7) Обробка інформації про жокеїв (додавання, редагування та видалення інформації)

Вхідні дані: ПІБ, дата народження, рейтинг

Вихідні дані: результат додавання, редагування, видалення.

8) Обробка інформації про власників (додавання, редагування та видалення інформації)

Вхідні дані: ПІБ, адреса, телефон

Вихідні дані: результат додавання, редагування, видалення.

10) Підрахунок результатів змагання

Вхідні дані: жокей, кінь, власник, кількість балів;

Вихідні дані: місце у забігу.

11) Розрахунок рейтингу жокеїв

Вхідні дані: жокей, результат, порушення;

Вихідні дані: рейтинг жокея.

12) Формування звітів з інформацією про скачки, дату їх проведення та результати

Вхідні дані: забіг, дата;

Вихідні дані: звіт з інформацією про скачки, дату їх проведення та результати.

13) Формування звітів з інформацією про загальну кількість жокеїв, коней, їх власників та змагань.

Вхідні дані: жокеї, коні, власники, змагання;

Вихідні дані: звіт з інформацією про загальну кількість жокеїв, коней, їх власників та змагань.

14) Формування звітів з відображення результатів змагань за заданим періодом (дата, іподром).

Вхідні дані: іподром, дата;

Вихідні дані: звіт з відображення результатів змагань за заданим періодом.

Умови виконання вебзастосунку

Для роботи з вебзастосунком необхідний наступ склад технічних засобів з мінімальними параметрами:

– CPU: AMD Ryzer 5 3600 3.6 МГц;
– RAM: 4 GB;
– SSD з 10гб вільного місця.
– пристрій вводу-виводу: маніпулятор типу «Миша», монітор, клавіатура.

Для вмілого користування вебзастосунком, навички користування комп’ютером та програмним забезпеченням повинні відповідати таким критеріям:

– Користувачі повинні володіти базовими навичками роботи із комп’ютерами, а саме: вміння працювати з мишею, клавіатурою. Знати, як

виконується увімкнення комп’ютера та запуск вебзастосунку. Як працювати з даними із бази даних.

– Користувачі повинні бути здатні до ефективної роботи з інтерфейсом вебзастосунку, а саме: знаходження необхідного функціоналу, використання користувацького меню та виконання операцій обробки інформації.

– Користувачі повинні мати розуміння веббраузеру, яким користуються та мати базове розуміння операційної системи, що використовується під час роботи.

Ці вимоги допоможуть забезпечити впевненість, що користувачі зможуть ефективно та правильно використовувати вебзастосунок на необхідному рівні та зможуть досягти необхідних цілей без перешкод.

Виконання програми

Вхід у вебзастосунок

Для запуску вебзастосунку користувач повинен натиснути двічі на значок програми OpenServer.exe та запустити сервер програми.

Після цього у браузері необхідно відкрити локальний сайт localhost:8080, де буде показано вікно авторизації, його представлено на рисунку Г.1. Для авторизації необхідно ввести логін та пароль, після чого натиснути на кнопку «Увійти».

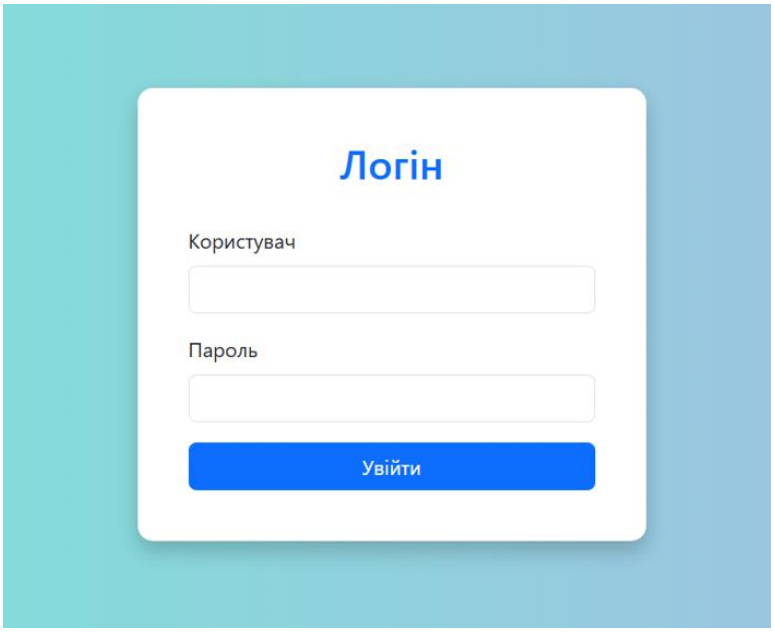


Рисунок Г.1 – Вікно авторизації користувача

Головна сторінка вебзастосунку

Після авторизації користувач потрапляє на головну сторінку вебзастосунку, яка є центральним навігаційним вузлом для доступу до всіх функціональних модулів системи. Інтерфейс виконано у сучасному стилі з акцентом на зручність користування. Головну сторінку представлено на рисунку Г.2

На головній сторінці розміщено дев’ять інтерактивних кнопок, кожна з яких відповідає за перехід до окремого функціонального розділу системи:

– Коні – управління обліком коней, включаючи додавання, редагування, перегляд та видалення інформації про кожного коня.

– Власники – перегляд і редагування даних власників коней.

– Жокеї – модуль для ведення інформації про жокеїв.

– Забіги – створення та перегляд інформації про конкретні забіги (час, місце, учасники).

– Змагання – модуль для організації змагань, що можуть об'єднувати

кілька забігів.

– Іподроми – база даних іподромів з детальним описом умов проведення перегонів.

– Порушення – реєстрація та перегляд порушень правил під час перегонів або підготовки.

– Результати – перегляд результатів змагань, зокрема місця, час проходження дистанцій, та іншу статистику.

– Звіти – формування та вивантаження звітної документації за різними критеріями.

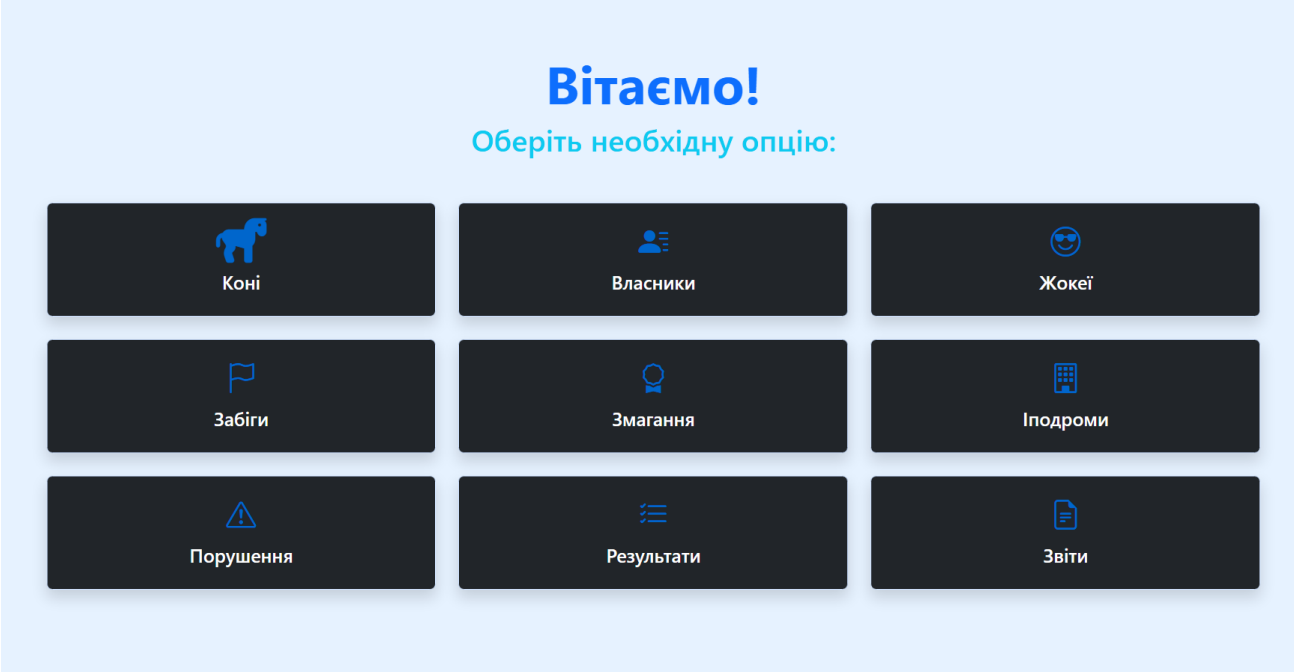


Рисунок Г.2 – Головна сторінка вебзастосунку

Натискаючи на кожний із пунктів, користувача буде перенаправлено на відповідну сторінку.

Робота з таблицями

На рисунку Г.3 представлено таблицю з інформацією про коней.

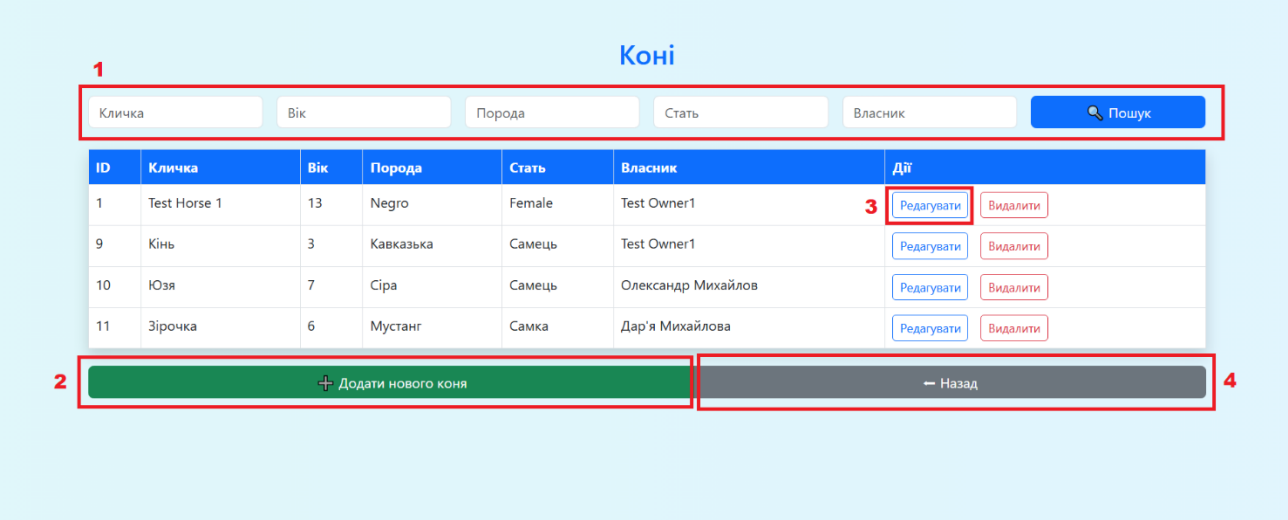


Рисунок Г.3 – Таблиця «Коні»

У вебзастосунку є можливість знайти коней за необхідними параметрами. Для цього необхідно ввести дані для пошуку у відповідний модуль пошуку – його зазначено на рисунку Г.3 під цифрою 1.

Виконання пошуку видає необхідні результати у таблиці після натискання кнопки «Пошук». Результати пошуку представлено на Рисунку Г.4

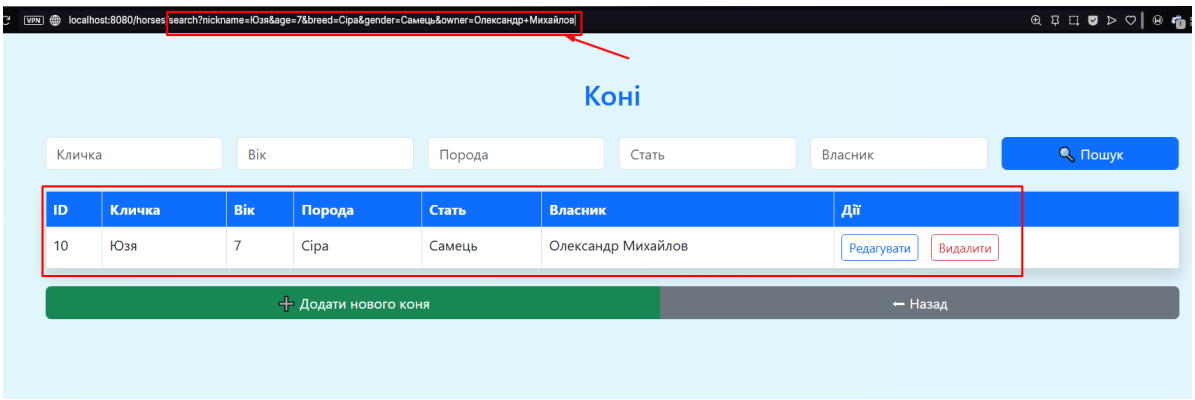


Рисунок Г.4 – Результати пошуку

При натисканні кнопки «Додати нового коня» відкриється форма, що дозволить ввести дані нового коня та додати його до бази даних. На рисунку Г.3 його зазначено цифрою 2.

На рисунку Г.5 представлено форму додавання нового коня.

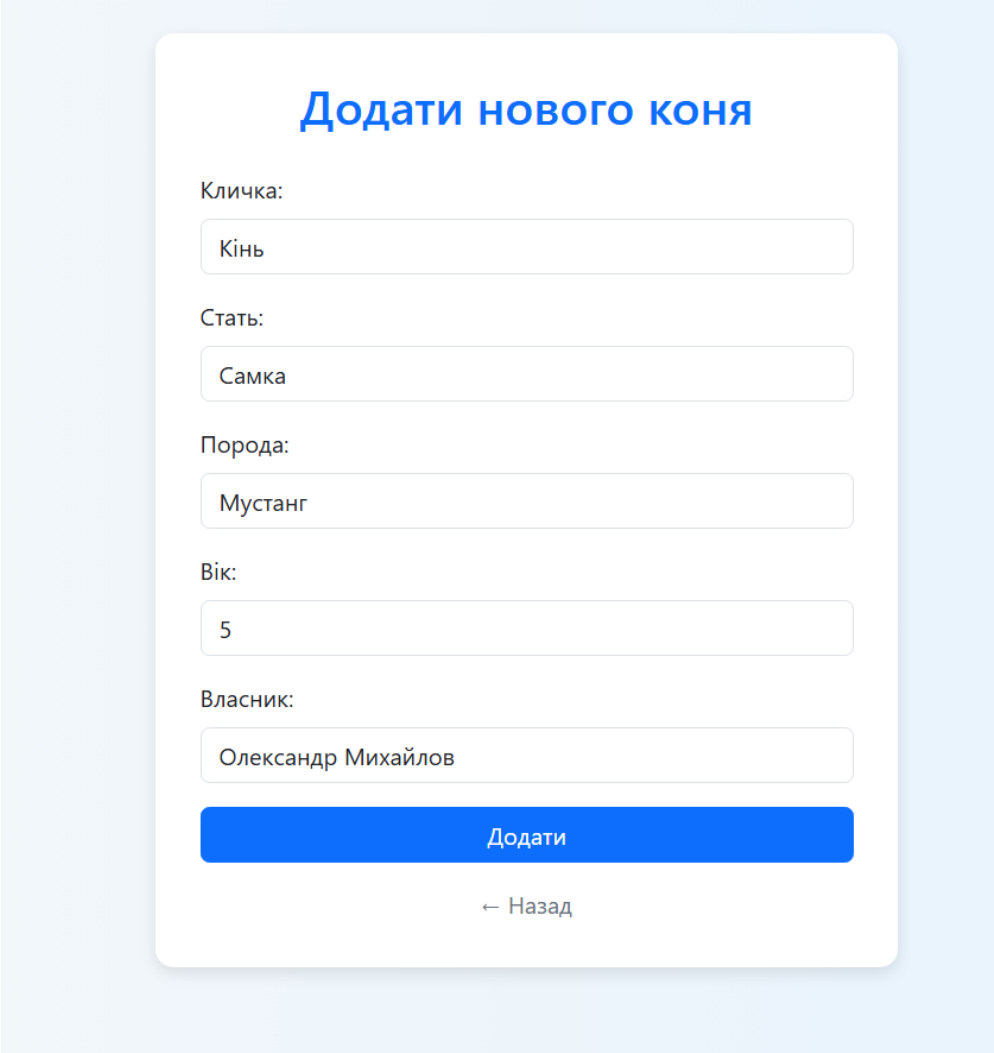


Рисунок Г.5 – Форма додавання нового коня

Натискаючи на кнопку «Редагувати» відкриється форма редагування вже існуючого коня із уже існуючими даними. Під час редагування можна змінити будь-яку інформацію про коня та натиснути на кнопку «Редагувати». Кнопка редагування представлена на рисунку Г.3 цифрою 3.

Поряд з кнопкою редагування присутня кнопка видалення коня. Натискаючи на неї, користувач повинен підтвердити видалення, після чого кінь буде видалений у випадку підтвердження.

Цифрою 4 на рисунку Г.3 позначена кнопка «Назад», що надає можливість повернутися на головну сторінку вебзастосунку.

Формування звітів

Натискаючи на кнопку «Звіти» на головному меню відкривається сторінка із можливими звітами. Обираючи варіант, користувач має можливість або одразу згенерувати необхідний звіт, якщо він не потребує додаткових параметрів, або відкрити пункт обирання необхідної інформації, яка надає конкретику для формування звіту. На рисунку Г.6 представлено сторінку формування звітів.

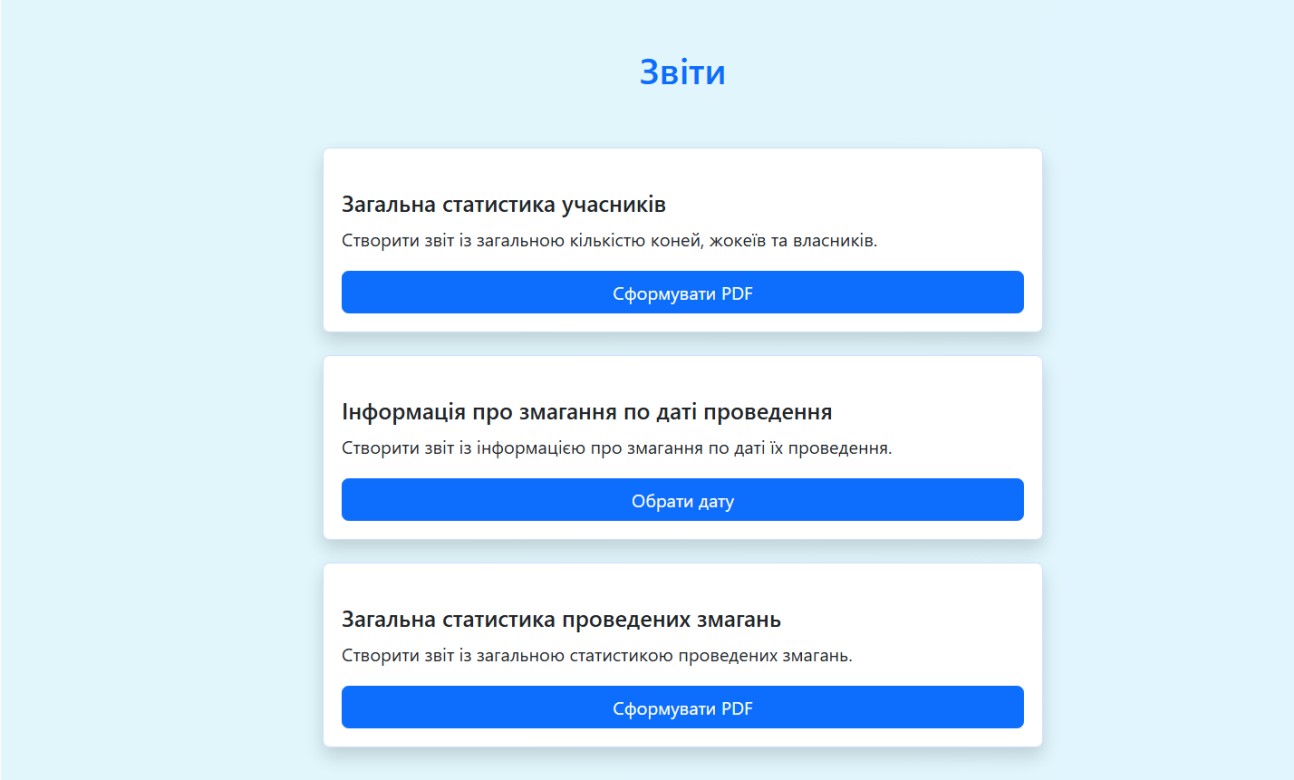


Рисунок Г.6 – Сторінка «Звіти»

Натискаючи на кнопку Сформувати PDF, звіт буде автоматично сформовано у вигляді PDF файлу та збережено на комп’ютері користувача.

Натискаючи «Обрати дату», користувач спочатку повинен обрати дату у спливаючому вікні, після чого звіт із необхідними даними сформується у форматі PDF.

На рисунку Г.7 представлено приклад звіту «Загальна статистика учасників», що надає інформацію про загальну кількість записів коней, жокеїв та власників у базі даних.

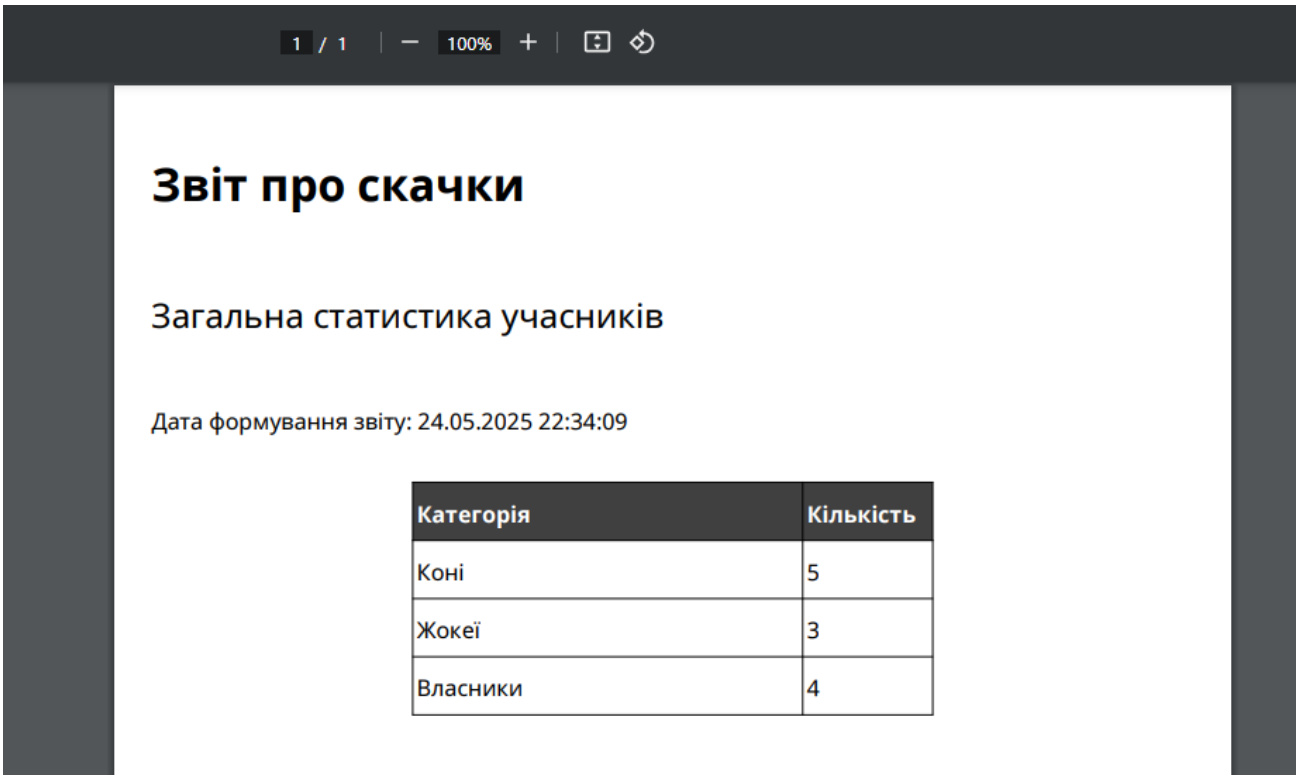


Рисунок Г.7 – Звіт «Загальна статистика учасників»

ДОДАТОК Д – ТЕКСТ ПРОГРАМИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ КІННИХ СКАЧОК

Нижче наведено текст вебзастосунку для автоматизації обробки результатів кінних скачок на мові Java.

HorseController.java

```
package org.example.labb.controller;

import org.example.labb.entity.Horse;
import org.example.labb.entity.Owner;
import org.example.labb.service.OwnerService;
import org.example.labb.service.HorseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
public class HorseController {
    @Autowired
    private HorseService horseService;
    @Autowired
    private OwnerService ownerService;


    @GetMapping("/horses")
    public String showHorses(Model model) {
        List<Horse> horses = horseService.getAllHorses();
        model.addAttribute("horses", horses);
        return "horses";
    }

    @GetMapping("/horses/search")
    public String searchHorses(@RequestParam(required = false) String nickname,
                               @RequestParam(required = false) Integer age,
                               @RequestParam(required = false) String breed,
                               @RequestParam(required = false) String gender,
                               @RequestParam(required = false) String owner,
                               Model model) {
        List<Horse> horses = horseService.searchHorses(nickname, age, breed,
gender, owner);
        model.addAttribute("horses", horses);
        return "horses";
    }

    @GetMapping("/showNewHorseForm")
    public String showNewHorseForm(Model model) {
```

```
        Horse horse = new Horse();
        model.addAttribute("horse", horse);
        return "new_horse";
    }


    @PostMapping("/saveHorse")
    public String saveHorse(@ModelAttribute("horse") Horse horse) {
        horseService.createHorse(horse);
        return "redirect:/horses";
    }

    @GetMapping("/showFormForUpdate/{id}")
    public String showFormForUpdate(@PathVariable(value = "id") int id, Model
model) {
        Horse horse = horseService.getHorseById(id).orElseThrow(() -> new
RuntimeException("Horse not found"));
        model.addAttribute("horse", horse);
        return "update_horse";
    }

    @PostMapping("/updateHorse/{id}")
    public String updateHorse(@PathVariable(value = "id") int id,
@ModelAttribute("horse") Horse newHorse) {

        Horse horse = horseService.getHorseById(id).orElseThrow(() -> new
RuntimeException("Horse not found"));

        Owner owner =
ownerService.getOwnerById(newHorse.getOwner().getId()).orElseThrow(() -> new
RuntimeException("Owner not found"));

        horse.setOwner(owner);

        horse.setNickname(newHorse.getNickname());
        horse.setGender(newHorse.getGender());
        horse.setBreed(newHorse.getBreed());
        horse.setAge(newHorse.getAge());

        horseService.updateHorse(id, horse);

        return "redirect:/horses";
    }
    @GetMapping("/deleteHorse/{id}")
    public String deleteHorse(@PathVariable(value = "id") int id) {
        this.horseService.deleteHorse(id);
        return "redirect:/horses";
    }

}
```

Horse.java

```
package org.example.labb.entity;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
```

```
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "horse", schema = "labs_java")
public class Horse {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idhorse", nullable = false)
    private Integer id;

    @Column(name = "nickname", length = 45)
    private String nickname;

    @Column(name = "gender", length = 45)
    private String gender;

    @Column(name = "breed", length = 45)
    private String breed;

    @Column(name = "age")
    private Integer age;

    @ManyToOne
    @JoinColumn(name = "owner_idowner", referencedColumnName = "idowner")
    private Owner owner;

    @OneToMany(mappedBy = "horse", cascade = CascadeType.ALL, orphanRemoval =
    true)
    private List<Result> results = new ArrayList<>();

    public Integer getId() {
        return id;
    }

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

    public String getNickname() {
        return nickname;
    }
    public void setNickname(String nickname) {
        this.nickname = nickname;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getBreed() {
        return breed;
    }
```

```
    public void setBreed(String breed) {
        this.breed = breed;
    }
    public Integer getAge() {
        return age;

    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Owner getOwner() {
        return owner;
    }

    public void setOwner(Owner owner) {
        this.owner = owner;
    }
    public List<Result> getResult() {
        return results;
    }

    public void setResult(List<Result> results) {
        this.results = results;
    }

}
```

HorseService.java

```
package org.example.labb.service;

import org.example.labb.entity.Horse;
import org.example.labb.entity.Jokey;
import org.example.labb.repository.HorseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class HorseService {

    @Autowired
    private HorseRepository horseRepository;

    public List<Horse> getHorsesByJokey(Jokey jokey) {
        return horseRepository.findByJokeysContains(jokey);
    }

    public Horse createHorse(Horse horse) {
        return horseRepository.save(horse);
    }

    public List<Horse> getAllHorses() {
```

```
        return (List<Horse>) horseRepository.findAll();
    }

    public Optional<Horse> getHorseById(int id) {
        return horseRepository.findById(id);
    }

    public Horse updateHorse(int id, Horse horse) {
        Horse existingHorse = horseRepository.findById(id).orElseThrow(() -> new
RuntimeException("Horse not found"));
        existingHorse.setNickname(horse.getNickname());
        return horseRepository.save(existingHorse);
    }

    public void deleteHorse(int id) {
        horseRepository.deleteById(id);
    }
    public List<Horse> searchHorses(String nickname, Integer age, String breed,
String gender, String ownerName) {
        return  horseRepository.findByParameters(nickname,  age,  breed,  gender,
ownerName);
    }


}
```

index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Скачки</title>
    <link                    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
    <link                          href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
    <link   rel="stylesheet"   href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">

    <style>
        body {
            background-color: #e6f2ff; /* Светло-голубой фон */
            color: #003366; /* Тёмно-синий текст */
            height: 100vh;
        }
        .overlay {
            height: 100%;
            padding-top: 80px;
        }
        .option-card {
            background-color: #ffffff;
            color: #003366;
            border: 1px solid #cce0ff;
            transition: transform 0.2s ease, box-shadow 0.2s ease;
        }
        .option-card:hover {
            transform: scale(1.05);
```

```
            box-shadow: 0 0 15px rgba(0, 102, 204, 0.3);
        }
        .option-card i {
            color: #0066cc;
        }
    </style>
</head>
<body>
<div class="overlay d-flex flex-column align-items-center text-center">
    <div class="container">
        <h1 class="display-4 text-primary fw-bold">Вітаємо!</h1>
        <h2 class="text-info mb-5">Оберіть необхідну опцію:</h2>

        <div class="row row-cols-1 row-cols-md-3 g-4">
            <div class="col">
                <a href="/horses" class="text-decoration-none">
                    <div class="card bg-dark text-white shadow option-card">
                        <div class="card-body">
                            <i class="fas fa-horse fa-3x"></i>
                            <h5 class="card-title mt-2">Коні</h5>
                        </div>
                    </div>
                </a>
            </div>
            <div class="col">
                <a href="/owners" class="text-decoration-none">
                    <div class="card bg-dark text-white shadow option-card">
                        <div class="card-body">
                            <i class="bi bi-person-lines-fill fs-2"></i>
                            <h5 class="card-title mt-2">Власники</h5>
                        </div>
                    </div>
                </a>
            </div>
            <div class="col">
                <a href="/jokeys" class="text-decoration-none">
                    <div class="card bg-dark text-white shadow option-card">
                        <div class="card-body">
                            <i class="bi bi-emoji-sunglasses fs-2"></i>
                            <h5 class="card-title mt-2">Жокеї</h5>
                        </div>
                    </div>
                </a>
            </div>
            <div class="col">
                <a href="/races" class="text-decoration-none">
                    <div class="card bg-dark text-white shadow option-card">
                        <div class="card-body">
                            <i class="bi bi-flag fs-2"></i>
                            <h5 class="card-title mt-2">Забіги</h5>
                        </div>
                    </div>
                </a>
            </div>
            <div class="col">
                <a href="/competitions" class="text-decoration-none">
                    <div class="card bg-dark text-white shadow option-card">
                        <div class="card-body">
                            <i class="bi bi-award fs-2"></i>
```

```
                        <h5 class="card-title mt-2">Змагання</h5>
                    </div>
                </div>
            </a>
        </div>
        <div class="col">
            <a href="/hippodromes" class="text-decoration-none">
                <div class="card bg-dark text-white shadow option-card">
                    <div class="card-body">
                        <i class="bi bi-building fs-2"></i>
                        <h5 class="card-title mt-2">Іподроми</h5>
                    </div>
                </div>
            </a>
        </div>
        <div class="col">
            <a href="/violations" class="text-decoration-none">
                <div class="card bg-dark text-white shadow option-card">
                    <div class="card-body">
                        <i class="bi bi-exclamation-triangle fs-2"></i>
                        <h5 class="card-title mt-2">Порушення</h5>
                    </div>
                </div>
            </a>
        </div>
        <div class="col">
            <a href="/results" class="text-decoration-none">
                <div class="card bg-dark text-white shadow option-card">
                    <div class="card-body">
                        <i class="bi bi-list-check fs-2"></i>
                        <h5 class="card-title mt-2">Результати</h5>
                    </div>
                </div>
            </a>
        </div>
        <div class="col">
            <a href="/reports" class="text-decoration-none">
                <div class="card bg-dark text-white shadow option-card">
                    <div class="card-body">
                        <i class="bi bi-file-earmark-text fs-2"></i>
                        <h5 class="card-title mt-2">Звіти</h5>
                    </div>
                </div>
            </a>
        </div>
    </div>
</div>
</div>

<script                        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

horses.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
```

```
<head>
  <meta charset="UTF-8">
  <title>Horses</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    body {
      background: linear-gradient(to right, #e0f7fa, #e1f5fe);
      font-family: 'Segoe UI', sans-serif;
    }
    .container {
      margin-top: 50px;
    }
    h2 {
      color: #0d6efd;
      text-align: center;
      margin-bottom: 30px;
    }
    table {
      background-color: white;
    }
    .table th {
      background-color: #0d6efd;
      color: white;
    }
    .action-links a {
      margin-right: 10px;
    }
    .btn-group {
      margin-top: 20px;
    }
  </style>
</head>
<body>
<div class="container">
  <h2>Коні</h2>
  <form class="row g-3 mb-4" method="get" action="/horses/search">
    <div class="col-md-2">
      <input       type="text"       name="nickname"       class="form-control"
placeholder="Кличка">
    </div>
    <div class="col-md-2">
      <input type="number" name="age" class="form-control" placeholder="Вік">
    </div>
    <div class="col-md-2">
      <input type="text" name="breed" class="form-control" placeholder="Порода">
    </div>
    <div class="col-md-2">
      <input type="text" name="gender" class="form-control" placeholder="Стать">
    </div>
    <div class="col-md-2">
      <input type="text" name="owner" class="form-control" placeholder="Власник">
    </div>
    <div class="col-md-2">
      <button type="submit" class="btn btn-primary w-100">🔍 Пошук</button>
    </div>
  </form>
```

```
  <table class="table table-bordered table-hover shadow">
    <thead>
    <tr>
      <th>ID</th>
      <th>Кличка</th>
      <th>Вік</th>
      <th>Порода</th>
      <th>Стать</th>
      <th>Власник</th>
      <th>Дії</th>
    </tr>
    </thead>
    <tbody>
    <tr th:if="${#lists.isEmpty(horses)}">
      <td colspan="7" class="text-center text-muted">Кінь не знайдений</td>
    </tr>
    <tr th:each="horse : ${horses}">
      <td th:text="${horse.id}"></td>
      <td th:text="${horse.nickname}"></td>
      <td th:text="${horse.age}"></td>
      <td th:text="${horse.breed}"></td>
      <td th:text="${horse.gender}"></td>
      <td th:text="${horse.owner.name}"></td>
      <td class="action-links">
        <a th:href="@{'/showFormForUpdate/' + ${horse.id}}" class="btn btn-sm btn-
outline-primary">Редагувати</a>
        <a th:href="@{'/deleteHorse/' + ${horse.id}}"
           class="btn btn-sm btn-outline-danger"
           onclick="return confirm('Ви впевнені, що хочете видалити цього
коня?')">
          Видалити
        </a>
      </td>
    </tr>
    </tbody>
  </table>

  <div class="btn-group d-flex justify-content-between">
    <a href="/showNewHorseForm" class="btn btn-success">➕ Додати нового коня</a>
    <a href="/" class="btn btn-secondary">⬅ Назад</a>
  </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

owners.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Власники</title>
    <link
```

```
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background: linear-gradient(to right, #e0f7fa, #e1f5fe);
            font-family: 'Segoe UI', sans-serif;
        }
        .container {
            margin-top: 50px;
        }
        h2 {
            color: #0d6efd;
            text-align: center;
            margin-bottom: 30px;
        }
        table {
            background-color: white;
        }
        .table th {
            background-color: #0d6efd;
            color: white;
        }
        .action-links a {
            margin-right: 10px;
        }
        .btn-group {
            margin-top: 20px;
        }
    </style>
</head>
<body>
<div class="container">
    <h2>Власники</h2>

    <table class="table table-bordered table-hover shadow">
        <thead>
        <tr>
            <th>ID</th>
            <th>Ім'я</th>
            <th>Дії</th>
        </tr>
        </thead>
        <tbody>
        <tr th:if="${#lists.isEmpty(owners)}">
            <td colspan="3" class="text-center text-muted">Жодного власника не
знайдено</td>
        </tr>
        <tr th:each="owner : ${owners}">
            <td th:text="${owner.id}"></td>
            <td th:text="${owner.name}"></td>
            <td class="action-links">
                <a    th:href="@{'/showFormForUpdateOwner/'    +    ${owner.id}}"
class="btn btn-sm btn-outline-primary">Редагувати</a>
                <a th:href="@{'/deleteOwner/' + ${owner.id}}"
                   class="btn btn-sm btn-outline-danger"
                   onclick="return confirm('Ви впевнені, що хочете видалити цього
власника?')">
                    Видалити
                </a>
```

```
            </td>
        </tr>
        </tbody>
    </table>

    <div class="btn-group d-flex justify-content-between">
        <a href="/showNewOwnerForm" class="btn btn-success">➕ Додати нового власника</a>
        <a href="/" class="btn btn-secondary">⬅ Назад</a>
    </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

ReportController1.java

```
package org.example.labb.controller;

import com.itextpdf.kernel.colors.ColorConstants;
import com.itextpdf.kernel.font.PdfFont;
import com.itextpdf.kernel.font.PdfFontFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.Cell;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;
import com.itextpdf.layout.properties.HorizontalAlignment;
import com.itextpdf.layout.properties.UnitValue;
import org.example.labb.service.HorseService;
import org.example.labb.service.JokeyService;
import org.example.labb.service.OwnerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.ClassPathResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
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 java.io.ByteArrayOutputStream;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

@Controller
public class ReportController1 {

    @Autowired
    private HorseService horseService;
    @Autowired
```

```
    private JokeyService jokeyService;
    @Autowired
    private OwnerService ownerService;

    @GetMapping("/reports")
    public String showReportsPage(Model model) {
        return "reports";
    }

    @PostMapping("/generateReport")
    public ResponseEntity<ByteArrayResource> generateSummaryReport() {
        try {
            long horseCount = horseService.getAllHorses().size();
            long jokeyCount = jokeyService.getAllJokeys().size();
            long ownerCount = ownerService.getAllOwners().size();

            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            PdfWriter writer = new PdfWriter(baos);
            PdfDocument pdf = new PdfDocument(writer);
            Document document = new Document(pdf);


            document.setMargins(20, 20, 20, 20);

            // Load Noto Sans fonts from resources
            PdfFont         font         =         PdfFontFactory.createFont(new
ClassPathResource("fonts/NotoSans-Regular.ttf").getFile().getAbsolutePath(),
PdfFontFactory.EmbeddingStrategy.PREFER_EMBEDDED);
            PdfFont        boldFont        =        PdfFontFactory.createFont(new
ClassPathResource("fonts/NotoSans-Bold.ttf").getFile().getAbsolutePath(),
PdfFontFactory.EmbeddingStrategy.PREFER_EMBEDDED);


            Paragraph title = new Paragraph("Звіт про скачки")
                    .setFont(boldFont)
                    .setFontSize(24)
                    .setHorizontalAlignment(HorizontalAlignment.CENTER)
                    .setMarginBottom(20);
            document.add(title);


            Paragraph subtitle = new Paragraph("Загальна статистика учасників")
                    .setFont(font)
                    .setFontSize(18)
                    .setHorizontalAlignment(HorizontalAlignment.CENTER)
                    .setMarginBottom(20);
            document.add(subtitle);


            Paragraph   date   =   new   Paragraph("Дата   формування   звіту:   "   +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss")))
                    .setFont(font)
                    .setFontSize(12)
                    .setHorizontalAlignment(HorizontalAlignment.RIGHT)
                    .setMarginBottom(20);
            document.add(date);


            float[] columnWidths = {150, 50}; // Category and Count columns
```

```
            Table table = new Table(UnitValue.createPercentArray(columnWidths));
            table.setWidth(UnitValue.createPercentValue(50));
            table.setHorizontalAlignment(HorizontalAlignment.CENTER);


            Cell          header1          =          new          Cell().add(new
Paragraph("Категорія").setFont(boldFont).setFontColor(ColorConstants.WHITE))
                    .setBackgroundColor(ColorConstants.DARK_GRAY)
                    .setHorizontalAlignment(HorizontalAlignment.CENTER);
            Cell          header2          =          new          Cell().add(new
Paragraph("Кількість").setFont(boldFont).setFontColor(ColorConstants.WHITE))
                    .setBackgroundColor(ColorConstants.DARK_GRAY)
                    .setHorizontalAlignment(HorizontalAlignment.CENTER);
            table.addHeaderCell(header1);
            table.addHeaderCell(header2);


            table.addCell(new Cell().add(new Paragraph("Коні").setFont(font)));
            table.addCell(new                                                Cell().add(new
Paragraph(String.valueOf(horseCount)).setFont(font).setHorizontalAlignment(Horiz
ontalAlignment.CENTER)));
            table.addCell(new Cell().add(new Paragraph("Жокеї").setFont(font)));
            table.addCell(new                                                Cell().add(new
Paragraph(String.valueOf(jokeyCount)).setFont(font).setHorizontalAlignment(Horiz
ontalAlignment.CENTER)));
            table.addCell(new                                                Cell().add(new
Paragraph("Власники").setFont(font)));
            table.addCell(new                                                Cell().add(new
Paragraph(String.valueOf(ownerCount)).setFont(font).setHorizontalAlignment(Horiz
ontalAlignment.CENTER)));

            document.add(table);


            document.close();

            ByteArrayResource              resource              =              new
ByteArrayResource(baos.toByteArray());

            return ResponseEntity.ok()
                    .header(HttpHeaders.CONTENT_DISPOSITION,         "attachment;
filename=summary_report.pdf")
                    .contentType(MediaType.APPLICATION_PDF)
                    .body(resource);
        } catch (Exception e) {
            throw new RuntimeException("Error generating PDF: " + e.getMessage(),
e);
        }
    }
}
```

OwnerController.java

```
package org.example.labb.controller;

import org.example.labb.entity.Owner;
import org.example.labb.service.OwnerService;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
public class OwnerController {
    @Autowired
    private OwnerService ownerService;

    @GetMapping("/owners")
    public String showOwners(Model model) {
        List<Owner> owners = ownerService.getAllOwners();
        model.addAttribute("owners", owners);
        return "owners";
    }

    @GetMapping("/showNewOwnerForm")
    public String showNewOwnerForm(Model model) {
        Owner owner = new Owner();
        model.addAttribute("owner", owner);
        return "new_owner";
    }

    @PostMapping("/saveOwner")
    public String saveOwner(@ModelAttribute("owner") Owner owner) {
        ownerService.createOwner(owner);
        return "redirect:/owners";
    }
    @GetMapping("/showFormForUpdateOwner/{id}")
    public String showFormForUpdateOwner(@PathVariable(value = "id") long id,
Model model) {
        Owner owner = ownerService.getOwnerById(id).orElseThrow(() -> new
RuntimeException("Owner not found"));
        model.addAttribute("owner", owner);
        return "update_owner";
    }
    @PostMapping("/updateOwner/{id}")
    public String updateOwner(@PathVariable(value = "id") int id,
@ModelAttribute("owner") Owner newOwner) {
        ownerService.updateOwner(id, newOwner);
        return "redirect:/owners";
    }
    @GetMapping("/deleteOwner/{id}")
    public String deleteOwner(@PathVariable(value = "id") int id) {
        this.ownerService.deleteOwner(id);
        return "redirect:/owners";
    }
}
```

new_horse.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Update Horse</title>
    <meta charset="UTF-8">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
```

```
rel="stylesheet">
    <style>
        body {
            background: linear-gradient(to right, #f8f9fa, #e3f2fd);
            font-family: 'Segoe UI', sans-serif;
        }
        .form-container {
            max-width: 500px;
            margin: 50px auto;
            padding: 30px;
            background-color: #ffffff;
            border-radius: 12px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
            color: #0d6efd;
            margin-bottom: 25px;
        }
    </style>
</head>
<body>
<div class="form-container">
    <h2>Додати нового коня</h2>
    <form th:action="@{/saveHorse}" th:object="${horse}" method="post">
        <div class="mb-3">
            <label class="form-label">Кличка:</label>
            <input type="text" class="form-control" th:field="*{nickname}" />
        </div>
        <div class="mb-3">
            <label class="form-label">Стать:</label>
            <input type="text" class="form-control" th:field="*{gender}" />
        </div>
        <div class="mb-3">
            <label class="form-label">Порода:</label>
            <input type="text" class="form-control" th:field="*{breed}" />
        </div>
        <div class="mb-3">
            <label class="form-label">Вік:</label>
            <input type="number" class="form-control" th:field="*{age}" />
        </div>
        <div class="mb-3">
            <label class="form-label">Власник:</label>
            <input type="text" class="form-control" th:field="*{owner}" />
        </div>
        <div class="d-grid">
            <button type="submit" class="btn btn-primary">Додати</button>
        </div>
    </form>
    <div class="text-center mt-3">
        <a href="/horses" class="text-decoration-none text-secondary">← Назад</a>
    </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```